CS 170: Algorithms
Lecture in a Minute

Simplex Implementation:

Start at a (feasible) vertex.
Lecture in a Minute

Simplex Implementation:
Start at a (feasible) vertex.
(defined by linear system \( A'x = [b', 0, \cdots, 0] \)).
Begin at origin. Move to better neighboring vertex.
Coordinate system: distance from tight constraints.
Vertex at origin in coordinate system.
\( O(mn) \) time to update linear system.
Until no better neighboring vertex.
Objective function in coordinate system is non-positive.
Dual Variables: new objective function!
Lecture in a Minute

Simplex Implementation:
Start at a (feasible) vertex.
(defined by linear system $A'x = [b', 0, \cdots, 0]$).

Begin at origin. Move to better neighboring vertex.
Coordinate system: distance from tight constraints.
Vertex at origin in coordinate system.
$O(mn)$ time to update linear system.

Until no better neighboring vertex.
Objective function in coordinate system is non-positive.
Dual Variables: new objective function!

Maximum flow.
“Greedy” augment path...
Except reverse old decisions..
Reverse residual capacities.
(Friday): Optimality?
No augmenting path $\implies$
$s - t$ cut size = flow value.
Find flow and $s - t$ cut with equal value!
Simplex algorithm.

Two tasks:

1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

\[
\begin{align*}
\text{max} & \quad c^T x \\
Ax & \leq b \\
x & \geq 0
\end{align*}
\]

Start at origin, supposing it is feasible.

Vertex since intersection of \( n \) constraints of form \( x_i = 0 \).

Optimal? If all \( c_i \leq 0 \) \( \Rightarrow \) increasing any \( x_i \) decreases value \( \Rightarrow \) optimal!

if there is \( c_i > 0 \) increasing \( x_i \) increases value \( \Rightarrow \) not optimal.

Done with task 1.
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

\[
\begin{align*}
\max & \quad c^T x \\
\text{s.t.} & \quad Ax \leq b \\
& \quad x \geq 0
\end{align*}
\]

Start at origin, supposing it is feasible.
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

\[
\begin{align*}
\max & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
& \quad x \geq 0
\end{align*}
\]

Start at origin, supposing it is feasible.

Vertex since intersection of \( n \) constraints of form \( x_i = 0 \).
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

\[ \max c^T x \]
\[ Ax \leq b \]
\[ x \geq 0 \]

Start at origin, supposing it is feasible.

Vertex since intersection of \( n \) constraints of form \( x_i = 0 \).

Optimal?
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

\[
\begin{align*}
\max & \quad c^T x \\
Ax & \leq b \\
& \quad x \geq 0
\end{align*}
\]

Start at origin, supposing it is feasible.

Vertex since intersection of \( n \) constraints of form \( x_i = 0 \).

Optimal?
If all \( c_i \leq 0 \)
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

$$\begin{align*}
\max & \quad c^T x \\
Ax & \leq b \\
& \quad x \geq 0
\end{align*}$$

Start at origin, supposing it is feasible.

Vertex since intersection of $n$ constraints of form $x_i = 0$.

Optimal?

If all $c_i \leq 0 \implies$ increasing any $x_i$ decreases value
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

\[
\begin{align*}
\text{max } & \mathbf{c}^T \mathbf{x} \\
\text{subject to } & \mathbf{A} \mathbf{x} \leq \mathbf{b} \\
& \mathbf{x} \geq \mathbf{0}
\end{align*}
\]

Start at origin, supposing it is feasible.

Vertex since intersection of \( n \) constraints of form \( x_i = 0 \).

Optimal?

If all \( c_i \leq 0 \) \( \implies \) increasing any \( x_i \) decreases value \( \implies \) optimal!
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

\[
\begin{align*}
\max & \quad c^T x \\
Ax & \leq b \\
x & \geq 0
\end{align*}
\]

Start at origin, supposing it is feasible.

Vertex since intersection of \( n \) constraints of form \( x_i = 0 \).

Optimal?
If all \( c_i \leq 0 \) \( \implies \) increasing any \( x_i \) decreases value \( \implies \) optimal!

if there is \( c_i > 0 \)
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

\[
\begin{align*}
\max & \quad c^T x \\
A x & \leq b \\
x & \geq 0
\end{align*}
\]

Start at origin, supposing it is feasible.

Vertex since intersection of \( n \) constraints of form \( x_i = 0 \).

Optimal?
If all \( c_i \leq 0 \) \( \implies \) increasing any \( x_i \) decreases value \( \implies \) optimal!

if there is \( c_i > 0 \) increasing \( x_i \) increases value
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

$$\max c^T x$$
$$Ax \leq b$$
$$x \geq 0$$

Start at origin, supposing it is feasible.

Vertex since intersection of $n$ constraints of form $x_i = 0$.

Optimal?
If all $c_i \leq 0 \implies$ increasing any $x_i$ decreases value $\implies$ optimal!
if there is $c_i > 0$ increasing $x_i$ increases value $\implies$ not optimal.
Simplex algorithm.

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

Canonical LP.

\[
\begin{align*}
\max \ & c^T x \\
A x & \leq b \\
x & \geq 0
\end{align*}
\]

Start at origin, supposing it is feasible.

Vertex since intersection of \( n \) constraints of form \( x_i = 0 \).

Optimal?

If all \( c_i \leq 0 \) \( \implies \) increasing any \( x_i \) decreases value \( \implies \) optimal!

if there is \( c_i > 0 \) increasing \( x_i \) increases value \( \implies \) not optimal.

Done with task 1.
Going to a better place..

Two tasks:

1. Check optimality of vertex?
2. Where to go next?

At origin, there is positive $c_i$, so increase $x_i$. ...until you hit another constraint. $x_i \geq 0$ is no longer tight, but new constraint is. $n$ constraints! At vertex!
Going to a better place..

Two tasks:
1. Check optimality of vertex?
2. Where to go next?
Two tasks:
1. Check optimality of vertex?
2. Where to go next?

At origin, there is positive \( c_i \), so increase \( x_i \).
Two tasks:
1. Check optimality of vertex?
2. Where to go next?

At origin, there is positive $c_i$, so increase $x_i$.
...until you hit another constraint.
Two tasks:
1. Check optimality of vertex?
2. Where to go next?

At origin, there is positive $c_i$, so increase $x_i$.

...until you hit another constraint.

$x_i \geq 0$ is no longer tight, but new constraint is.
Two tasks:
1. Check optimality of vertex?
2. Where to go next?

At origin, there is positive $c_i$, so increase $x_i$.

...until you hit another constraint.

$x_i \geq 0$ is no longer tight, but new constraint is.

$\implies n$ constraints!
Going to a better place..

Two tasks:
1. Check optimality of vertex?
2. Where to go next?

At origin, there is positive $c_i$, so increase $x_i$.
...until you hit another constraint.

$x_i \geq 0$ is no longer tight, but new constraint is.
⇒ $n$ constraints!

At vertex!
Example.

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]
Example.

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)} \\
\end{align*}
\]

Origin: feasible, value 0.
Example.

\[ \text{max } 2x_1 + 5x_2 \]
\[ 2x_1 - x_2 \leq 4 \] \hspace{1cm} 1
\[ x_1 + 2x_2 \leq 9 \] \hspace{1cm} 2
\[ -x_1 + x_2 \leq 3 \] \hspace{1cm} 3
\[ x_1 \geq 0 \] \hspace{1cm} 4
\[ x_2 \geq 0 \] \hspace{1cm} 5

Origin: feasible, value 0. Inequalities 4 and 5 are tight.
Example.

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 & 1 \\
x_1 + 2x_2 & \leq 9 & 2 \\
-x_1 + x_2 & \leq 3 & 3 \\
x_1 & \geq 0 & 4 \\
x_2 & \geq 0 & 5
\end{align*}
\]

Origin: feasible, value 0.
Inequalities 4 and 5 are tight.
Relax constraint \( x_2 = 0 \).
Example.

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)} 
\end{align*}
\]

Origin: feasible, value 0.
Inequalities (4) and (5) are tight.

Relax constraint \( x_2 = 0 \).
Increase \( x_2 \) until
Example.

\[
\begin{align*}
\text{max} & \quad 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
 x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

Origin: feasible, value 0.
Inequalities (4) and (5) are tight.

Relax constraint \( x_2 = 0 \).
Increase \( x_2 \) until
...Inequality (3) becomes tight constraint.
Example.

\[
\text{max } 2x_1 + 5x_2 \\
2x_1 - x_2 \leq 4 \\
x_1 + 2x_2 \leq 9 \\
-x_1 + x_2 \leq 3 \\
x_1 \geq 0 \\
x_2 \geq 0
\]

Origin: feasible, value 0. Inequalities 4 and 5 are tight.

Relax constraint \( x_2 = 0 \).
Increase \( x_2 \) until
...Inequality 3 becomes tight constraint.
...Tight constraints: 3 and 4.
Example.

\[
\begin{align*}
\text{max } & \quad 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

Origin: feasible, value 0.
Inequalities (4) and (5) are tight.

Relax constraint \(x_2 = 0\).
Increase \(x_2\) until
...Inequality (3) becomes tight constraint.
...Tight constraints: (3) and (4).
...new vertex: \((0,3)\) with value 15.
Example.

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad (1) \\
x_1 + 2x_2 & \leq 9 \quad (2) \\
-x_1 + x_2 & \leq 3 \quad (3) \\
x_1 & \geq 0 \quad (4) \\
x_2 & \geq 0 \quad (5)
\end{align*}
\]

Origin: feasible, value 0.
Inequalities (4) and (5) are tight.

Relax constraint \(x_2 = 0\).
Increase \(x_2\) until
...Inequality (3) becomes tight constraint.
...Tight constraints: (3) and (4).
...new vertex: \((0,3)\) with value 15.

Easy process from origin:
Example.

\[ \text{max } 2x_1 + 5x_2 \]
\[ 2x_1 - x_2 \leq 4 \quad 1 \]
\[ x_1 + 2x_2 \leq 9 \quad 2 \]
\[ -x_1 + x_2 \leq 3 \quad 3 \]
\[ x_1 \geq 0 \quad 4 \]
\[ x_2 \geq 0 \quad 5 \]

Origin: feasible, value 0.
Inequalities 4 and 5 are tight.

