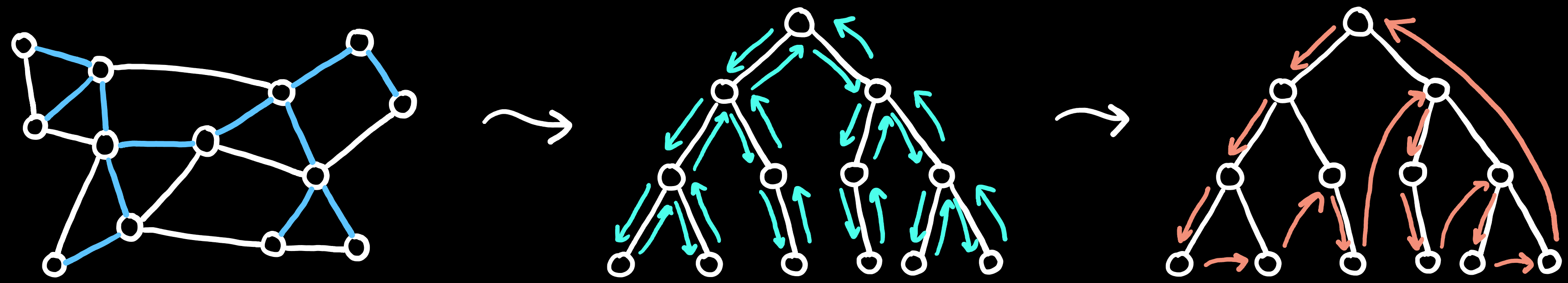
Lecture 19: Approximation Algorithms



Goals for today

- Understand the **motivation** and **definition** of approximation algorithms
- Demonstrate three common techniques for approximation algorithms:
  - **Greedy** (Job Scheduling)
  - **LP rounding** (Vertex Cover)
  - **Reduce to similar problems** (Traveling Salesperson)

Approximation algorithms: what & why

Some problems are NP-hard

- Give up?
- "Pretty good" solution?

Pretty good means provably close to optimal

Formal definition

Definition (c-approximation algorithm):

Algorithms give solution within factor of c of optimal solution

ALG = value of our algorithm's solution
OPT = value of optimal solution

Minimize

$ALG \leq c \cdot OPT$
$(c > 1)$

Maximize

$ALG \geq c \cdot OPT$
$(c < 1)$

Technique #1

Greedy algorithms

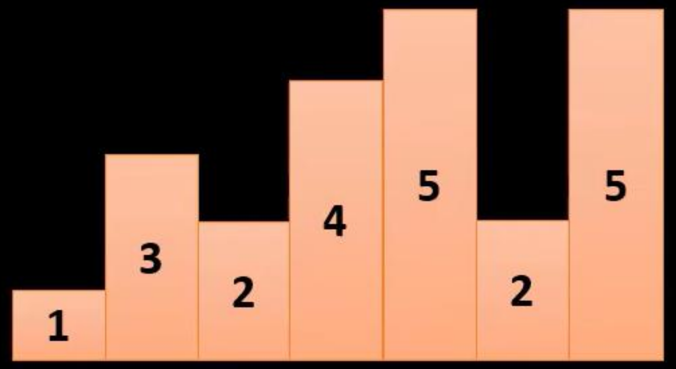
Job Scheduling

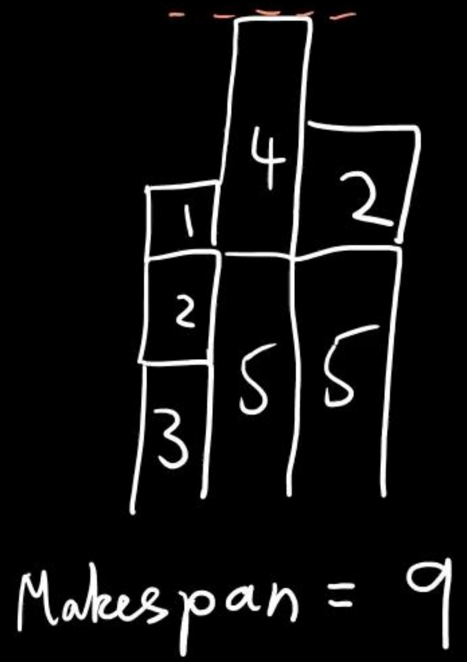
Problem: Given m identical "machines" and n "jobs", where job i takes p_i processing time to run, assign jobs to machines to minimize the ***makespan***, the time at which the last job finishes

