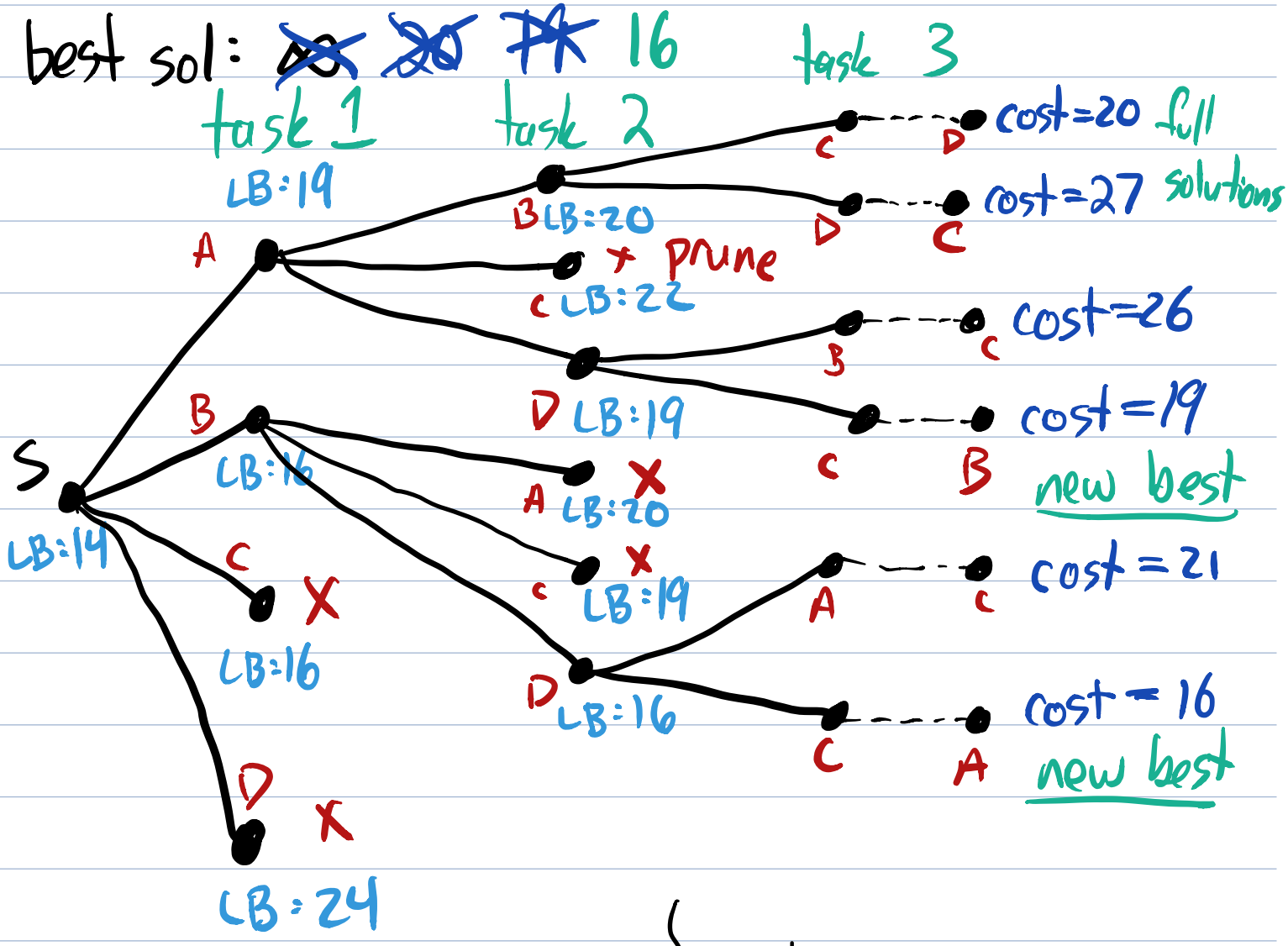
Wednesday, March 3

Job Assignment Problem



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

* Hardest part is finding a good bound
* At the start, we had no best solution. Start by finding some solution before doing B+B. (greedy sol)

General Procedure: ← search space (maximization)

```
function bb(S, best_sol=None):
    if best_sol is None:
        best_score = -∞
    else:
        best_score = score(best_sol)

    if |S|=1:
        candidate = the one thing in S
        value = score(candidate)
        if value > best_score:
            return candidate
        else:
            return best_sol

    S1, S2 = branch(S)        → assuming two at a time
    if bound(S1) > best_score:
        best_sol = bb(S1, best_sol)
        best_score = score(best_sol)
    if bound(S2) > best_score:
        best_sol = bb(S2, best_sol)
    →

    return best_sol
```

base case

Relaxation

Knapsack

Capacity: 14

item	weight	value
1	8	13
2	3	7
3	5	10
4	5	10
5	2	1
6	2	1
7	2	1

Branching:
- → item 1 is in or out
- → item 2 is in or out

...

Bounding: Suppose we have put item 1 out and item 2 in. How can we find an upper bound on the best we could possibly do with the rest?

* "Add up the value of all remaining items"

Technically an UB, but a useless one.

The trick is **relaxation:** sometimes it's easier to find an UB if you adjust the problem to be more permissible.

Fractional Knapsack: You are allowed to take fractions of items.

Any UB on Fractional Knapsack is also an UB on Regular Knapsack.