Relax constraint \( x_2 = 0 \).
Increase \( x_2 \) until
...Inequality 3 becomes tight constraint.
...Tight constraints: 3 and 4.
...new vertex: (0,3) with value 15.

Easy process from origin: just increase one variable with positive \( c \).
Example.

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\ 
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
 x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

Origin: feasible, value 0. Inequalities (4) and (5) are tight.

Relax constraint \(x_2 = 0\). Increase \(x_2\) until... Inequality (3) becomes tight constraint. ...Tight constraints: (3) and (4). ...new vertex: (0,3) with value 15.

Easy process from origin: just increase one variable with positive \(c\).
Now what?
A new coordinate system.

New coordinates: Distance from new tight constraints.
A new coordinate system.

New coordinates: Distance from new tight constraints.

Constraint 1  Constraint 2

\[ y_1 \] \[ y_2 \]

\[ x \]
A new coordinate system.

New coordinates: Distance from new tight constraints.

Constraint 1

Constraint 2

$y_1$ is distance from constraint $i$

$y_i$ is distance from constraint $i$
A new coordinate system.

New coordinates: Distance from new tight constraints.

Constraint 1

Constraint 2

$y_1$ is distance from constraint $i$

$x$ is at $(y_1, y_2)$ in new coordinate system.
A new coordinate system.

New coordinates: Distance from new tight constraints.

Constraint 1

Constraint 2

$y_i$ is distance from constraint $i$

$x$ is at $(y_1, y_2)$ in new coordinate system.

For constraint $i$: $y_i = b_i - a_i x$
A new coordinate system.

New coordinates: Distance from new tight constraints.

Constraint 1

Constraint 2

$y_i$ is distance from constraint $i$

$x$ is at $(y_1, y_2)$ in new coordinate system.

For constraint $i$: $y_i = b_i - a_i x$

Recall that for origin: $x_i$ was distance from constraint $x_i \geq 0$. 
A new coordinate system.

New coordinates: Distance from new tight constraints.

Constraint 1

Constraint 2

$y_i$ is distance from constraint $i$

$x$ is at $(y_1, y_2)$ in new coordinate system.

For constraint $i$: $y_i = b_i - a_i x$

Recall that for origin: $x_i$ was distance from constraint $x_i \geq 0$.

At origin in new coordinate system!
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max} & \quad 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1 \), \( y_2 = 3 + x_1 - x_2 \).
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max } & \quad 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1 \), \( y_2 = 3 + x_1 - x_2 \).

Solve for \( x_i \)’s: \( x_1 = y_1 \) and \( x_2 = 3 - y_2 + y_1 \).
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
\begin{align*}
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)}
\end{align*} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1, \ y_2 = 3 + x_1 - x_2 \).

Solve for \( x_i \)'s: \( x_1 = y_1 \) and \( x_2 = 3 - y_2 + y_1 \).

Plug in for \( x_1 \) and \( x_2 \):
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max} & \quad 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1 \), \( y_2 = 3 + x_1 - x_2 \).

Solve for \( x_i \)'s: \( x_1 = y_1 \) and \( x_2 = 3 - y_2 + y_1 \).

Plug in for \( x_1 \) and \( x_2 \): objective function
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max } & \quad 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1, \ y_2 = 3 + x_1 - x_2 \).

Solve for \( x_i \)'s: \( x_1 = y_1 \) and \( x_2 = 3 - y_2 + y_1 \).

Plug in for \( x_1 \) and \( x_2 \): objective function

\[
\begin{align*}
\text{max } & \quad 2x_1 + 5x_2 \\
\text{max } & \quad 2y_1 + 5(3 + y_1 - y_2 + y_1) \\
\end{align*}
\]
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\text{max } 2x_1 + 5x_2
\]
\[
2x_1 - x_2 \leq 4 \tag{1}
\]
\[
x_1 + 2x_2 \leq 9 \tag{2}
\]
\[
-x_1 + x_2 \leq 3 \tag{3}
\]
\[
x_1 \geq 0 \tag{4}
\]
\[
x_2 \geq 0 \tag{5}
\]

New variables: \(y_1 = x_1\), \(y_2 = 3 + x_1 - x_2\).

Solve for \(x_i\)'s: \(x_1 = y_1\) and \(x_2 = 3 - y_2 + y_1\).

Plug in for \(x_1\) and \(x_2\): objective function
\[
\text{max } 2x_1 + 5x_2
\]
\[
\text{max } 2(y_1) + 5(3 - y_2 + y_1)
\]

Are we optimal? Yes!

Maybe not!

No.

Positive coefficient for increasing \(y_1\).
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max} & \quad 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1, \ y_2 = 3 + x_1 - x_2 \).

Solve for \( x_i \)'s: \( x_1 = y_1 \) and \( x_2 = 3 - y_2 + y_1 \).

Plug in for \( x_1 \) and \( x_2 \): objective function

\[
\begin{align*}
\text{max} & \quad 2x_1 + 5x_2 \\
\text{max} & \quad 2(y_1) + 5(3 - y_2 + y_1) \\
\text{max} & \quad 15 + 7y_1 - 5y_2
\end{align*}
\]
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1 \), \( y_2 = 3 + x_1 - x_2 \).

Solve for \( x_i \)'s: \( x_1 = y_1 \) and \( x_2 = 3 - y_2 + y_1 \).

Plug in for \( x_1 \) and \( x_2 \): objective function

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
\text{max } & 2(y_1) + 5(3 - y_2 + y_1) \\
\text{max } & 15 + 7y_1 - 5y_2 \text{ Are we optimal?}
\end{align*}
\]
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \(y_1 = x_1\), \(y_2 = 3 + x_1 - x_2\).

Solve for \(x_i\)'s: \(x_1 = y_1\) and \(x_2 = 3 - y_2 + y_1\).

Plug in for \(x_1\) and \(x_2\): objective function

\[
\begin{align*}
\text{max } & 2x_1 + 5x_2 \\
\text{max } & 2(y_1) + 5(3 - y_2 + y_1) \\
\text{max } & 15 + 7y_1 - 5y_2 \text{ Are we optimal? Yes!}
\end{align*}
\]
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max} & \quad 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1 \), \( y_2 = 3 + x_1 - x_2 \).

Solve for \( x_i \)'s: \( x_1 = y_1 \) and \( x_2 = 3 - y_2 + y_1 \).

Plug in for \( x_1 \) and \( x_2 \): objective function

\[
\begin{align*}
\text{max} & \quad 2x_1 + 5x_2 \\
\text{max} & \quad 2(y_1) + 5(3 - y_2 + y_1) \\
\text{max} & \quad 15 + 7y_1 - 5y_2 \quad \text{Are we optimal? Yes! Maybe not!}
\end{align*}
\]
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\text{max } & \quad 2x_1 + 5x_2 \\
2x_1 - x_2 & \leq 4 \quad \text{(1)} \\
x_1 + 2x_2 & \leq 9 \quad \text{(2)} \\
-x_1 + x_2 & \leq 3 \quad \text{(3)} \\
x_1 & \geq 0 \quad \text{(4)} \\
x_2 & \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1, \ y_2 = 3 + x_1 - x_2 \).

Solve for \( x_i \)'s: \( x_1 = y_1 \) and \( x_2 = 3 - y_2 + y_1 \).

Plug in for \( x_1 \) and \( x_2 \): objective function

\[
\begin{align*}
\text{max } & \quad 2x_1 + 5x_2 \\
\text{max } & \quad 2(y_1) + 5(3 - y_2 + y_1) \\
\text{max } & \quad 15 + 7y_1 - 5y_2 \quad \text{Are we optimal? Yes! Maybe not! No.}
\end{align*}
\]
Rewrite linear program.

Rewrite linear program with new coordinates.

\[
\begin{align*}
\max & \quad 2x_1 + 5x_2 \\
& \quad 2x_1 - x_2 \leq 4 \quad \text{(1)} \\
& \quad x_1 + 2x_2 \leq 9 \quad \text{(2)} \\
& \quad -x_1 + x_2 \leq 3 \quad \text{(3)} \\
\end{align*}
\]

\[
\begin{align*}
&\quad x_1 \geq 0 \quad \text{(4)} \\
&\quad x_2 \geq 0 \quad \text{(5)}
\end{align*}
\]

New variables: \( y_1 = x_1, \ y_2 = 3 + x_1 - x_2 \).

Solve for \( x_i \)'s: \( x_1 = y_1 \) and \( x_2 = 3 - y_2 + y_1 \).

Plug in for \( x_1 \) and \( x_2 \): objective function
\[
\begin{align*}
\max & \quad 2x_1 + 5x_2 \\
& \quad 2(y_1) + 5(3 - y_2 + y_1) \\
& \quad 15 + 7y_1 - 5y_2 \quad \text{Are we optimal? Yes! Maybe not! No.}
\end{align*}
\]

Positive coefficient for increasing \( y_1 \).
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
       & \quad y_1 + y_2 \leq 7 \quad \text{①} \\
       & \quad 3y_1 - 2y_2 \leq 3 \quad \text{②} \\
       & \quad y_2 \geq 0 \quad \text{③} \\
       & \quad y_1 \geq 0 \quad \text{④} \\
       & \quad -y_1 + y_2 \leq 3 \quad \text{⑤}
\end{align*}
\]
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \quad (1) \\
3y_1 - 2y_2 & \leq 3 \quad (2) \\
y_2 & \geq 0 \quad (3)
\end{align*}
\]

\[
\begin{align*}
y_1 & \geq 0 \quad (4) \\
y_1 - y_2 & \leq 3 \quad (5)
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)
Rewriting example..

\[ \begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \quad \text{①} \\
3y_1 - 2y_2 & \leq 3 \quad \text{②} \\
y_2 & \geq 0 \quad \text{③} \\
y_1 & \geq 0 \quad \text{④} \\
-y_1 + y_2 & \leq 3 \quad \text{⑤}
\end{align*} \]

\( y_1, y_2 \) are non-negative just like \( x_i \)'s. (Constraints are satisfied!)