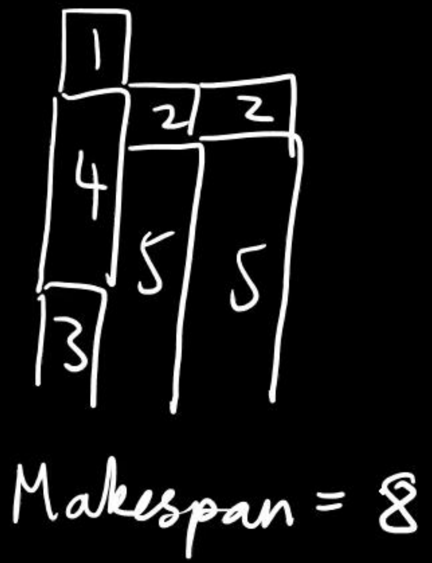
Alternative interpretation: Given n blocks where block i has height p_i, we want to make m stacks of blocks, with the goal of minimizing the height of the tallest stack

Job Scheduling

Example: $p = \{1, 3, 2, 4, 5, 2, 5\}, m = 3$







Approximation algorithm for job scheduling

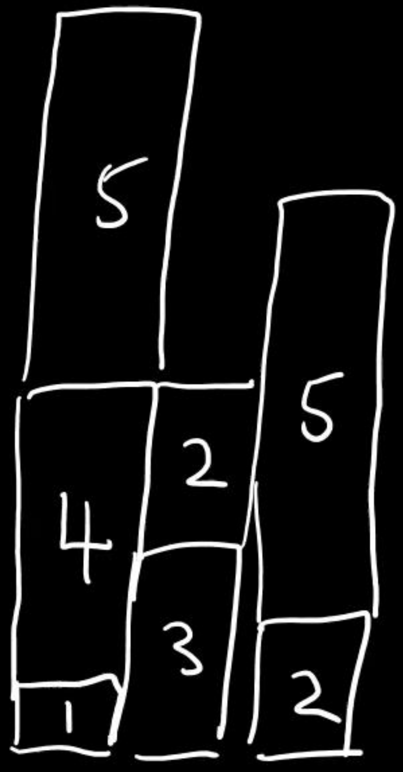
Algorithm (Greedy)

Start with m empty stacks

for each block i

put it in shortest current stack

$\cancel{1}, \cancel{3}, \cancel{2}, \cancel{4}, \cancel{5}, \cancel{2}, \cancel{5}$



ALG = 10

Analysis of greedy job scheduling

Claim: Greedy job scheduling is a **2**-approximation algorithm

Proof: p_{i^*} is height of final block on tall stack

L is rest of height of tallest stack

How to relate to OPT?

$OPT \geq p_{i^*}$ $\quad$ $OPT \geq L$

$\Rightarrow ALG = p_{i^*} + L \leq OPT + OPT$

$= 2\,OPT$

$ALG = p_{i^*} + L$

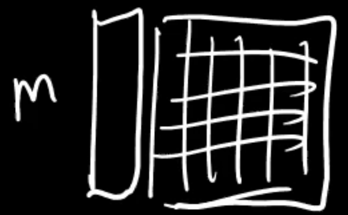
Can we do better than 2?

Question: What is a worst-case input for greedy scheduling?

Idea: Lots of small, end with big

Example: $m(m-1)$ blocks of 1, one size m

$ALG = 2m-1$



$OPT = m$

Better algorithm for job scheduling

Algorithm (sorted greedy)

Sorting big to small

Greedy

Analysis of sorted greedy job scheduling

Claim: Sorted greedy job scheduling is a **1.5**-approximation algorithm

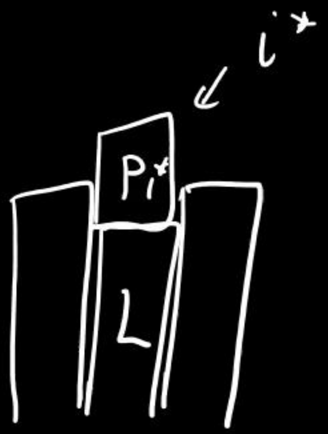
Proof: $OPT \geq p_{i^*}$ $\quad OPT \geq L$

