Scientific Computing

Feb 19, 2025

Announcements

→ HW 3 assigned today
due in two weeks, March 5

→ Wednesday, March 5 is also the in-person midterm exam

→ Friday, March 7, no lecture, extra office hours while you work on take-home (time TBD)

Today

→ Backtracking

Office Hours:

Mon + Fri

9:30am - 10:30am

Cudahy 307

Topic 7 – Backtracking

Like Divide + Conquer, Backtracking is a framework for finding the optimal solution in a search space without checking every candidate one-by-one.

Very simple idea: Build solutions one part at a time, and give up when a partial solution violates the constraints.

Ex #1: Knapsack

Capacity: 10

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

Search space: All subsets of $\{1,2,3,4,5,6,7\}$ $2^7 = 128$

With brute force:

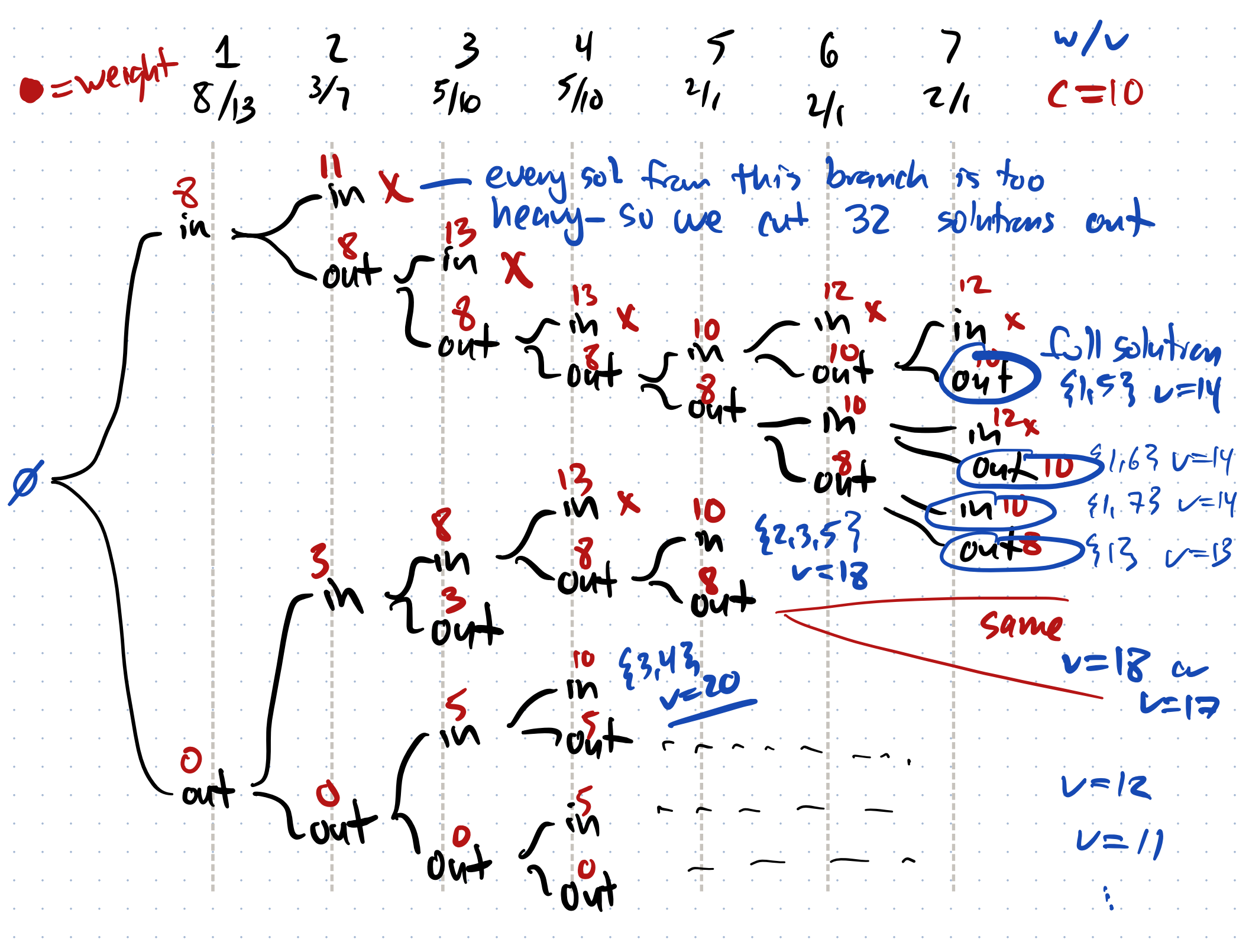
Possibilities: $\emptyset$, $\{1\}$, $\{2\}$, ...