Improve by increasing \( y_1 \).
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
\text{subject to} & \quad y_1 + y_2 \leq 7 \quad \text{(1)} \\
& \quad 3y_1 - 2y_2 \leq 3 \quad \text{(2)} \\
& \quad y_2 \geq 0 \quad \text{(3)} \\
& \quad y_1 \geq 0 \quad \text{(4)} \\
& \quad -y_1 + y_2 \leq 3 \quad \text{(5)}
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

Which is tight?
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \quad (1) \\
3y_1 - 2y_2 & \leq 3 \quad (2) \\
y_2 & \geq 0 \quad (3) \\
y_1 & \geq 0 \quad (4) \\
-y_1 + y_2 & \leq 3 \quad (5)
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

Which is tight? \(\circlearrowleft\)?
Rewriting example..

\[
\begin{align*}
\text{max } & \quad 15 + 7y_1 - 5y_2 \\
& y_1 + y_2 \leq 7 \quad \textcircled{1} \\
& 3y_1 - 2y_2 \leq 3 \quad \textcircled{2} \\
& y_2 \geq 0 \quad \textcircled{3} \\
& y_1 \geq 0 \quad \textcircled{4} \\
& -y_1 + y_2 \leq 3 \quad \textcircled{5}
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

Which is tight? \textcircled{1}? \textcircled{2}?
Rewriting example..

$$\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
& y_1 + y_2 \leq 7 \quad (1) \\
& 3y_1 - 2y_2 \leq 3 \quad (2) \\
& y_2 \geq 0 \quad (3) \\
& y_1 \geq 0 \quad (4) \\
& -y_1 + y_2 \leq 3 \quad (5)
\end{align*}$$


$$y_1, y_2$$ are non-negative just like $$x_i$$’s. (Constraints are satisfied!)

Improve by increasing $$y_1$$.

Which is tight? (1)? (2)? (3)?
Rewriting example..

\[
\text{max } 15 + 7y_1 - 5y_2
\]

\[
y_1 + y_2 \leq 7 \quad \text{(1)}
\]
\[
3y_1 - 2y_2 \leq 3 \quad \text{(2)}
\]
\[
y_2 \geq 0 \quad \text{(3)}
\]
\[
y_1 \geq 0 \quad \text{(4)}
\]
\[
-y_1 + y_2 \leq 3 \quad \text{(5)}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

Which is tight? ①? ②? ③? ④?
Rewriting example..

$$\text{max } 15 + 7y_1 - 5y_2$$

$$y_1 + y_2 \leq 7 \quad ①$$

$$3y_1 - 2y_2 \leq 3 \quad ②$$

$$y_2 \geq 0 \quad ③$$

$$y_1 \geq 0 \quad ④$$

$$-y_1 + y_2 \leq 3 \quad ⑤$$

$y_1, y_2$ are non-negative just like $x_i$’s. (Constraints are satisfied!)

Improve by increasing $y_1$.

Which is tight? ①? ②? ③? ④? ⑤?
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
\phantom{\text{max} } & \quad y_1 + y_2 \leq 7 \quad \text{(1)} \\
\phantom{\text{max} } & \quad 3y_1 - 2y_2 \leq 3 \quad \text{(2)} \\
\phantom{\text{max} } & \quad y_2 \geq 0 \quad \text{(3)} \\
\phantom{\text{max} } & \quad y_1 \geq 0 \quad \text{(4)} \\
\phantom{\text{max} } & \quad -y_1 + y_2 \leq 3 \quad \text{(5)}
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

Which is tight? ①? ②? ③? ④? ⑤?

Note: \(y_2 = 0\).
Rewriting example..

\[ \max \quad 15 + 7y_1 - 5y_2 \]
\[ y_1 + y_2 \leq 7 \quad \text{(1)} \]
\[ 3y_1 - 2y_2 \leq 3 \quad \text{(2)} \]
\[ y_2 \geq 0 \quad \text{(3)} \]
\[ y_1 \geq 0 \quad \text{(4)} \]
\[ -y_1 + y_2 \leq 3 \quad \text{(5)} \]

\( y_1, y_2 \) are non-negative just like \( x_i \)'s. (Constraints are satisfied!)

Improve by increasing \( y_1 \).

Which is tight? ①? ②? ③? ④? ⑤?

Note: \( y_2 = 0 \).

Smallest right hand side divided by (positive) coefficient of \( y_2 \)!
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \quad (1) \\
3y_1 - 2y_2 & \leq 3 \quad (2) \\
y_2 & \geq 0 \quad (3) \\
y_1 & \geq 0 \quad (4) \\
-y_1 + y_2 & \leq 3 \quad (5)
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

**Which is tight?** 1? 2? 3? 4? 5?

Note: \(y_2 = 0\).

Smallest right hand side divided by (positive) coefficient of \(y_2\)!
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
& \quad y_1 + y_2 \leq 7 \\
& \quad 3y_1 - 2y_2 \leq 3 \\
& \quad y_2 \geq 0 \\
& \quad y_1 \geq 0 \\
& \quad -y_1 + y_2 \leq 3
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

Which is tight? ①? ②? ③? ④? ⑤?

Note: \(y_2 = 0\).

Smallest right hand side divided by (positive) coefficient of \(y_2\)!

Inequality ②!
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
& \quad y_1 + y_2 \leq 7 \quad (1) \\
& \quad 3y_1 - 2y_2 \leq 3 \quad (2) \\
& \quad y_2 \geq 0 \quad (3) \\
& \quad y_1 \geq 0 \quad (4) \\
& \quad -y_1 + y_2 \leq 3 \quad (5)
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

Which is tight? \(1\)? \(2\)? \(3\)? \(4\)? \(5\)?

Note: \(y_2 = 0\).

Smallest right hand side divided by (positive) coefficient of \(y_2\)!

Inequality \(2\)!

New vertex: tight constraints \(3\) and \(2\).
Rewriting example..

max \quad 15 + 7y_1 - 5y_2

\begin{align*}
    y_1 + y_2 & \leq 7 \\
    3y_1 - 2y_2 & \leq 3 \\
    y_2 & \geq 0 \\
    y_1 & \geq 0 \\
    -y_1 + y_2 & \leq 3
\end{align*}

\( y_1, y_2 \) are non-negative just like \( x_i \)'s. (Constraints are satisfied!)

Improve by increasing \( y_1 \).

Which is tight? ①? ②? ③? ④? ⑤?

Note: \( y_2 = 0 \).

Smallest right hand side divided by (positive) coefficient of \( y_2 \)!

Inequality ②!

New vertex: tight constraints ③ and ②.

New solution: \( y_1 = 1, y_2 = 0 \).
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \quad (1) \\
3y_1 - 2y_2 & \leq 3 \quad (2) \\
y_2 & \geq 0 \quad (3) \\
y_1 & \geq 0 \quad (4) \\
-y_1 + y_2 & \leq 3 \quad (5)
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)’s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

Which is tight? ①? ②? ③? ④? ⑤?

Note: \(y_2 = 0\).

Smallest right hand side divided by (positive) coefficient of \(y_2\)!

Inequality ②!

New vertex: tight constraints ③ and ②.

New solution: \(y_1 = 1, y_2 = 0\). New Objective Value:
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
& \quad y_1 + y_2 \leq 7 \quad (1) \\
& \quad 3y_1 - 2y_2 \leq 3 \quad (2) \\
& \quad y_2 \geq 0 \quad (3) \\
& \quad y_1 \geq 0 \quad (4) \\
& \quad -y_1 + y_2 \leq 3 \quad (5)
\end{align*}
\]

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

Which is tight? ①? ②? ③? ④? ⑤?

Note: \(y_2 = 0\).

Smallest right hand side divided by (positive) coefficient of \(y_2\)!

Inequality ②!

New vertex: tight constraints ③ and ②.

New solution: \(y_1 = 1, y_2 = 0\). New Objective Value: 
\[12 + 7(1) - 5(0)\]
Rewriting example..

$$\text{max} \quad 15 + 7y_1 - 5y_2$$

$$y_1 + y_2 \leq 7 \quad \textcircled{1}$$

$$3y_1 - 2y_2 \leq 3 \quad \textcircled{2}$$

$$y_2 \geq 0 \quad \textcircled{3}$$

$$y_1 \geq 0 \quad \textcircled{4}$$

$$-y_1 + y_2 \leq 3 \quad \textcircled{5}$$

\(y_1, y_2\) are non-negative just like \(x_i\)'s. (Constraints are satisfied!)

Improve by increasing \(y_1\).

**Which is tight? \(\textcircled{1}\)? \(\textcircled{2}\)? \(\textcircled{3}\)? \(\textcircled{4}\)? \(\textcircled{5}\)?**

Note: \(y_2 = 0\).

Smallest right hand side divided by (positive) coefficient of \(y_2\)!

Inequality \(\textcircled{2}\)!

New vertex: tight constraints \(\textcircled{3}\) and \(\textcircled{2}\).

New solution: \(y_1 = 1, y_2 = 0\). New Objective Value:

\(12 + 7(1) - 5(0) = 22\).
Rewriting example..

\[
\begin{align*}
\text{max } & 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \\
3y_1 - 2y_2 & \leq 3 \\
y_2 & \geq 0 \\
y_1 & \geq 0 \\
-y_1 + y_2 & \leq 3 \\
\end{align*}
\]

Optimal? Yes! Maybe not! Optimal point!

Increasing \(z_1, z_2\) makes things worse.
Rewriting example.