Blocks under i^* are $\geq p_{i^*}$

$\Rightarrow$ $m+1$ blocks are $\geq p_{i^*}$

$\Rightarrow$ Pigeonhole: Some stack has ≥ 2

$\Rightarrow$ $OPT \geq 2p_{i^*}$



$$ALG = p_{i^*} + L \leq \frac{1}{2}OPT + OPT = 1.5\,OPT$$

Analysis of sorted greedy job scheduling

Claim: Sorted greedy job scheduling is a **1.5**-approximation algorithm

Proof:

Summary of Greedy

Take-home messages:

- Greedy algorithms are often good approximations
- Hardest part is the proof
  - Need to find a way to connect OPT to ALG
  - Often achieved by lower bounding OPT and relating this to ALG

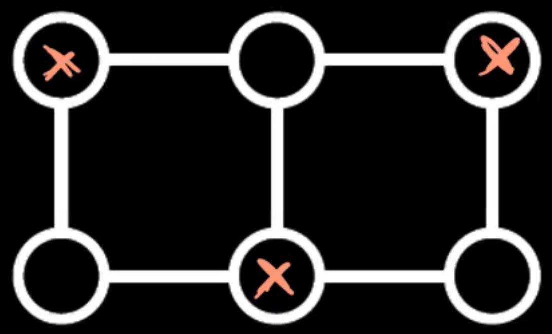
Technique #2

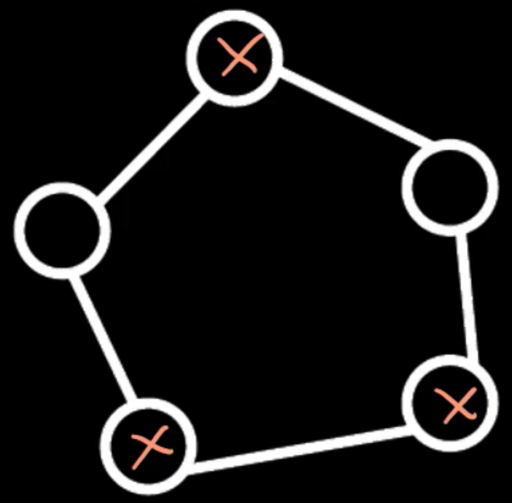
LP Rounding

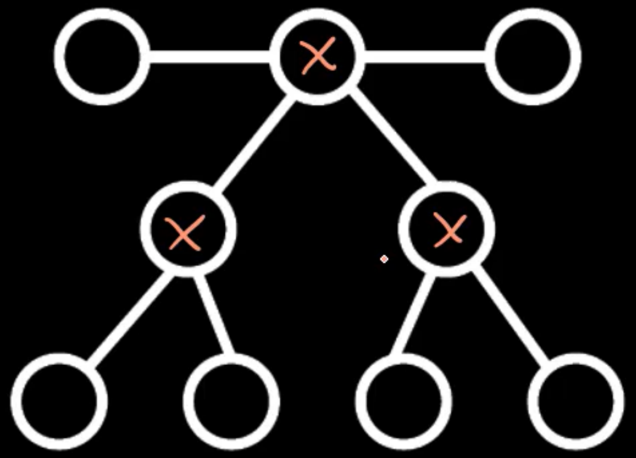
Problem: Vertex Cover

Given an undirected graph $G = (V, E)$, a ***vertex cover*** is a subset of the vertices $C \subseteq V$ such that every edge is adjacent to at least one $v \in C$.

A ***minimum vertex cover*** is a smallest possible vertex cover







Linear program for vertex cover?

