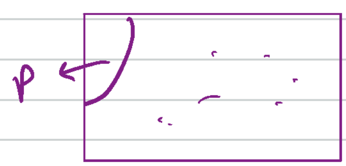
Lecture 22

CS 170

Sanjam Garg.

NP - complete problems still need a solution.



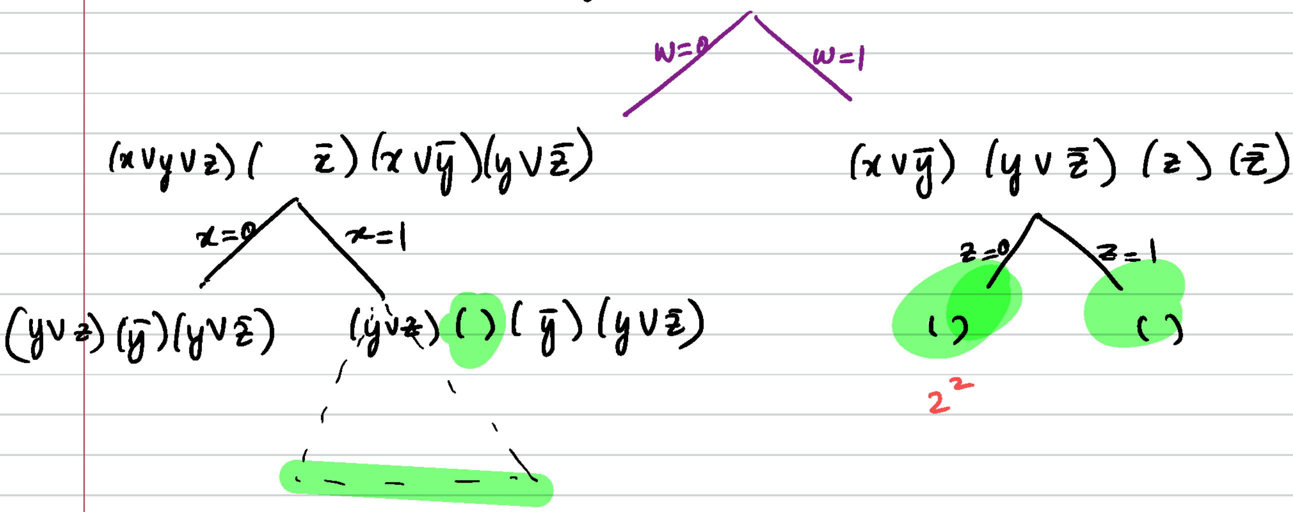
1) "Intelligent" exponential search.
   - → Running time could be exponential
   - → Practical instances
     - ↳ run efficiently.

2) Approximation Algorithms. ⟶ polytime
   - ↳ relationship with the optimal solution.

3) Heuristic → no guarantees on the run time or the optimality of solution.

Backtracking

$$\phi = (w \vee x \vee y \vee z)(w \vee \bar{x})(x \vee \bar{y})(y \vee \bar{z})(z \vee \bar{w})(\bar{w} \vee \bar{z})$$



① which subproblem to consider?

- ↳ expand
- ↳ choose

② each subproblem has a test

- ↳ success
- ↳ fail
- ↳ maybe.

Start with some problem P_0

Let $S = \{P_0\}$, the set of active subproblems

Repeat while S is not empty:

choose subproblem $P \in S$, and remove it from S
expand into smaller subproblems $P_1 \dots P_k$
For each P_i
if test(P_i) succeeds : halt and announce this solution
test(P_i) fails : discard P_i
Otherwise : add P_i to S

Announce that there is no solution

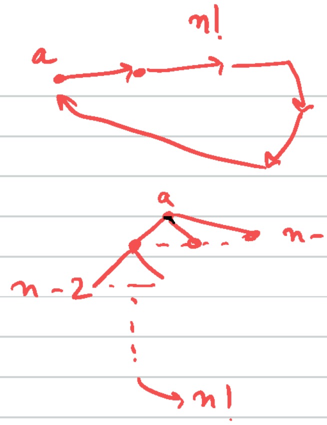
Branch and Bound: Generalizing backtracking to search problems

Start with some problem P_0

Let $S = \{P_0\}$, the set of active subproblems
bestsofar $= \infty$
Repeat while S is not empty:
choose subproblem (partial solution) $P \in S$, and remove it from S
expand into smaller subproblems $P_1 \dots P_k$
For each P_i
if P_i is a complete solution: update bestsofar
else if lowerbound(P_i) < bestsofar : add P_i to S

Return bestsofar

Travelling Salesman - branch and bound.