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \quad (1) \\
3y_1 - 2y_2 & \leq 3 \quad (2) \\
y_2 & \geq 0 \quad (3) \\
y_1 & \geq 0 \quad (4) \\
-y_1 + y_2 & \leq 3 \quad (5)
\end{align*}
\]

Rewrite: \( z_2 = y_2 \)
\( z_1 = 3 - 3y_1 + 2y_2 \)
Rewriting example..

\[
\begin{align*}
\text{max } & 15 + 7y_1 - 5y_2 \\
& y_1 + y_2 \leq 7 \quad \text{(1)} \\
& 3y_1 - 2y_2 \leq 3 \quad \text{(2)} \\
& y_2 \geq 0 \quad \text{(3)} \\
& y_1 \geq 0 \quad \text{(4)} \\
& -y_1 + y_2 \leq 3 \quad \text{(5)}
\end{align*}
\]

Rewrite: \( z_2 = y_2 \)
\( z_1 = 3 - 3y_1 + 2y_2 \rightarrow \)
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \quad (1) \\
3y_1 - 2y_2 & \leq 3 \quad (2) \\
y_2 & \geq 0 \quad (3) \\
y_1 & \geq 0 \quad (4) \\
-y_1 + y_2 & \leq 3 \quad (5)
\end{align*}
\]

Rewrite: \( z_2 = y_2 \)
\[
z_1 = 3 - 3y_1 + 2y_2 \rightarrow y_1 = -\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1
\]

Objective function.
Rewriting example..

\[ \text{max } 15 + 7y_1 - 5y_2 \]
\[ y_1 + y_2 \leq 7 \] ①
\[ 3y_1 - 2y_2 \leq 3 \] ②
\[ y_2 \geq 0 \] ③
\[ y_1 \geq 0 \] ④
\[ -y_1 + y_2 \leq 3 \] ⑤

Rewrite: \( z_2 = y_2 \)
\( z_1 = 3 - 3y_1 + 2y_2 \rightarrow y_1 = -\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1 \)
Objective function.
\[ \text{max } 15 + 7(\frac{-1}{3}z_1 + \frac{2}{3}z_2 + 1) - 5z_2 \]
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \\
3y_1 - 2y_2 & \leq 3 \\
y_2 & \geq 0 \\
y_1 & \geq 0 \\
-y_1 + y_2 & \leq 3
\end{align*}
\]

\[
\text{Rewrite: } z_2 = y_2 \\
z_1 = 3 - 3y_1 + 2y_2 \rightarrow y_1 = -\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1
\]

Objective function.

\[
\begin{align*}
\text{max} & \quad 15 + 7(-\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1) - 5z_2 \\
\text{max} & \quad 22 - \frac{7}{3}z_1 - \frac{1}{3}z_2
\end{align*}
\]
Rewriting example..

\[
\begin{align*}
\max & \quad 15 + 7y_1 - 5y_2 \\
& \quad y_1 + y_2 \leq 7 \\
& \quad 3y_1 - 2y_2 \leq 3 \\
& \quad y_2 \geq 0 \\
& \quad y_1 \geq 0 \\
& \quad -y_1 + y_2 \leq 3
\end{align*}
\]

Rewrite: \( z_2 = y_2 \)
\[z_1 = 3 - 3y_1 + 2y_2 \rightarrow y_1 = -\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1\]

Objective function.
\[
\begin{align*}
\max & \quad 15 + 7(-\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1) - 5z_2 \\
\max & \quad 22 - \frac{7}{3}z_1 - \frac{1}{3}z_2 \text{ Optimal?}
\end{align*}
\]
Rewriting example..

\[
\begin{align*}
\text{max} & \quad 15 + 7y_1 - 5y_2 \\
& y_1 + y_2 \leq 7 \quad \text{①} \\
& 3y_1 - 2y_2 \leq 3 \quad \text{②} \\
& y_2 \geq 0 \quad \text{③} \\
& y_1 \geq 0 \quad \text{④} \\
& -y_1 + y_2 \leq 3 \quad \text{⑤}
\end{align*}
\]

Rewrite: \( z_2 = y_2 \)
\( z_1 = 3 - 3y_1 + 2y_2 \rightarrow y_1 = -\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1 \)

Objective function.
\[
\begin{align*}
\text{max} & \quad 15 + 7(-\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1) - 5z_2 \\
\text{max} & \quad 22 - \frac{7}{3}z_1 - \frac{1}{3}z_2 \quad \text{Optimal? Yes!}
\end{align*}
\]
Rewriting example..

\[
\begin{align*}
\text{max } & 15 + 7y_1 - 5y_2 \\
y_1 + y_2 & \leq 7 \quad (1) \\
3y_1 - 2y_2 & \leq 3 \quad (2) \\
y_2 & \geq 0 \quad (3) \\
y_1 & \geq 0 \quad (4) \\
-y_1 + y_2 & \leq 3 \quad (5)
\end{align*}
\]

Rewrite: \( z_2 = y_2 \)
\[
\begin{align*}
z_1 = 3 - 3y_1 + 2y_2 \rightarrow y_1 = -\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1 \\
\text{Objective function.}
\end{align*}
\]

\[
\begin{align*}
\text{max } & 15 + 7(-\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1) - 5z_2 \\
\text{max } & 22 - \frac{7}{3}z_1 - \frac{1}{3}z_2 \text{ Optimal? Yes! Maybe not!}
\end{align*}
\]
Rewriting example..

\[
\begin{align*}
\text{max } & 15 + 7y_1 - 5y_2 \\
& y_1 + y_2 \leq 7 \quad \text{(1)} \\
& 3y_1 - 2y_2 \leq 3 \quad \text{(2)} \\
& y_2 \geq 0 \quad \text{(3)} \\
& y_1 \geq 0 \quad \text{(4)} \\
& -y_1 + y_2 \leq 3 \quad \text{(5)}
\end{align*}
\]

Rewrite: \( z_2 = y_2 \)

\( z_1 = 3 - 3y_1 + 2y_2 \rightarrow y_1 = -\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1 \)

Objective function.

\[
\begin{align*}
\text{max } & 15 + 7(-\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1) - 5z_2 \\
& 22 - \frac{7}{3}z_1 - \frac{1}{3}z_2 \quad \text{Optimal? Yes! Maybe not! Optimal point!}
\end{align*}
\]
Rewriting example.

\[ \text{max } 15 + 7y_1 - 5y_2 \]
\[ y_1 + y_2 \leq 7 \quad (1) \]
\[ 3y_1 - 2y_2 \leq 3 \quad (2) \]
\[ y_2 \geq 0 \quad (3) \]
\[ y_1 \geq 0 \quad (4) \]
\[ -y_1 + y_2 \leq 3 \quad (5) \]

Rewrite: \( z_2 = y_2 \)
\[ z_1 = 3 - 3y_1 + 2y_2 \rightarrow y_1 = -\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1 \]

Objective function.
\[ \text{max } 15 + 7\left(-\frac{1}{3}z_1 + \frac{2}{3}z_2 + 1\right) - 5z_2 \]
\[ \text{max } 22 - \frac{7}{3}z_1 - \frac{1}{3}z_2 \text{ Optimal? Yes! Maybe not! Optimal point!} \]
Increasing \( z_1, z_2 \) makes things worse.
Review.

In each step:

\[
\begin{align*}
&\max 15 + 7y_1 - 5y_2 \\
&3y_1 - 2y_2 \leq 3
\end{align*}
\]
In each step:

LP in coordinate system from tight constraints.
In each step:
LP in coordinate system from tight constraints.
Optimal?

\[
\begin{align*}
\max & \quad 15 + 7y_1 - 5y_2 \\
\text{s.t.} & \quad 3y_1 - 2y_2 \leq 3
\end{align*}
\]
In each step:
LP in coordinate system from tight constraints.
Optimal?
Does objective function have nonnegative multiplier?
In each step:
LP in coordinate system from tight constraints.
Optimal?
Does objective function have nonnegative multiplier?

\[
\max 15 + 7y_1 - 5y_2.
\]
In each step:
LP in coordinate system from tight constraints.
Optimal?
Does objective function have nonnegative multiplier?
\[ \max 15 + 7y_1 - 5y_2. \]
Go to tight constraint along improving coordinate.
In each step:
LP in coordinate system from tight constraints.
Optimal?
Does objective function have nonnegative multiplier?
\[
\max 15 + 7y_1 - 5y_2.
\]
Go to tight constraint along improving coordinate.
\[
3y_1 - 2y_2 \leq 3.
\]
In each step:
LP in coordinate system from tight constraints.
Optimal?
Does objective function have nonnegative multiplier?
\[
\max 15 + 7y_1 - 5y_2.
\]
Go to tight constraint along improving coordinate.
\[
3y_1 - 2y_2 \leq 3.
\]
Express LP in coordinate system for new tight constraints.
In each step:
LP in coordinate system from tight constraints.
Optimal?
Does objective function have nonnegative multiplier?
\[ \max 15 + 7y_1 - 5y_2. \]
Go to tight constraint along improving coordinate.
\[ 3y_1 - 2y_2 \leq 3. \]
Express LP in coordinate system for new tight constraints.
See previous slides!
Review.

In each step:
LP in coordinate system from tight constraints.
Optimal?
Does objective function have nonnegative multiplier?
\[ \max 15 + 7y_1 - 5y_2. \]
Go to tight constraint along improving coordinate.
\[ 3y_1 - 2y_2 \leq 3. \]
Express LP in coordinate system for new tight constraints.
See previous slides!
Repeat.
Details: getting started.

What if origin is not feasible?
Details: getting started.

What if origin is not feasible?

How do you find a feasible vertex?
What if origin is not feasible?
How do you find a feasible vertex?
An $x$ where $Ax \leq b$ and at vertex.
Details: getting started.

What if origin is not feasible?
How do you find a feasible vertex?
An $x$ where $Ax \leq b$ and at vertex.
Make a new linear program.
What if origin is not feasible?

How do you find a feasible vertex?

An $x$ where $Ax \leq b$ and at vertex.

Make a new linear program.

Introduce positive variables $z_i$ for inequality $i$. 
Details: getting started.

What if origin is not feasible?
How do you find a feasible vertex?
An $x$ where $Ax \leq b$ and at vertex.
Make a new linear program.
Introduce positive variables $z_i$ for inequality $i$.
Constraints: $a_i x - z_i \leq b_i$. 
What if origin is not feasible?
How do you find a feasible vertex?
An $x$ where $Ax \leq b$ and at vertex.
Make a new linear program.
Introduce positive variables $z_i$ for inequality $i$.
Constraints: $a_i x - z_i \leq b_i$.

$$\max \sum -z_i.$$
What if origin is not feasible?