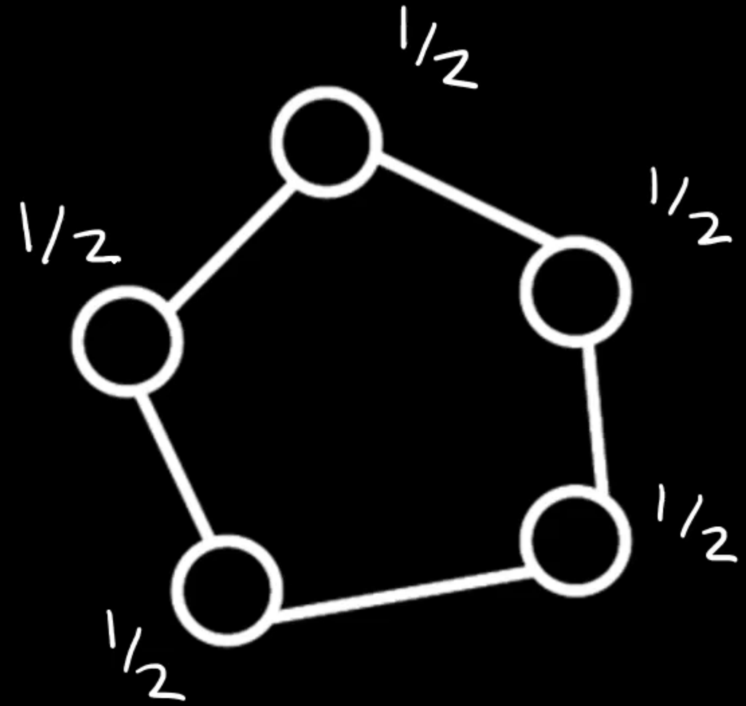
Vertex variables x_v

minimize $\sum x_v$

s.t $x_u + x_v \geq 1$

$x_v \geq 0$

Can get fractional!



Approximation algorithm for vertex cover

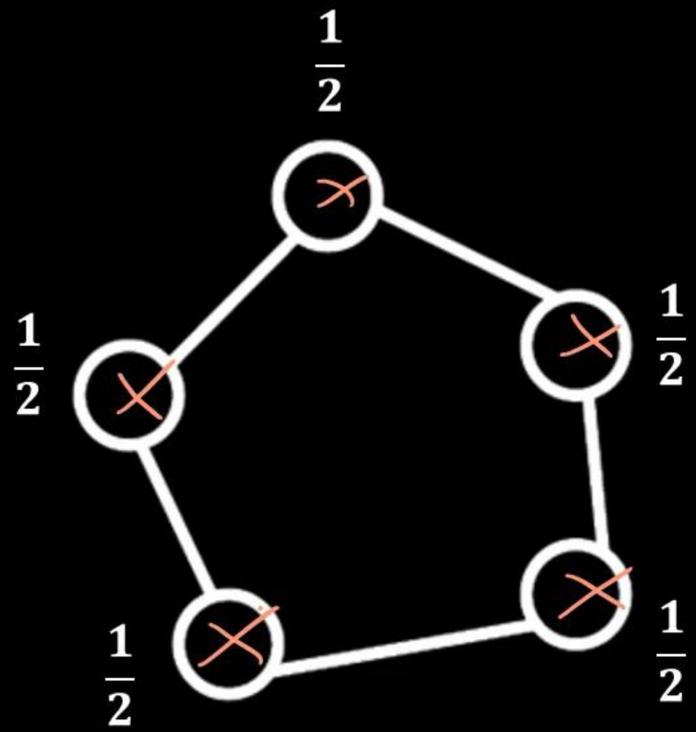
Algorithm (Round-and-relax)

Solve LP relaxation for x_v

for each vertex v

  if $x_v \geq 1/2$

    take v



Analysis of LP rounding for vertex cover

Claim 1: The LP rounding algorithm outputs a valid vertex cover

Proof: AFSOC (u,v) not covered

→ Did not pick u or v

$x_u < 1/2 \qquad x_v < 1/2$

$x_u + x_v < 1$

Contradiction

Analysis of LP rounding for vertex cover

***Claim 2*:** The LP rounding algorithm is a **2**-approximation algorithm

Proof: Variables at most double

Objective $\sum x_v$ doubles

$ALG \leq 2 \cdot LP \leq 2 \cdot OPT$

Check your understanding

Question: Can we apply this algorithm to *any* LP relaxation and get a good approximation? Why or why not

Summary of LP Rounding

Take-home messages:

- Linear program relaxations are powerful tools for approximation
- Need to prove **feasibility** and bound the **change in objective**
- Rounding $\geq 1/2$ up and $< 1/2$ down doesn't always work
  - Might need more sophisticated rounding rules (see homework)

Technique #3

Reduce to similar problems

Problem: Metric Traveling Salesperson

Given a *complete* **undirected** graph $G = (V, E)$ and a ***distance metric:*** $d(u, v)$ between *every vertex* u, v, find the length of the shortest tour that visits every vertex exactly once and returns to the start