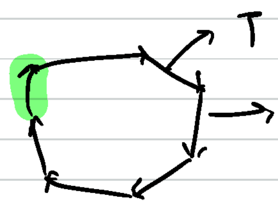
Instance: distances d_{ij}

Solution: a permutation $\tau : \{1 \ldots n\} \to \{1..n\}$ s.t.

$W_{TSP} = d_{\tau(1)\tau(2)} + d_{\tau(2)\tau(3)} \; \ldots \; d_{\tau(n)\tau(1)}$ is minimized.

Lemma: $W_{TSP} \geq W_{MST}$

Proof:



ⓐ T is a spanning tree

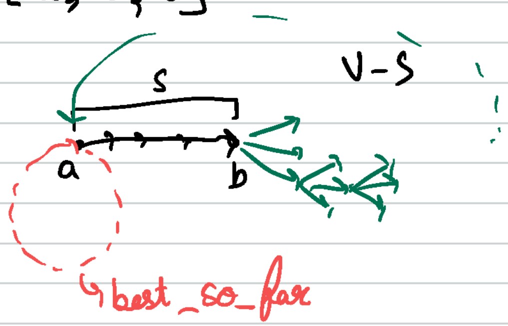
ⓑ $W_{TSP} \geq W_T$

ⓒ $W_T \geq W_{MST}$

$\hookrightarrow W_{TSP} \geq W_{MST}$

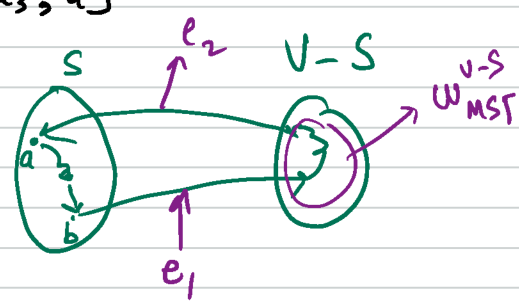
w.l.o.g → start with a

$[a, S, b]$



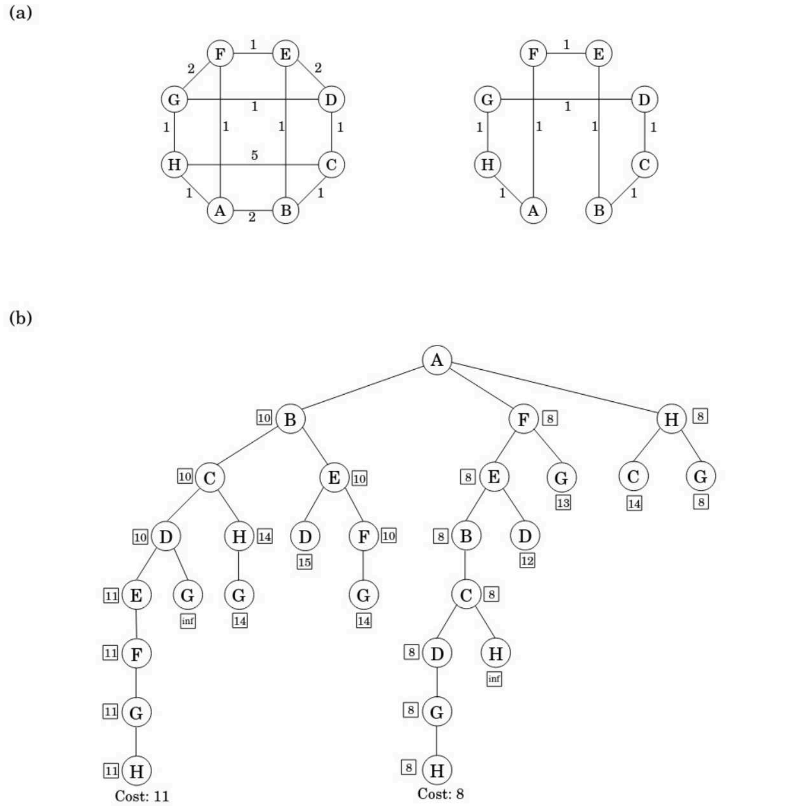
Starting point.

$[a, \{a\}, a]$



$\geq W_{e_1} + W_{e_2} + W^{V-S}_{MST}$

Figure 9.2 (a) A graph and its optimal traveling salesman tour. (b) The branch-and-bound search tree, explored left to right. Boxed numbers indicate lower bounds on cost.