How do you find a feasible vertex?

An $x$ where $Ax \leq b$ and at vertex.

Make a new linear program.

Introduce positive variables $z_i$ for inequality $i$.

Constraints: $a_i x - z_i \leq b_i$.

$$\max \sum -z_i.$$ 

Vertex solution $(x, z)$ of value zero
What if origin is not feasible?
How do you find a feasible vertex?
An $x$ where $Ax \leq b$ and at vertex.
Make a new linear program.
Introduce positive variables $z_i$ for inequality $i$.
Constraints: $a_i x - z_i \leq b_i$.

$$\max \sum -z_i.$$ 

Vertex solution $(x, z)$ of value zero
\[\Rightarrow\text{ all } z\text{'s are zero}\]
What if origin is not feasible?
How do you find a feasible vertex?
An $x$ where $Ax \leq b$ and at vertex.
Make a new linear program.
Introduce positive variables $z_i$ for inequality $i$.
Constraints: $a_i x - z_i \leq b_i$.
$$\max \sum -z_i.$$ 
Vertex solution $(x, z)$ of value zero
$\implies$ all $z$’s are zero
$\implies$ all inequalities are satisfied
What if origin is not feasible?
How do you find a feasible vertex?
An $x$ where $Ax \leq b$ and at vertex.
Make a new linear program.
Introduce positive variables $z_i$ for inequality $i$.
Constraints: $a_i x - z_i \leq b_i$.
$$\max \sum -z_i.$$ 
Vertex solution $(x, z)$ of value zero
$\implies$ all $z$’s are zero
$\implies$ all inequalities are satisfied
$\implies$ $x$ is a feasible vertex of $Ax \leq b$. 
Degeneracy.

Degenerate vertices.
Degeneracy.

Degenerate vertices.
Intersection of more than \( n \) constraints.

Feasible
Degeneracy.

Degenerate vertices.
Intersection of more than $n$ constraints.

Problem: all neighboring vertices are no better.
Degeneracy.

Degenerate vertices. Intersection of more than $n$ constraints.

Problem: all neighboring vertices are no better.

Infinite looping: Bland's anticycling rule.
Degeneracy.

Degenerate vertices.
Intersection of more than $n$ constraints.

Problem: all neighboring vertices are no better.
Infinite looping: Bland’s anticycling rule.
Or Perturb problem a bit.
Degeneracy.

Degenerate vertices.
Intersection of more than $n$ constraints.

Problem: all neighboring vertices are no better.
Infinite looping: Bland’s anticycling rule.
Or Perturb problem a bit. Unlikely to intersect!
Unboundedness.

Simplex can tell difference. From X: either unbounded improvement or optimal.
Unboundedness.

Unbounded value.
Unboundedness.

Simplex can tell difference.
From X: either unbounded improvement or optimal.
Running Time

Check optimality? $O(n)$. 
Running Time

Check optimality? $O(n)$.

Find tight constraint:
Running Time

Check optimality? $O(n)$.  
Find tight constraint:  
$O(m)$ constraints.
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.
Recall $y_i = b_i - a_i x$.
Rewrite in terms of $y_i$.
Solve for $x_i$ in terms of $y_i$.
Plug in.
Naively: $O(n^3)$ time.
Only one new constraint.
Have $x_i$ in terms of $y_1, ... , y_n$.
Only one $y_i$ goes to $y_i'$.
$O(nm)$ time to update LP. (Like backsolving.)
How many simplex steps?
Could be large.
Exponential in worst case!
Fast, in practice!
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint. $O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$. 
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$.
  Solve for $x_i$ in terms of $y_i$. 
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$.
  - Solve for $x_i$ in terms of $y_i$.
  - Plug in.
Running Time

Check optimality? $O(n)$. 

Find tight constraint: 
$O(m)$ constraints. $O(1)$ time per constraint. 
$O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$.
  Solve for $x_i$ in terms of $y_i$.
  Plug in.

Naively: $O(n^3)$ time.
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$.
   Solve for $x_i$ in terms of $y_i$.
   Plug in.

Naively: $O(n^3)$ time.

Only one new constraint.
Running Time

Check optimality? \( O(n) \).

Find tight constraint:
\( O(m) \) constraints. \( O(1) \) time per constraint. \( O(m) \) total.

Find new coordinate system, rewrite LP.

Recall \( y_i = b_i - a_i x \)

Rewrite in terms of \( y_i \).
- Solve for \( x_i \) in terms of \( y_i \).
- Plug in.

Naively: \( O(n^3) \) time.

Only one new constraint. Have \( x \) in terms of \( y_1, \ldots, y_n \).
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint. $O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$.
  Solve for $x_i$ in terms of $y_i$.
  Plug in.

Naively: $O(n^3)$ time.

Only one new constraint. Have $x$ in terms of $y_1, \ldots, y_n$.
Only one $y_i$ goes to $y'_i$. 
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$.
  Solve for $x_i$ in terms of $y_i$.
  Plug in.

Naively: $O(n^3)$ time.

Only one new constraint. Have $x$ in terms of $y_1, \ldots, y_n$.
Only one $y_i$ goes to $y'_i$.
$O(nm)$ time to update LP. (Like backsolving.)
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$.
  Solve for $x_i$ in terms of $y_i$.
  Plug in.

Naively: $O(n^3)$ time.

Only one new constraint. Have $x$ in terms of $y_1, \ldots, y_n$.
Only one $y_i$ goes to $y'_i$.
$O(nm)$ time to update LP. (Like backsolving.)

How many simplex steps?
Running Time

Check optimality? $O(n)$.  

Find tight constraint: 
$O(m)$ constraints. $O(1)$ time per constraint. 
$O(m)$ total.  

Find new coordinate system, rewrite LP.  

Recall $y_i = b_i - a_i x$  

Rewrite in terms of $y_i$.
  Solve for $x_i$ in terms of $y_i$.  
  Plug in.  

Naively: $O(n^3)$ time.  

Only one new constraint. Have $x$ in terms of $y_1, \ldots, y_n$.  
Only one $y_i$ goes to $y'_i$.  
$O(nm)$ time to update LP. (Like backsolving.)  

How many simplex steps?  
Could be large.
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$.
  Solve for $x_i$ in terms of $y_i$.
  Plug in.

Naively: $O(n^3)$ time.

Only one new constraint. Have $x$ in terms of $y_1, \ldots, y_n$.
Only one $y_i$ goes to $y'_i$.
$O(nm)$ time to update LP. (Like backsolving.)

How many simplex steps?
Could be large. Exponential in worst case!
Running Time

Check optimality? $O(n)$.

Find tight constraint:
$O(m)$ constraints. $O(1)$ time per constraint.
$O(m)$ total.

Find new coordinate system, rewrite LP.

Recall $y_i = b_i - a_i x$

Rewrite in terms of $y_i$.
  Solve for $x_i$ in terms of $y_i$.
  Plug in.

Naively: $O(n^3)$ time.

Only one new constraint. Have $x$ in terms of $y_1, \ldots, y_n$.
Only one $y_i$ goes to $y'_i$.
$O(nm)$ time to update LP. (Like backsolving.)

How many simplex steps?
Could be large. Exponential in worst case!
Fast, in practice!
Extra: Where’s the dual?

The negations of coefficients of new function!
Extra: Where’s the dual?

The negations of coefficients of new function!
Let $A'$ be matrix of “tight constraints.”
Extra: Where’s the dual?

The negations of coefficients of new function!
Let $A'$ be matrix of “tight constraints.”
Coordinate System: $y = b' - A'x$. 
Extra: Where’s the dual?

The negations of coefficients of new function!
Let $A'$ be matrix of “tight constraints.”
Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$
The negations of coefficients of new function!

Let $A'$ be matrix of “tight constraints.”

Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$

$max cx = max c((A')^{-1})(b' - y) = max c((A')^{-1})b') - (c(A')^{-1})y$. 
Extra: Where’s the dual?

The negations of coefficients of new function!
Let $A'$ be matrix of “tight constraints.”
Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$
$max cx = max c((A')^{-1})(b' - y) = max c((A')^{-1})b' - (c(A')^{-1})y.$
$z = ((A')^{-1})^T c$ gives coefficients of new objective function.
Extra: Where’s the dual?

The negations of coefficients of new function!

Let $A'$ be matrix of “tight constraints.”

Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$

$\max cx = \max c((A')^{-1})(b' - y) = \max c((A')^{-1})b' - (c(A')^{-1})y$.

$z = ((A')^{-1})^Tc$ gives coefficients of new objective function.

All positive at optimal!
Extra: Where’s the dual?

The negations of coefficients of new function!

Let $A'$ be matrix of “tight constraints.”

Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$

$max \ c x = max \ c((A')^{-1})(b' - y) = max \ c((A')^{-1})b' - (c(A')^{-1})y$.

$z = ((A')^{-1})^T c$ gives coefficients of new objective function.

All positive at optimal! $\rightarrow z \geq 0$
Extra: Where’s the dual?

The negations of coefficients of new function!

Let $A'$ be matrix of “tight constraints.”

Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$

$$\max c x = \max c((A')^{-1}) (b' - y) = \max c((A')^{-1}) b' - (c(A')^{-1}) y.$$ 

$z = ((A')^{-1})^T c$ gives coefficients of new objective function.

All positive at optimal! $\rightarrow z \geq 0$

$$A'^T z = A^T((A')^{-1})^T c = c$$ for subset of tight equations.
The negations of coefficients of new function!

Let $A'$ be matrix of “tight constraints.”

Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$

$\max cx = \max c((A')^{-1})(b' - y) = \max c((A')^{-1})b' - (c(A')^{-1})y$.

$z = ((A')^{-1})^Tc$ gives coefficients of new objective function.

All positive at optimal! $\rightarrow z \geq 0$

$A'^Tz = A^T((A')^{-1})^Tc = c$ for subset of tight equations.
Extra: Where’s the dual?