Definition (Metric):

1. $d(u,u) = 0$
2. $d(u,v) = d(v,u)$
3. $d(u,v) \le d(u,k) + d(k,v) \quad \forall u, v, k$

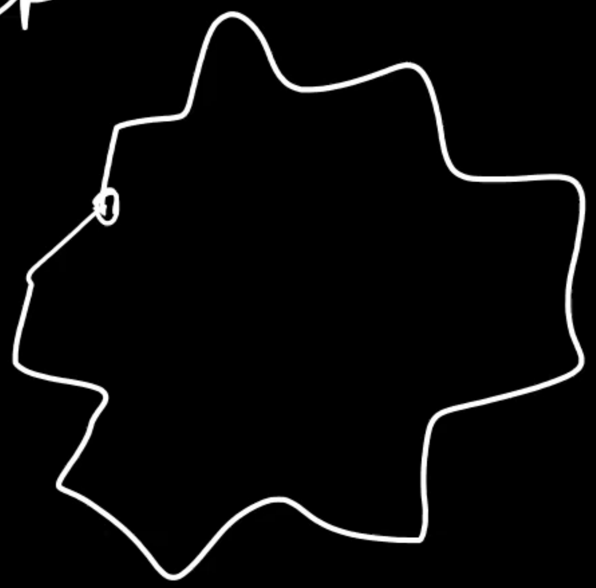
What other graph problems do we know?

