Friday, March 3, 2023
Lecture #20
MSSC 6000

Announcements

* HW 3 due Wed, March 8, 11:59pm
* Midterm Exam, Wed, March 8 in class
  (up to backtracking, no branch + bound)
* Friday, March 10, office hours 10am-10:50am
  in my office, no lecture

Topic 8 - Branch and Bound

Ex: Job Assignment Problem
You have n tasks that need to be done and n workers. Each task has a different cost to complete depending on which worker does it.

Each worker can do 1 task. Goal: minimize total cost.

	tasks 1	2	3	4
workers A	3	5	2	2
B	6	8	10	8
C	2	6	4	9
D	10	4	7	5

cost = 21

Many applications:
→ Drivers picking up passengers
→ Shipments from mines to factories

* Search Space: All assignments of workers to tasks.

How big? n! $(4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24)$

Constraints? None, every candidate is valid.
Backtracking is useless (equivalent to brute force)

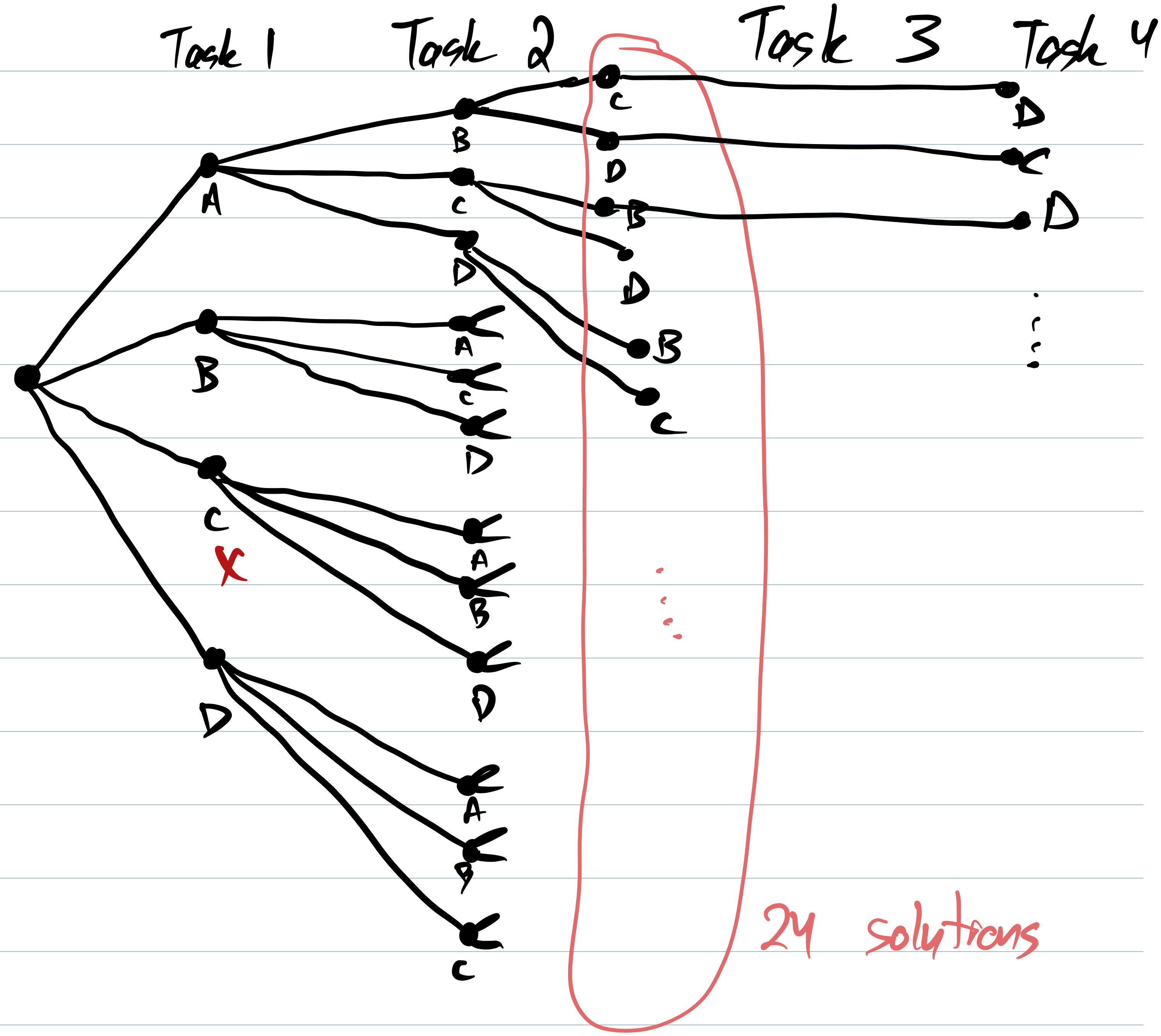
Two things to describe

(1) Branching (2) Bounding

↓

how we're going to build the partial solutions

* Pick which worker does a certain task



Bounding: in this problem we are **minimizing** so what we want is a **lower bound** for the best way to complete any partial solution.

"I don't know how cheaply I can finish this partial solution, but I know for sure I can't do it cheaper than X."

↑ lower bound

Suppose we've already decided that worker B will do task 1.

	1	2	3	4
A	3	5	2	2
B	6	8	10	8
C	2	6	4	9
D	10	4	7	5

Under this assumption how can we find a lower bound for the best way to complete this partial solution?