Approximation Algorithms for an optimization problem

Instance I $\longrightarrow$ $OPT(I)$
(minimization problem e.g. TSP) $\xrightarrow{A}$ $A(I)$

$\alpha = \max_{I} \frac{A(I)}{OPT(I)}$

approx ratio

$\alpha = \max_{I} \frac{OPT(I)}{A(I)}$

for maximization

$\alpha > 1$

Set Cover

Input: A set of elements B; sets $S_1, S_2 \dots S_m \subseteq B$

Output: Smallest selection of S_i whose union is B

$B = \{1 \dots 6\}$

Example: $S_1 = \{1, 2, 3\}$ $S_2 = \{2, 3, 4, 5\}$ $S_3 = \{4, 5, 6\}$

Greedy Algorithm

Repeat until all elements of B are covered Pick the set S_i with the largest number of uncovered elements

S_2, S_1, S_3

Claim: Say $|B| = n$ and $OPT(I) = k$. Then greedy algorithm uses atmost $k \ln n$ sets.

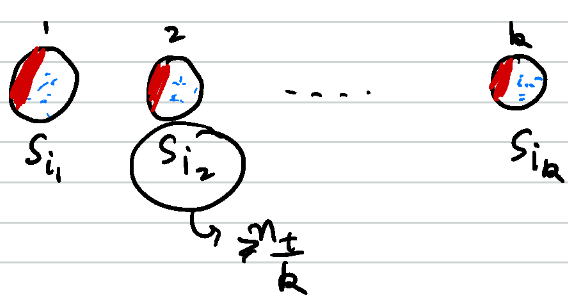
Proof:

$0 \rightarrow 1 \rightarrow 2 \quad \dots \quad t \rightarrow t+1$

$n_0 \quad n_1 \quad n_2 \quad \dots \quad n_t$ → # of uncovered elements in B after t iterations of our greedy algo.

$n_0 = n$

after t^{th} iteration



Claim: atleast one of these sets has $\geq \frac{n_t}{k}$ uncovered elements.

Proof: $< \frac{n_t}{k} \times k = n_t$

$$n_{t+1} \leq n_t - \frac{n_t}{k} \leq n_t\left(1 - \frac{1}{k}\right)$$

$1 - \frac{1}{k} < e^{-\frac{1}{k}}$

$$n_t \le n\left(1 - \frac{1}{k}\right)^t < \underbrace{n e^{-\frac{t}{k}}}_{1}$$

$$\underbrace{t = k \ln n}$$

$n_t < 1$

Vertex Cover

Input: undirected graph $G = (V, E)$

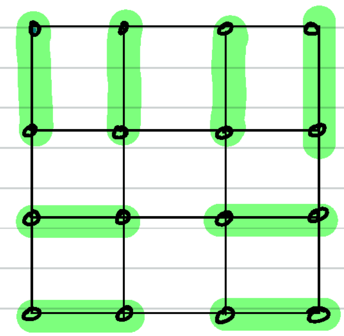
Output: subset $S \subseteq V$ such that $|S|$ is minimized and S touches every edge

$\rightarrow$ $B = \{e_1 \ldots e_m\}$

$S_u = \{e \mid$ one of the vertices in e is u & $e \in E\}$

$\searrow \ln n$

- Find a maximal matching $M \subseteq E$
- Return $S = \{\text{all endpoints of edges in } M\}$



(i) 2 Size of any VC $\geq 2|M|$

↳ you have pick atleast one vertex per edge.

$2(\text{Size of OPT VC}) \geq |S|$

(ii) $|S| = 2|M| \leq 2$ Size of any VC

(iii) S is a VC.

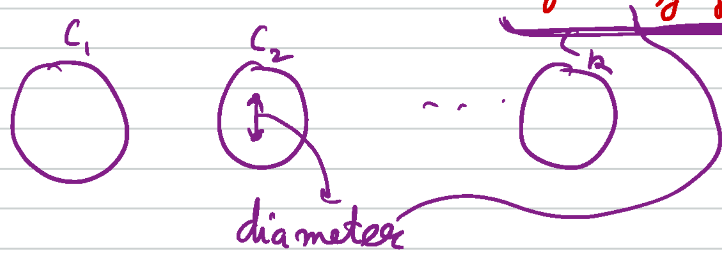
↳ if S is not a VC then $\exists$ an edge $e \stackrel{=(u,v)}{}$ s.t neither $u \in S$ nor $v \in S$

↳ $\underline{M \cup \{(u,v)\}} \rightarrow$ larger matching than M.

Clustering

Input: Points $X = \{x_1 \dots x_n\}$ with distance metric $d(\cdot,\cdot)$; integer k

Output: k clusters $C_1 \dots C_k$ s.t. $\underbrace{\max_j \max_{x,y \in C_j} d(x,y)}$ is minimized



① $\forall x,y \; d(x,y) \geq 0$

② $d(x,y) = 0$ iff $x = y$

③ $d(x,y) = d(y,x)$

④ TI $d(x,y) + d(y,z) \geq d(x,z)$