$\{1,3,4,5,7\}$, ...

not just too heavy, but still too heavy if you remove any single item, so this is silly to even try!

128 possibilities

Waste of time to check $\{1,3,4,5,7\}$ because $\{1,3,4,5\}$ was already too heavy


●=weight
1 8/13
2 3/7
3 5/10
4 5/10
5 2/1
6 2/1
7 2/1
w/v
C=10
∅
8 in
11 in X — every sol from this branch is too heavy- so we cut 32 solutions out
8 out
13 in X
8 out
13 in X
8 out
10 in
3 out
12 in X
10 out
12 in X
10 out
full solution {1,5} v=14
10 in
3 out
12 in X
10 out
{1,6} v=14
10 in
8 out
10 in
8 out
{1,7} v=14
{1,3} v=13
3 in
13 in X
8 in
3 out
10 in
8 out
{2,3,5} v=18
8 out
same v=18 or v=17
0 out
10 in
5 out
{3,4} v=20
5 in
0 out
v=12
5 in
0 out
v=11
⋮

Messy picture, but way better than brute force, especially with lots of items!

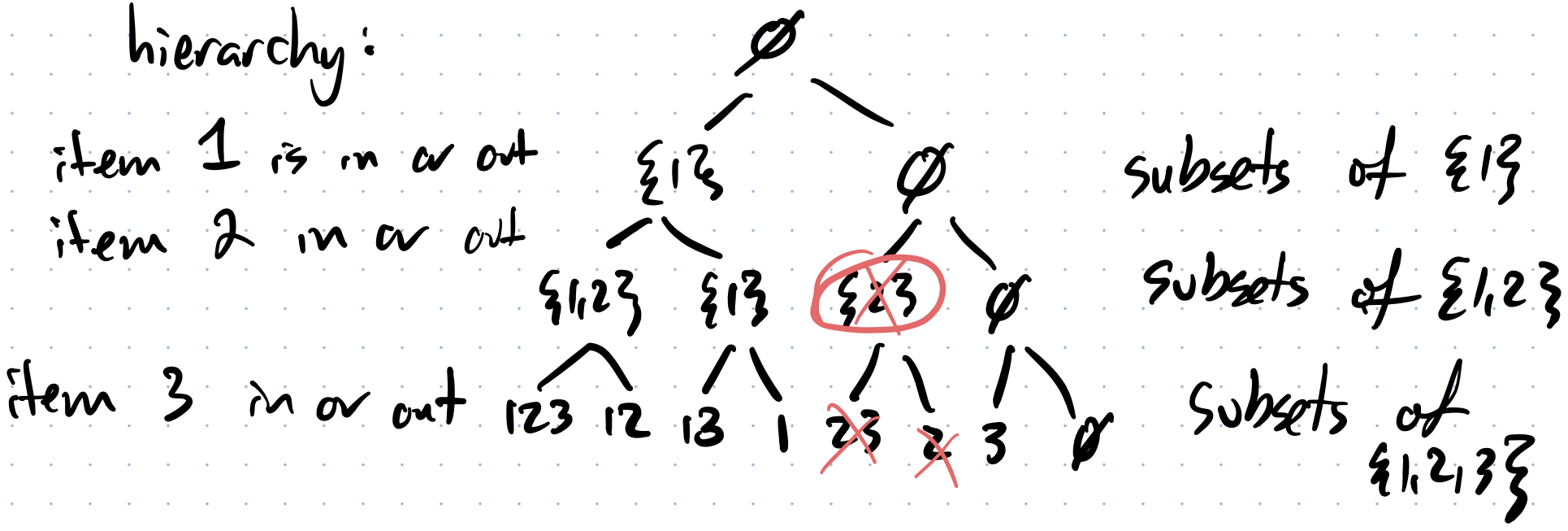
What are we doing?

- Putting a hierarchy on decisions that builds the whole space (math: poset) with the critical property that: if a partially built candidate is bad, then the candidates below it must be bad.

Knapsack with 7 items:

Candidates: subsets of $\{1,2,3,4,5,6,7\}$

hierarchy:



Traverse this tree, and whenever you reach a candidate that is bad, stop traversing that branch.

So, we are checking or ruling out every candidate in the search space. In bad cases (high capacity, light items), we might not rule anything out, and so in the worst case this is as bad as brute force.

[demo]