If every other task is free, the cost already incurred (6) is a lower bound. → True, but not a strong bound

Better: each worker will have to do a task, they could never do better than everyone doing the cheapest remaining task

	1	2	3	4
→ A	3	5	2	2
B	6	8	10	8
→ C	2	6	4	9
→ D	10	4	7	5

This is a lower bound of $6+2+4+4=16$.

Alternative: Every remaining task has to be done. They can never be done cheaper than their cheapest cost.

	1	↓ 2	↓ 3	↓ 4
A	3	5	2	2
B	6	8	10	8
C	2	6	4	9
D	10	4	7	5

This is a lower bound of $6+4+2+2=14$.

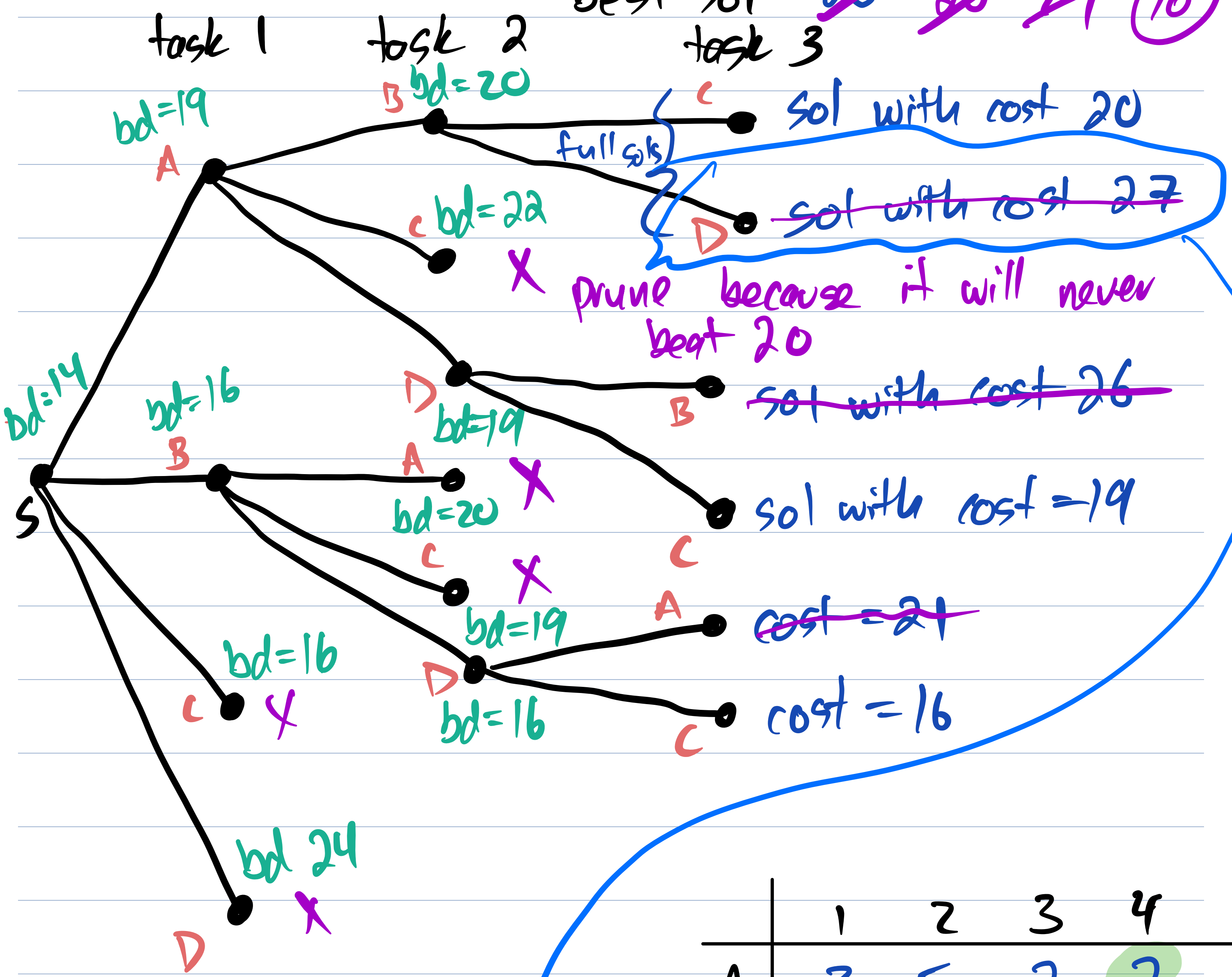
For <u>this</u> partial solution the first version was stronger, but in general you can try both and always use the stronger (higher) one.

Lower Bound:

max(sum of smallest # in each remaining row,
sum of smallest # in each remaining col)
+ cost of already-decided tasks

Fully worked example.

best sol: ~~∞~~ ~~20~~ ~~19~~ 16



	1	2	3	4
A	3	5	2	2
B	6	8	10	8
C	2	6	4	9
D	10	4	7	5

best sol of 16

special trick: once we find that the sol ABC w/ cost 20, since the parent has a bound of 20, there's no need to explore the other children

Notes:

* In general, the hardest part is finding a good bound — highly problem specific. The stronger the bound, the more pruning you get => faster algorithms
* At the start we had no "best sol" so we started at ∞. Instead, we can we can run a greedy algo first to have a starting solution.

With this we would have done a lot more pruning.

	1	2	3	4
A	3	5	2	2
B	6	8	10	8
C	2	6	4	9
D	10	4	7	5

cost = 16