Minimum Spanning Trees

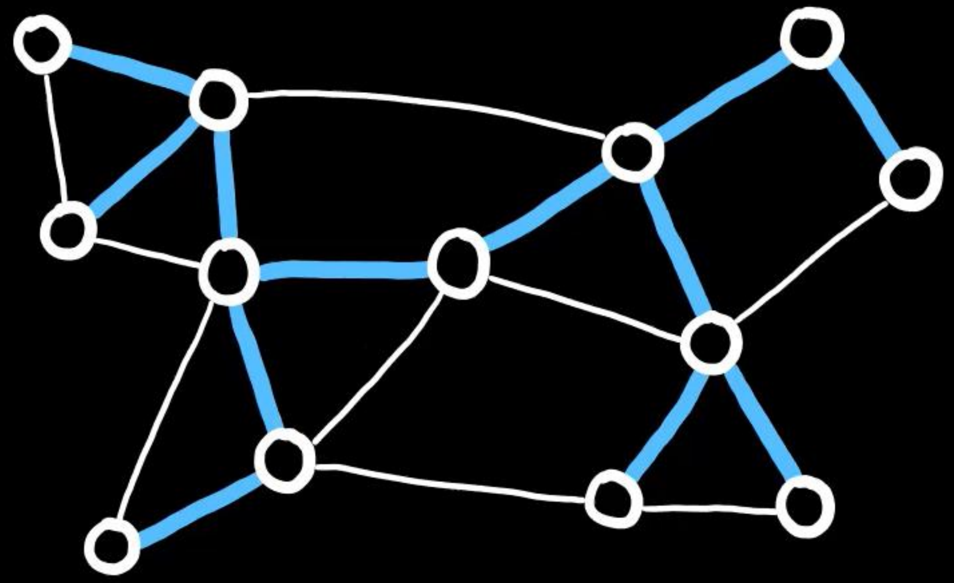
Network Flow / Minimum Cut

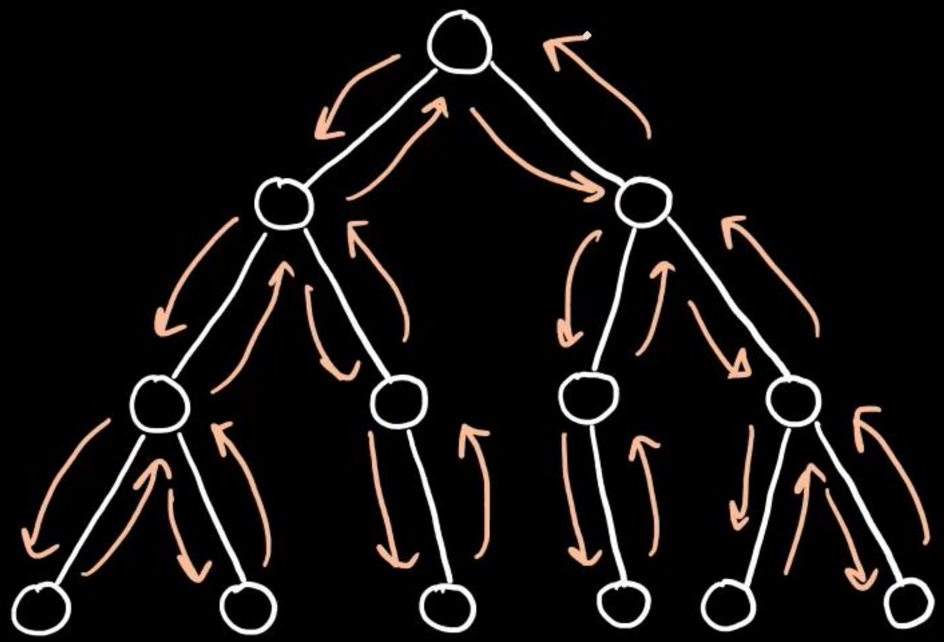
Shortest paths

Turn MST into a cycle?