The negations of coefficients of new function!
Let $A'$ be matrix of “tight constraints.”
Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$
$max cx = \max c((A')^{-1})(b' - y) = \max c((A')^{-1})b') - (c(A')^{-1})y.$
$z = ((A')^{-1})^{T} c$ gives coefficients of new objective function.
All positive at optimal! $\rightarrow z \geq 0$
$A'^{T}z = A^{T}((A')^{-1})^{T} c = c$ for subset of tight equations.
$\rightarrow A'^{T}z \geq c.$
Extra: Where’s the dual?

The negations of coefficients of new function!
Let $A'$ be matrix of “tight constraints.”
Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$
$max cx = max c((A')^{-1})(b' - y) = max c((A')^{-1})b' - (c(A')^{-1})y.$
$z = ((A')^{-1})^T c$ gives coefficients of new objective function.
All positive at optimal! $\rightarrow z \geq 0$
$A'^T z = A^T ((A')^{-1})^T c = c$ for subset of tight equations.
$\rightarrow A'^T z \geq c.$
Set all other dual variables to 0. $\implies A^T z \geq c.$
The negations of coefficients of new function!

Let $A'$ be matrix of “tight constraints.”

Coordinate System: $y = b' - A'x$. $x = (A')^{-1}(b' - y)$

$max cx = max c((A')^{-1})(b' - y) = max c((A')^{-1})b' - (c(A')^{-1})y.$

$z = ((A')^{-1})^T c$ gives coefficients of new objective function.

All positive at optimal! $\rightarrow z \geq 0$

$A'^T z = A^T ((A')^{-1})^T c = c$ for subset of tight equations.

$\rightarrow A'^T z \geq c.$

Set all other dual variables to 0. $\implies A^T z \geq c.$

Feasible!
Next Up: Maximum Flow.

Maximum Flow.
Maximum flow

Flow network $G = (V, E)$, source $s$, sink $t \in V$, capacities $c_e > 0$. 
Maximum flow

Flow network $G = (V, E)$, source $s$, sink $t \in V$, capacities $c_e > 0$. 

Find Flow:

1. $0 \leq f_e \leq c_e$. "Capacity constraints."

2. If $u$ is not $s$ or $t$, $\sum (w, u) \in E f_wu = \sum (u, w) \in E f_uw$.

3. $f_s + f_{sc} = 1 + 3 = 4$.

Maximize: size $(f) = \sum (s, u) \in E f_{su}$.

$4 = c_{ad} + c_{cd} + c_{et}$.

Any $s-t$ cut gives an upper bound.
Maximum flow

Flow network $G = (V, E)$, source $s$, sink $t \in V$, capacities $c_e > 0$.

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. If $u$ is not $s$ or $t$
   \[
   \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}.
   \]

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$.
Maximum flow

Flow network $G = (V, E)$, source $s$, sink $t \in V$, capacities $c_e > 0$.

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. If $u$ is not $s$ or $t$
   \[
   \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}.
   \]

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. 
Maximum flow

Flow network \( G = (V, E) \), source \( s \), sink \( t \in V \), capacities \( c_e > 0 \).

Find Flow: \( f_e \)

1. \( 0 \leq f_e \leq c_e \). “Capacity constraints.”

2. If \( u \) is not \( s \) or \( t \)
   
   \[ \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}. \]

Maximize: \( \text{size}(f) = \sum_{(s,u) \in E} f_{su}. \)
Maximum flow

Flow network $G = (V, E)$, source $s$, sink $t \in V$, capacities $c_e > 0$.

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$
   \[ \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}. \]

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. 
Maximum flow

Flow network $G = (V, E)$, source $s$, sink $t \in V$, capacities $c_e > 0$.

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$

   $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1$.

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. 
Maximum flow

Flow network $G = (V, E)$, source $s$, sink $t \in V$, capacities $c_e > 0$.

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$
   \[ \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw} \cdot 3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1. \]

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$
Maximum flow

Flow network \( G = (V, E) \), source \( s \), sink \( t \in V \), capacities \( c_e > 0 \).

Find Flow: \( f_e \)

1. \( 0 \leq f_e \leq c_e \). “Capacity constraints.” \( 3 = f_{s,c} \leq c_{s,c} = 4 \).

2. If \( u \) is not \( s \) or \( t \)

\[
\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}. \quad 3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.
\]

Maximize: \( \text{size}(f) = \sum_{(s,u) \in E} f_{su}. \quad f_{sa} + f_{sc} = 1 + 3 = 4 \)

Optimal?
Maximum flow

Flow network $G = (V, E)$, source $s$, sink $t \in V$, capacities $c_e > 0$.

![Flow network diagram]

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$

$$\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw} \cdot 3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.$$

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su} \cdot f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal?
Maximum flow

Flow network \( G = (V, E) \), source \( s \), sink \( t \in V \), capacities \( c_e > 0 \).

Find Flow: \( f_e \)

1. \( 0 \leq f_e \leq c_e \). “Capacity constraints.” \( 3 = f_{s,c} \leq c_{s,c} = 4 \).

2. If \( u \) is not \( s \) or \( t \)
   \[ \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw} \cdot 3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1. \]

Maximize: \( \text{size}(f) = \sum_{(s,u) \in E} f_{su} \cdot f_{sa} + f_{sc} = 1 + 3 = 4 \)

Optimal? \( c_{ad} + c_{cd} + c_{et} = 1 + 1 + 2 = 4. \)
Maximum flow

Flow network $G = (V, E)$, source $s$, sink $t \in V$, capacities $c_e > 0$.

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$

$$\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}. \quad 3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.$$

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}. \quad f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal? $c_{ad} + c_{cd} + c_{et} = 1 + 1 + 2 = 4$.

Any $s-t$ cut gives an upper bound.
An $s - t$ cut is a partition of $V$ into $S$ and $T$ where $s \in S$ and $t \in T$. Its capacity is the total capacity of edges from $S$ to $T$. 
Do you know the definition?

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. If $u$ is not $s$ or $t$
\[
\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}.
\]
Do you know the definition?

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. If $u$ is not $s$ or $t$
   \[ \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}. \]

Valid or Invalid?
Do you know the definition?

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. If $u$ is not $s$ or $t$

   $$\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}.$$

Valid or Invalid?
Do you know the definition?

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. If $u$ is not $s$ or $t$
   \[
   \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}.
   \]

Valid or Invalid?

```
 s  b
 \downarrow  \uparrow  \downarrow  \uparrow  \downarrow  \uparrow
 3  3  2  2  2  2
 c  d  t
```

1 $\neq$ 2
Do you know the definition?

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. If $u$ is not $s$ or $t$
   \[
   \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}.
   \]

Valid or Invalid?
Do you know the definition?

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. If $u$ is not $s$ or $t$
   \[ \sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}. \]

Valid or Invalid?

\[ 2 + 1 \neq 2 \]
FindFlow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.

3. maximize $\sum_{su} f_{su}$.
Algorithms.

FindFlow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.

3. maximize $\sum_{su} f_{su}$.

Linear program!
Algorithms.

FindFlow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”
2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables $f_e$, linear constraints, linear optimization function.
Algorithms.

FindFlow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.

3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables $f_e$, linear constraints, linear optimization function.

Cool!
FindFlow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.

3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables $f_e$, linear constraints, linear optimization function.

Cool!

Note...
Algorithms.

FindFlow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”
2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables $f_e$, linear constraints, linear optimization function.

Cool!

Note...

Integer? (Given integer capacities.)
Algorithms.

FindFlow: \( f_e \)

1. \( 0 \leq f_e \leq c_e \). “Capacity constraints.”
2. \( \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw} \).
3. maximize \( \sum_{su} f_{su} \).

Linear program!
Variables \( f_e \), linear constraints, linear optimization function.

Cool!

Note...

Integer? (Given integer capacities.)
Yes. There is an integer vertex solution!
Algorithms.

FindFlow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”

2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.

3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables $f_e$, linear constraints, linear optimization function.

Cool!

Note...

Integer? (Given integer capacities.)

Yes. There is an integer vertex solution!

Constraint matrix has every subdeterminant being 1, 0, −1.
Algorithms.

FindFlow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”
2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!
Variables $f_e$, linear constraints, linear optimization function.
Cool!

Note...
Integer? (Given integer capacities.)
Yes. There is an integer vertex solution!
Constraint matrix has every subdeterminant being 1, 0, $-1$.
Vertex solution to linear program must be integral!
Ford-Fulkerson.

“Simplex” method.
Ford-Fulkerson.

“Simplex” method.

Find $s$ to $t$ path with remaining capacity.
Ford-Fulkerson.

“Simplex” method.

Find \( s \) to \( t \) path with remaining capacity.

Add to flow variables along path.
Ford-Fulkerson.

“Simplex” method.

Find $s$ to $t$ path with remaining capacity.
Add to flow variables along path.
Update remaining capacity.
Ford-Fulkerson.

“Simplex” method.
Find $s$ to $t$ path with remaining capacity.
Add to flow variables along path.
Update remaining capacity.
Repeat.

```
4 X 3
1
1 X 0
2
3
4
1
1
4
```
Ford-Fulkerson.

“Simplex” method.

Find $s$ to $t$ path with remaining capacity.
Add to flow variables along path.
Update remaining capacity.

Repeat.
Ford-Fulkerson.

“Simplex” method.
Find $s$ to $t$ path with remaining capacity.
Add to flow variables along path.
Update remaining capacity.
Repeat.

![Graph](image)
Ford-Fulkerson.

“Simplex” method.

Find $s$ to $t$ path with remaining capacity.
Add to flow variables along path.
Update remaining capacity.
Repeat.
Residual Capacity.

Find $s$ to $t$ path with remaining capacity. Add to flow along path. Update remaining capacity.

Repeat.
Residual Capacity.

Find $s$ to $t$ path with remaining capacity.
Add to flow along path.
Update remaining capacity.

Repeat.
Residual Capacity.

Find $s$ to $t$ path with remaining capacity.
Add to flow along path.
Update remaining capacity.

