LINEAR PROGRAMS

General Linear Program

Canonical Form

n variables $x_1, \ldots, x_n$, n+m constraints

$$\max \sum_{i=1}^{n} c_i x_i$$
subject to $x_i \geq 0$ for $i = 1, \ldots, n$

$\sum_{j=1}^{m} a_{ij} x_j \leq b_i$ for $i = 1, \ldots, m$

Ax $\leq$ b

Let $c^T x$ be the objective function.

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$
$$c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$$

$s.t. x \geq 0$

Remark: If we don't have the constraints $x_i \geq 0$, we can transform to the canonical form by considering the 2n variables $x_i^+$, setting $x_i = x_i^+ - x_i^-$ and imposing the constraints $x_i^+ \geq 0$

Simplex Algorithm

Feasible Region: Polytop of $x \in \mathbb{R}^n$ s.th. x satisfies all constraints

Vertex: Point x in the feasible region such that n of the constraints are tight, i.e., satisfied as equalities.

Neighbor of a vertex x: Vertex $x'$ which differs in one the defining constraint

Value: $c^T x$
**Simplex Algorithm**

- Start at vertex \( x^* \)
- Find neighbor \( y^* \) with maximal value
- If value \( (y^*) > \text{value}(x^*) \), move to \( y^* \).
- Repeat

**Running time**

- \( O(n^3 \cdot m) \) per step
- Worst case exponentially many steps

**Maximum Flow**

**Input:**
1) Directed graph \( G = (V, E) \)
2) "Source" vertex \( s \in V \)
3) "Sink" vertex \( t \in V \)
4) For each edge \( e \in E \) capacity \( c_e \in N \)

**Goal:** Route maximum amount of water from \( s \) to \( t \)

---

Flow 1

Flow 2
**Def:** A flow $f$ is an assignment of a number $f_e$ to each directed edge $e \in E$ such that:

- **(Nonnegativity)** $f_e \geq 0$ for all edges
- **(Capacity)** $f_e \leq c_e$
- **(Flow-in = Flow-out)** For each vertex $v \neq s, t$

$$\text{Flow-in } \sum_{uv \in E} f_{uv} = \text{Flow-out} \sum_{vw \in E} f_{vw}$$

**Maximum Flow Problem**

Maximize $\text{size}(f) := \sum_{v : s \not\in E} f_{sv}$ (flow from $s$ to $t$)

s.t. $f$ is a flow

**Algorithm (first, greedy)**

1) Find a path $P$ from $s$ to $t$ which is not yet saturated

2) Send more flow along $P$

3) Repeat

**Flow $n \rightarrow A \rightarrow t$: 1 unit**

**Flow $n \rightarrow A \rightarrow B \rightarrow t$: 1 unit**

**Wrong Answer**

**Total:** 2 units
Need to undo flow $\nu \rightarrow A \rightarrow B \rightarrow \epsilon$.

**Algorithm to undo flow**

```
\[ S \rightarrow A \rightarrow B \rightarrow L : \text{one unit} \quad \n \rightarrow B \rightarrow A \rightarrow t \quad \text{2 units total} \]
```

**Def:** Given a graph $G$ and a flow $f$ on $G$ the residual graph $G_f$ is constructed as follows:

For each edge $e = uv$

```
flow $f_e$
\[ u \xrightarrow{\text{capacity } c_e} v \quad \Rightarrow \quad u \xrightarrow{\text{capacity } c_e - f_e} v \]
```

**(original graph)**  

**(residual graph)**

**Ford Fulkerson Algorithm**

1) Find path $P$ from $s$ to $t$ which is not yet saturated in the residual graph
2) Send more flow along $P$
3) Repeat
Example:

\[S \rightarrow A \rightarrow B \rightarrow \epsilon \quad \text{Flow 3}\]
\[S \rightarrow A \rightarrow \epsilon \quad \ni 2\]
\[S \rightarrow B \rightarrow A \rightarrow \epsilon \quad \text{Flow 3}\]

Total 8

Proof this is opt.

Look at cut

flow \(s \overset{\epsilon}{\longrightarrow} t = 8\)

Rem: A path \(P\) which is not yet saturated in \(G\)

(i.e., has > 0 capacity for all edges in \(P\)) is called an augmenting path

Def: An \(s-t\) cut is a partition \(V = L \cup R\) of the vertex set such that \(s \in L\) and \(t \in R\)

Def: The capacity of the cut is \(\text{capacity}(L,R) = \sum_{u \in L, v \in R} \text{Cap}_{uv}\)

Thm: For any flow \(f\) on \(G\)

any cut \((L,R)\)

size \(|f| \leq \text{capacity}(L,R)\)

Cor: Max Flow \(\leq\) Min Cut

capacity of \(\overset{\epsilon}{\longrightarrow} = 8\)
Thm: Max Flow = Min Cut

Need to show ≥

Idea: Run Ford-Fulkerson on G. Let f be the flow it outputs. Prove \[ \text{size}(f) \geq \min \text{Cut} \]

⇒ \text{capacity}(L,R) ≥ \text{size}(f) ≥ \min \text{Cut} \quad \forall \text{cuts } (L,R)

⇒ \min \text{Cut} = \max \text{Flow} = \text{Output of Ford-Fulkerson}

Proof:

• When Ford-F. terminates, there exists no s-t path in residual graph with >0 capacity for all edges.

• \( L \) = set of vertices reachable from \( s \) in \( G_f \)
  \( R \) everything else \( \Rightarrow t \in R \) (since FF terminated)

• We will prove that \[ \text{size}(f) = \text{capacity}(L,R) \]

• First, note that for all edges uv crossing the cut from \( L \) to \( R \) (\( u \in L, v \in R \)), the capacity in \( G_f \) must be zero (otherwise, \( v \) would be reachable from \( s \), and hence in \( L \)).
• This means that the flow \( f \) has the following property:

**Property:** Assume the edge \( e = uv \) is a directed edge crossing the cut

a) If \( u \in L, v \in R \) (LR-edge) \( f_{uv} = c_{uv} \)

b) If \( u \in R, v \in L \) (RL-edge) \( f_{uv} = 0 \)

**Proof:** In \( G_f \), \( uv \) has capacity \( c_{uv} - f_{uv} \) which is zero, see above.

b) Assume \( f_{uv} > 0 \). Then the capacity of the residual edge \( vu \) would be \( f_{uv} > 0 \) in \( G_f \). But \( vu \) is a LR-edge, and must have residual capacity 0.

• This allows us to complete the proof:

\[
\text{size} N = \sum_{u \in L, v \in R} f_{uv} - \sum_{u \in R, v \in L} f_{uv} = \sum_{u \in L} c_{uv} = \text{capacity}(L, R)
\]
Q: Why was it important that $f_{u,v} = 0$ for right to left edges.

A: In general, a path could go back and forth, so not all edges from $L$ to $R$ would contribute to the flow from $s$ to $t$. If $f_{u,v} = 0$ for all right-to-left edges, this can't happen.

Rem: If the capacities are integer, in each step Ford-Fulkerson updates the capacities by integer amounts, increasing the size by an integer amount $\Delta \geq 1$. So in particular, F-F uses at most $U = \text{max-flow}$ many path.

Runtime: $\frac{\# \text{ of augmenting paths} \times \text{time to find paths}}{\text{depth} \text{ First search}} \leq U = \text{max-flow}$. 

0(n+m)