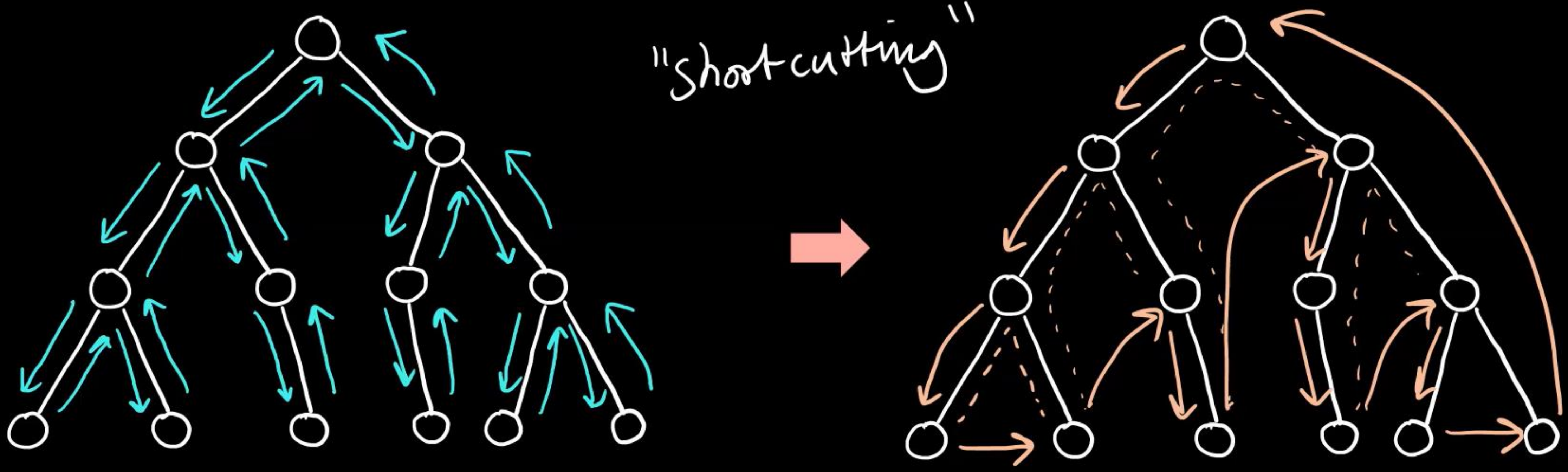


Approximating Metric TSP using an MST





Approximating Metric TSP using an MST

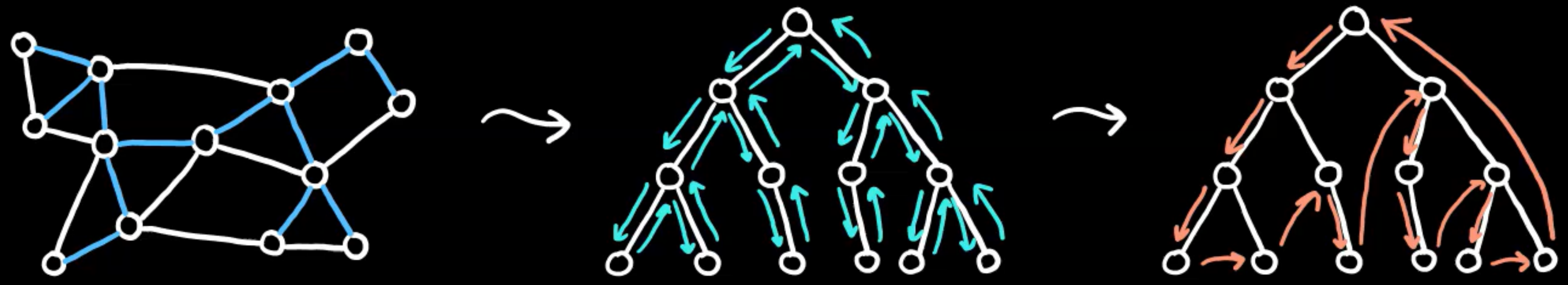


Pre-order traversal!

Approximation algorithm for Metric TSP

Algorithm

Output a pre-order of the MST

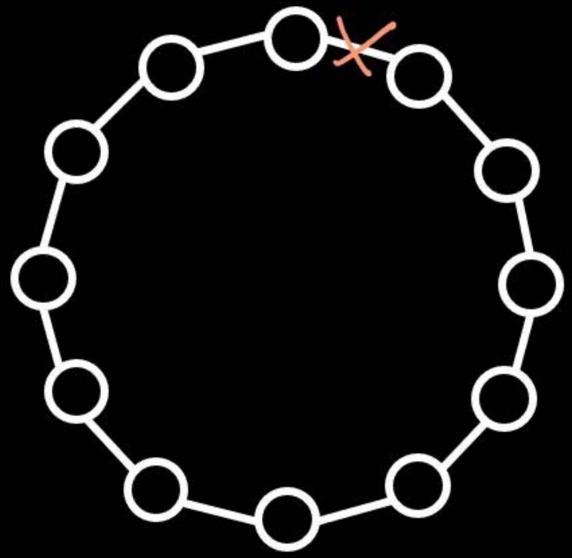


Analysis of MST-TSP approximation

Claim: The MST-TSP algorithm is a **2**-approximation algorithm

Lemma 1: weight(MST) $\leq$ weight(TSP)

Proof:



$$\text{weight}(MST) \leq \text{weight}(TSP - \{e\}) \leq \text{weight}(TSP).$$

Analysis of MST-TSP approximation

Claim: The MST-TSP algorithm is a **2**-approximation algorithm

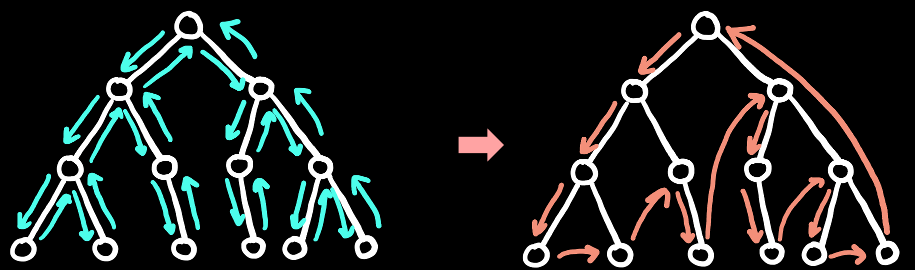
Lemma 1: weight$(MST) \leq$ weight(TSP)

Lemma 2: $ALG \leq 2$ weight(MST)

Proof: The original traversal uses every edge twice, cost = 2 MST

Adding shortcuts can not increase distance (triangle inequality)

Therefore ALG ≤ 2 weight (MST)



Analysis of MST-TSP approximation

Claim: The MST-TSP algorithm is a **2**-approximation algorithm

Lemma 1: weight$(MST) \leq$ weight(TSP)

Lemma 2: $ALG \leq\ 2$ weight(MST)

Proof of claim:

$$ALG \leq 2\,\text{weight}(MST) \leq 2\,\text{weight}(TSP) = 2\,OPT$$

Summary

- We **defined** the concept of approximation algorithms
- We practiced **three techniques** for building approximation algorithms:
  - **Greedy** (Job Scheduling)
  - **LP rounding** (Vertex Cover)
  - **Reduce to similar problems** (TSP $\rightarrow$ MST)