Repeat.

![Graph with nodes labeled s, a, b, t and edges connecting them with capacities of 1. The edge from s to a, a to t, and b to t are highlighted in red.](image-url)
Residual Capacity.

Find $s$ to $t$ path with remaining capacity.
Add to flow along path.
Update remaining capacity.

Repeat.

Uh oh! Optimal is 2! (At most 2 due to cut.)
Add reverse arcs to indicate "reverse" capacity.
Residual Capacity.

Find $s$ to $t$ path with remaining capacity.
Add to flow along path.
Update remaining capacity.

Repeat.

No remaining path.
Residual Capacity.

Find $s$ to $t$ path with remaining capacity.
Add to flow along path.
Update remaining capacity.

Repeat.

No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)
Residual Capacity.

Find \( s \) to \( t \) path with remaining capacity.
Add to flow along path.
Update remaining capacity.

Repeat.

No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)
Residual Capacity.

Find $s$ to $t$ path with remaining capacity. Add to flow along path. Or reduce flow on reverse edge. Update remaining capacity.

Repeat.

No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)
Residual Capacity.

Find $s$ to $t$ path with remaining capacity.
Add to flow along path. Or reduce flow on reverse edge.
Update remaining capacity.
Reduce $r_e = c_e - f_e$

Repeat.

No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)
Residual Capacity.

Find $s$ to $t$ path with remaining capacity. Add to flow along path. Or reduce flow on reverse edge. Update remaining capacity.

Reduce $r_e = c_e - f_e$

and add reverse $r_{uv} = f_{vu}$

Repeat.

No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.) Add reverse arcs to indicate “reverse” capacity.
Residual Capacity.

Find $s$ to $t$ path with remaining capacity. 
Add to flow along path. **Or reduce flow on reverse edge.** 
Update remaining capacity. 
  Reduce $r_e = c_e - f_e$ 
  *and add reverse* $r_{uv} = f_{vu}$ 
Repeat.

No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.) 
Add reverse arcs to indicate “reverse” capacity.
Residual Capacity.

Find $s$ to $t$ path with remaining capacity. Add to flow along path. Or reduce flow on reverse edge. Update remaining capacity.

Reduce $r_e = c_e - f_e$

and add reverse $r_{uv} = f_{vu}$

Repeat.

No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.) Add reverse arcs to indicate “reverse” capacity.
Bigger Example.

Find $s$ to $t$ path with remaining capacity.

Add to flow along path.

Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$. 
Bigger Example.

Find $s$ to $t$ path with remaining capacity.

Add to flow along path.

Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.
Bigger Example.

Find $s$ to $t$ path with remaining capacity.
Add to flow along path.
Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$.
Repeat.
Bigger Example.

Find $s$ to $t$ path with remaining capacity.
Add to flow along path.
Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$.
Repeat.
Bigger Example.

Find $s$ to $t$ path with remaining capacity.

Add to flow along path.

Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.
Find $s$ to $t$ path with remaining capacity.

Add to flow along path.

Update residual capacities: $r_e = c_e - f_e$; $r_{uv} = f_{vu}$.

Repeat.
Bigger Example.

Find $s$ to $t$ path with remaining capacity.
Add to flow along path.
Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$.
Repeat.
Bigger Example.

Find $s$ to $t$ path with remaining capacity.

Add to flow along path.

Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.
Find $s$ to $t$ path with remaining capacity.
Add to flow along path.
Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$.
Repeat.
Bigger Example.

Find $s$ to $t$ path with remaining capacity.

Add to flow along path.

Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.
Bigger Example.

Find $s$ to $t$ path with remaining capacity.

Add to flow along path.

Update residual capacities: $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.
Check Result...
Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. "Capacity constraints."

2. If $u$ is not $s$ or $t$
   \[
   \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}.
   \]

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. 
Check Result...

Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$
   \[
   \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}.
   \]

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. 
Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$
   \[
   \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}. \quad 3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.
   \]

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. 
Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$
   
   $$\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}.$$  
   $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1$.

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$.  
   $f_{sa} + f_{sc} = 1 + 3 = 4$
Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$

   $$\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}. \quad 3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.$$ 

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}. \quad f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal?
Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$
   \[ \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}. \]
   $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1$.

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal?
Find Flow: $f_e$

1. $0 \leq f_e \leq c_e$. “Capacity constraints.” $3 = f_{s,c} \leq c_{s,c} = 4$.

2. If $u$ is not $s$ or $t$
   \[
   \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}. \quad 3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.
   \]

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal? $c_{ad} + c_{cd} + c_{et} = 1 + 1 + 2 = 4$. 
Find Flow: \( f_e \)

1. \( 0 \leq f_e \leq c_e \). “Capacity constraints.” \( 3 = f_{s,c} \leq c_{s,c} = 4 \).

2. If \( u \) is not \( s \) or \( t \)
   \[
   \sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}.
   \]
   \( 3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1 \).

Maximize: size(\( f \)) = \( \sum_{(s,u) \in E} f_{su} \).
   \( f_{sa} + f_{sc} = 1 + 3 = 4 \)

Optimal? \( c_{ad} + c_{cd} + c_{et} = 1 + 1 + 2 = 4 \).

Any \( s - t \) cut gives an upper bound.
Correctness.

1. Capacity Constraints: 0 \leq f \leq c.
   - Only increase flow to c.
   - Or use reverse arcs decrease to 0.
   - Flow values to be between 0 and c.

2. Conservation Constraints:
   - "flow into v = flow out of v" (if not s or t).
   - Algorithm adds flow, say f, to path from s to t.
   - Each internal node has f in, and f out.
Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$. 

Only increase flow to $c_e$. Or use reverse arcs decrease to 0. Flow values to be between 0 and $c_e$. 

Algorithm adds flow, say $f$, to path from $s$ to $t$. Each internal node has $f$ in, and $f$ out.
Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
   Only increase flow to $c_e$. 
Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$. 
   Only increase flow to $c_e$. 
   Or use reverse arcs decrease to 0.
Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
   Only increase flow to $c_e$.
   Or use reverse arcs decrease to 0.
   Flow values to be between 0 and $c_e$. 
Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$. Only increase flow to $c_e$. Or use reverse arcs decrease to 0. Flow values to be between 0 and $c_e$.

2. Conservation Constraints:
Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$. Only increase flow to $c_e$. Or use reverse arcs decrease to 0. Flow values to be between 0 and $c_e$.

2. Conservation Constraints: "flow into $v$" = "flow out of $v$" (if not $s$ or $t$.)
Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
   Only increase flow to $c_e$.
   Or use reverse arcs decrease to 0.
   Flow values to be between 0 and $c_e$.

2. Conservation Constraints:
   “flow into $v$” = “flow out of $v$” (if not $s$ or $t$.)
   Algorithm adds flow, say $f$, to path from $s$ to $t$. 
Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
Only increase flow to $c_e$.
Or use reverse arcs decrease to 0.
Flow values to be between 0 and $c_e$.

2. Conservation Constraints:
“flow into $v$” = “flow out of $v$” (if not $s$ or $t$.)
Algorithm adds flow, say $f$, to path from $s$ to $t$.
Each internal node has $f$ in, and $f$ out.
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$. 

For valid flow:

- Flow out of $(S)$ = Flow out of $s$.
- Flow into $(T)$ = Flow into $t$.

For any valid flow, $f: E \rightarrow \mathbb{Z}^+$, the flow out of $S$ (into $T$) is

$$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S, T).$$

The value of any valid flow is at most $C(S, T)$!
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s$–$t$ cut is an upper bound on the flow.
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s$ – $t$ cut is an upper bound on the flow.

$C(S, T)$ - sum of capacities of all arcs from $S$ to $T$
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S, T)$ - sum of capacities of all arcs from $S$ to $T$

$C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e$
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s$–$t$ cut is an upper bound on the flow.

$C(S, T)$ - sum of capacities of all arcs from $S$ to $T$

$C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e$

For valid flow:
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s$–$t$ cut is an upper bound on the flow.

$C(S, T)$ - sum of capacities of all arcs from $S$ to $T$

$C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e$

For valid flow:
Flow out of $(S) = \text{Flow out of } s$. 
Optimality: upper bound.

**s-t Cut:** $V = S \cup T$ and $s \in S$ and $t \in T$.

**Lemma:** Capacity of any $s-t$ cut is an upper bound on the flow.

$C(S, T)$ - sum of capacities of all arcs from $S$ to $T$

$C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e$

For valid flow:
Flow out of $(S) = $ Flow out of $s$.
Flow into $(T) = $ Flow into $t$. 
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s$ – $t$ cut is an upper bound on the flow.

\[ C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e \]

For valid flow:
Flow out of $(S) = \text{Flow out of } s$.
Flow into $(T) = \text{Flow into } t$. 
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s$–$t$ cut is an upper bound on the flow.

$C(S, T)$ - sum of capacities of all arcs from $S$ to $T$

$$C(S, T) = \sum_{e = (u, v): u \in S, v \in T} c_e$$

For valid flow:

Flow out of $(S) = $ Flow out of $s$.

Flow into $(T) = $ Flow into $t$.

For any valid flow, $f : E \to \mathbb{Z}_+$, the flow out of $S$ (into $T$)
Optimality: upper bound.

$s-t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s-t$ cut is an upper bound on the flow.

$C(S, T)$ - sum of capacities of all arcs from $S$ to $T$

$C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e$

For valid flow:
Flow out of $(S) = $ Flow out of $s$.
Flow into $(T) = $ Flow into $t$.

For any valid flow, $f : E \rightarrow Z_+$, the flow out of $S$ (into $T$)
Optimality: upper bound.

**s-t Cut:** \( V = S \cup T \) and \( s \in S \) and \( t \in T \).

**Lemma:** Capacity of any \( s-t \) cut is an upper bound on the flow.

- \( C(S, T) \) - sum of capacities of all arcs from \( S \) to \( T \)
  \[
  C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e
  \]

For valid flow:
- Flow out of \((S)\) = Flow out of \( s \).
- Flow into \((T)\) = Flow into \( t \).

For any valid flow, \( f : E \to \mathbb{Z}_+ \), the flow out of \( S \) (into \( T \))
  \[
  \sum_{e \in S \times T} f_e
  \]
Optimality: upper bound.

\( s-t \) Cut: \( V = S \cup T \) and \( s \in S \) and \( t \in T \).

Lemma: Capacity of any \( s-t \) cut is an upper bound on the flow.

\[ C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e \]

For valid flow:
Flow out of \( (S) \) = Flow out of \( s \).
Flow into \( (T) \) = Flow into \( t \).

For any valid flow, \( f : E \rightarrow \mathbb{Z}_+ \), the flow out of \( S \) (into \( T \))
\[ \sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \]
Optimality: upper bound.

$s-t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s-t$ cut is an upper bound on the flow.

$C(S, T)$ - sum of capacities of all arcs from $S$ to $T$

$C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e$

For valid flow:
Flow out of $(S) = \text{Flow out of } s$.
Flow into $(T) = \text{Flow into } t$.

For any valid flow, $f : E \rightarrow Z_+$, the flow out of $S$ (into $T$)

$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0$
Optimality: upper bound.

\[ s-t \text{ Cut: } V = S \cup T \text{ and } s \in S \text{ and } t \in T. \]

Lemma: Capacity of any \( s - t \) cut is an upper bound on the flow.

\[ C(S, T) - \text{sum of capacities of all arcs from } S \text{ to } T \]

\[ C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e \]

For valid flow:

Flow out of \( (S) \) = Flow out of \( s \).
Flow into \( (T) \) = Flow into \( t \).

For any valid flow, \( f : E \rightarrow \mathbb{Z}_+ \), the flow out of \( S \) (into \( T \))

\[ \sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S, T). \]
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s$–$t$ cut is an upper bound on the flow.

$C(S, T) - \text{sum of capacities of all arcs from } S \text{ to } T$

$C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e$

For valid flow:
Flow out of $(S)$ = Flow out of $s$.
Flow into $(T)$ = Flow into $t$.

For any valid flow, $f : E \to \mathbb{Z}^+$, the flow out of $S$ (into $T$)

$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S, T)$.

$\rightarrow$ The value of any valid flow is at most $C(S, T)$!
Optimality: upper bound.

$s$-$t$ Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

Lemma: Capacity of any $s$ – $t$ cut is an upper bound on the flow.

$C(S, T)$ - sum of capacities of all arcs from $S$ to $T$

$C(S, T) = \sum_{e=(u,v): u \in S, v \in T} c_e$

For valid flow:
Flow out of $(S)$ = Flow out of $s$.
Flow into $(T)$ = Flow into $t$.

For any valid flow, $f : E \to Z^+$, the flow out of $S$ (into $T$)

$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S, T)$.

$\Rightarrow$ The value of any valid flow is at most $C(S, T)!$
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Value of flow = min cut.

Valid flow = all that flow from source.

Optimal is ≤ min cut.

→ Flow is maximum!! Cut is minimum s–t cut too!
Optimality: \( \text{max flow} = \text{min cut} \).

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at \( s \) does not reach \( t \).
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.
Optimality: max flow $= \min$ cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.
No arc with positive residual capacity leaving $S$

Total flow leaving $S$ is $C(S, T)$.

Valid flow $\Rightarrow$ all that flow from source.
Value of flow equals value of $C(S, T)$.

Optimal is $\leq C(S, T)$.

$\rightarrow$ Flow is maximum!!
Cut is minimum $s-t$ cut too!

"any flow" $\leq$ "any cut" and this flow $=$ this cut.

$\rightarrow$ Maximum flow and minimum $s-t$ cut!
Optimality: \( \text{max flow} = \text{min cut} \).

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at \( s \) does not reach \( t \).

\[ f_e = c_e \]

\( S \) be reachable nodes.
No arc with positive residual capacity leaving \( S \)

\[ \implies \text{All arcs leaving } S \text{ are full.} \]
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
   No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$s$ be reachable nodes.

No arc with positive residual capacity leaving $S$

$\implies$ All arcs leaving $S$ are full.
$\implies$ No arcs into $S$ have flow.
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.

No arc with positive residual capacity leaving $S$

$\Rightarrow$ All arcs leaving $S$ are full.
$\Rightarrow$ No arcs into $S$ have flow.

Total flow leaving $S$ is $C(S, T)$. 
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at \( s \) does not reach \( t \).

\[ f_e = c_e \]

\( S \) be reachable nodes.
No arc with positive residual capacity leaving \( S \)

\[ f_e = 0 \]

\[ \implies \text{All arcs leaving } S \text{ are full.} \]
\[ \implies \text{No arcs into } S \text{ have flow.} \]

Total flow leaving \( S \) is \( C(S, T) \).

Valid flow \( \implies \) all that flow from source.
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.
No arc with positive residual capacity leaving $S$

$\Rightarrow$ All arcs leaving $S$ are full.
$\Rightarrow$ No arcs into $S$ have flow.
Total flow leaving $S$ is $C(S, T)$.

Valid flow $\Rightarrow$ all that flow from source.
Value of flow equals value of $C(S, T)$. 
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$s \rightarrow t$

$f_e = c_e$

$f_e = 0$

$S$ be reachable nodes.
No arc with positive residual capacity leaving $S$

$\Rightarrow$ All arcs leaving $S$ are full.
$\Rightarrow$ No arcs into $S$ have flow.

Total flow leaving $S$ is $C(S, T)$.

Valid flow $\Rightarrow$ all that flow from source.
Value of flow equals value of $C(S, T)$. and
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.

No arc with positive residual capacity leaving $S$

$\implies$ All arcs leaving $S$ are full.
$\implies$ No arcs into $S$ have flow.

Total flow leaving $S$ is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$. 
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.

No arc with positive residual capacity leaving $S$

$\implies$ All arcs leaving $S$ are full.
$\implies$ No arcs into $S$ have flow.

Total flow leaving $S$ is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$. 

→
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.
No arc with positive residual capacity leaving $S$

$\implies$ All arcs leaving $S$ are full.
$\implies$ No arcs into $S$ have flow.

Total flow leaving $S$ is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.

$\rightarrow$ Flow is maximum!!
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.
No arc with positive residual capacity leaving $S$
$\implies$ All arcs leaving $S$ are full.
$\implies$ No arcs into $S$ have flow.
Total flow leaving $S$ is $C(S, T)$.

Valid flow $\implies$ all that flow from source.
Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!
Cut is minimum $s – t$ cut too!
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.
No arc with positive residual capacity leaving $S$

$\implies$ All arcs leaving $S$ are full.
$\implies$ No arcs into $S$ have flow.

Total flow leaving $S$ is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.

$\rightarrow$ Flow is maximum!!

Cut is minimum $s - t$ cut too!
“any flow” $\leq$ “any cut”
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at \( s \) does not reach \( t \).

\[ S \text{ be reachable nodes.} \]

No arc with positive residual capacity leaving \( S \)

\[ \implies \text{All arcs leaving } S \text{ are full.} \]
\[ \implies \text{No arcs into } S \text{ have flow.} \]

Total flow leaving \( S \) is \( C(S, T) \).

Valid flow \( \implies \) all that flow from source.

Value of flow equals value of \( C(S, T) \). and Optimal is \( \leq C(S, T) \).

\[ \rightarrow \text{Flow is maximum!!} \]

Cut is minimum \( s - t \) cut too!
“any flow” \( \leq \) “any cut” and
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
   No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.
   No arc with positive residual capacity leaving $S$
   $\implies$ All arcs leaving $S$ are full.
   $\implies$ No arcs into $S$ have flow.

Total flow leaving $S$ is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
→ Flow is maximum!!

Cut is minimum $s - t$ cut too!
“any flow” $\leq$ “any cut” and this flow = this cut.
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$s \rightarrow t$

$f_e = c_e$

$f_e = 0$

$S$ be reachable nodes.
No arc with positive residual capacity leaving $S$

$\Rightarrow$ All arcs leaving $S$ are full.
$\Rightarrow$ No arcs into $S$ have flow.
Total flow leaving $S$ is $C(S, T)$.

Valid flow $\Rightarrow$ all that flow from source.
Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s – t$ cut too!
“any flow” $\leq$ “any cut” and this flow $=$ this cut.
$\rightarrow$
Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at $s$ does not reach $t$.

$S$ be reachable nodes.
No arc with positive residual capacity leaving $S$

$\implies$ All arcs leaving $S$ are full.
$\implies$ No arcs into $S$ have flow.

Total flow leaving $S$ is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s - t$ cut too!
“any flow” $\leq$ “any cut” and this flow = this cut.
$\rightarrow$ Maximum flow and minimum $s - t$ cut!
Theorem: In any flow network, the maximum $s$-$t$ flow is equal to the minimum cut.
Celebrated max flow -minimum cut theorem.

Theorem: In any flow network, the maximum $s$-$t$ flow is equal to the minimum cut.

Celebrate!
Simplex Implementation:
Start at a (feasible) vertex.
Lecture in a Minute

Simplex Implementation:
Start at a (feasible) vertex.
(defined by linear system $A'x = [b', 0, \cdots, 0]$).
Begin at origin. Move to better neighboring vertex.
Coordinate system: distance from tight constraints.
Vertex at origin in coordinate system.
$O(mn)$ time to update linear system.
Until no better neighboring vertex.
Objective function in coordinate system is non-positive.
Dual Variables: new objective function!
Lecture in a Minute

Simplex Implementation:
Start at a (feasible) vertex.
(defined by linear system $A'x = [b', 0, \cdots, 0]$).
Begin at origin. Move to better neighboring vertex.
Coordinate system: distance from tight constraints.
Vertex at origin in coordinate system.
$O(mn)$ time to update linear system.
Until no better neighboring vertex.
Objective function in coordinate system is non-positive.
Dual Variables: new objective function!

Maximum flow.
“Greedy” augment path... 
Except reverse old decisions ..
Reverse residual capacities.
(Friday): Optimality?
No augmenting path $\implies$
$s – t$ cut size = flow value.
Find flow and $s – t$ cut with equal value!