Friday, Feb 24, 2023
Lecture # 17
MSSC 6000

covers D+C, brute force, backtracking
Q2

Announcements

* HW 3 assigned today, due Wed, March 8, 11:59pm
* HW 2 due tonight, 11:59pm
* Extra Office Hours today, 1pm-2pm, in-person
* Office Hours next Wednesday cancelled
* Midterm Exam Wed, March 8 in class

Topic #7 - Backtracking

Like D+C, backtracking is a style of algorithm for finding optimal solutions in a search space without actually checking every candidate.

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

Ex #1: Knapsack Problem

Capacity: 10

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

With brute force, you check every subset.

Possibilities (candidates):

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

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

not only is this too heavy, but it's still too heavy if you remove any single item

Checking all $2^7 = 128$ possibilities includes a lot of over-capacity candidates that we could have predicted are bad.

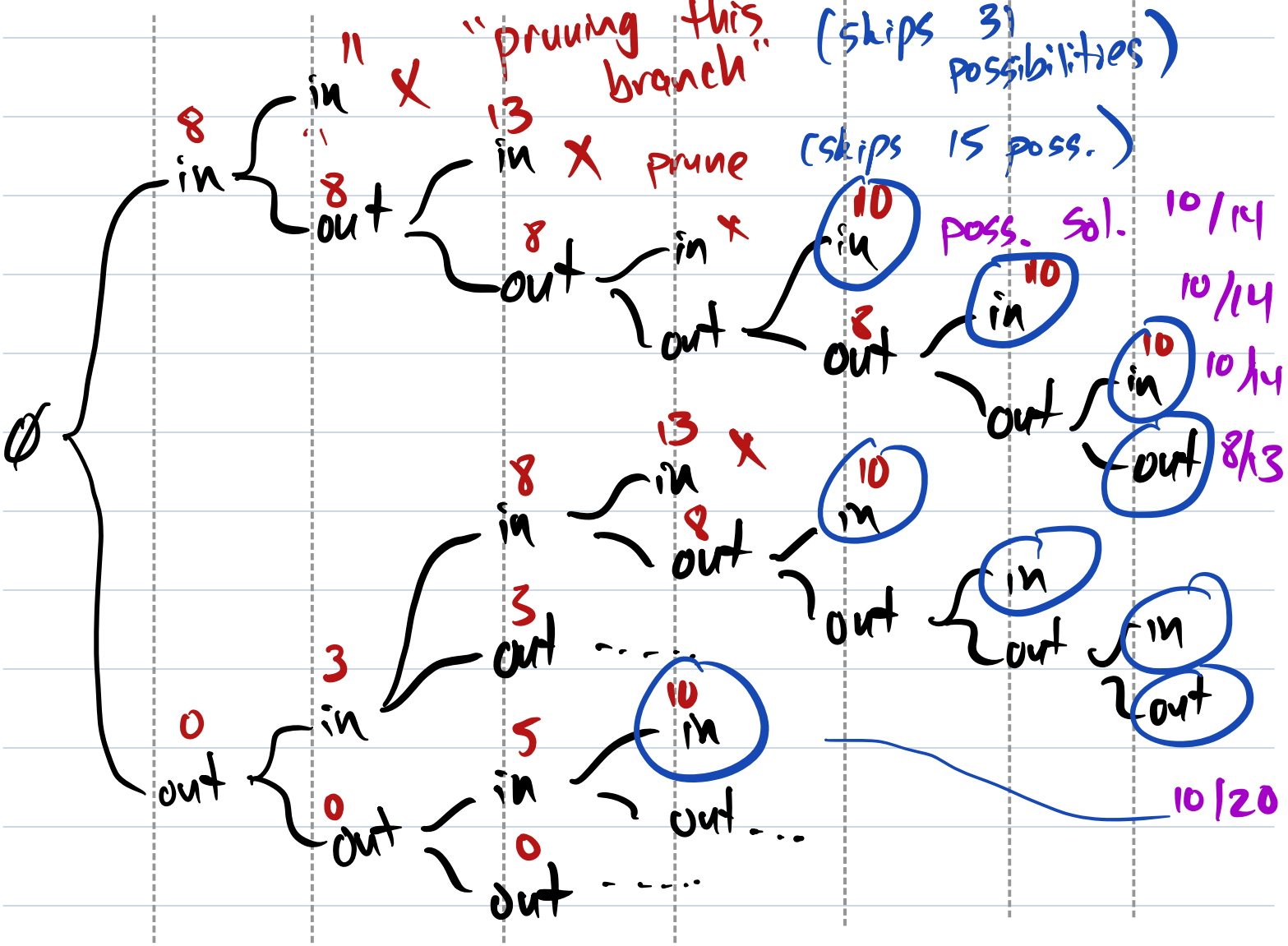
Ex: $\{1,2\}$ already over capacity

⇒ skip $\{1,2,3\}$ $\{1,2,4\}$, ..., $\{1,2,7\}$
$\{1,2,3,4\}$, $\{1,2,3,5\}$, ... $\{1,2,\ldots,7\}$

skip ~~32~~ 31 move possibilities

{1,2} + any subset of {3,4,5,6,7}

Backtracking: for each item, decide whether or not to keep it



Way better than brute force in terms of speed but trickier to implement.

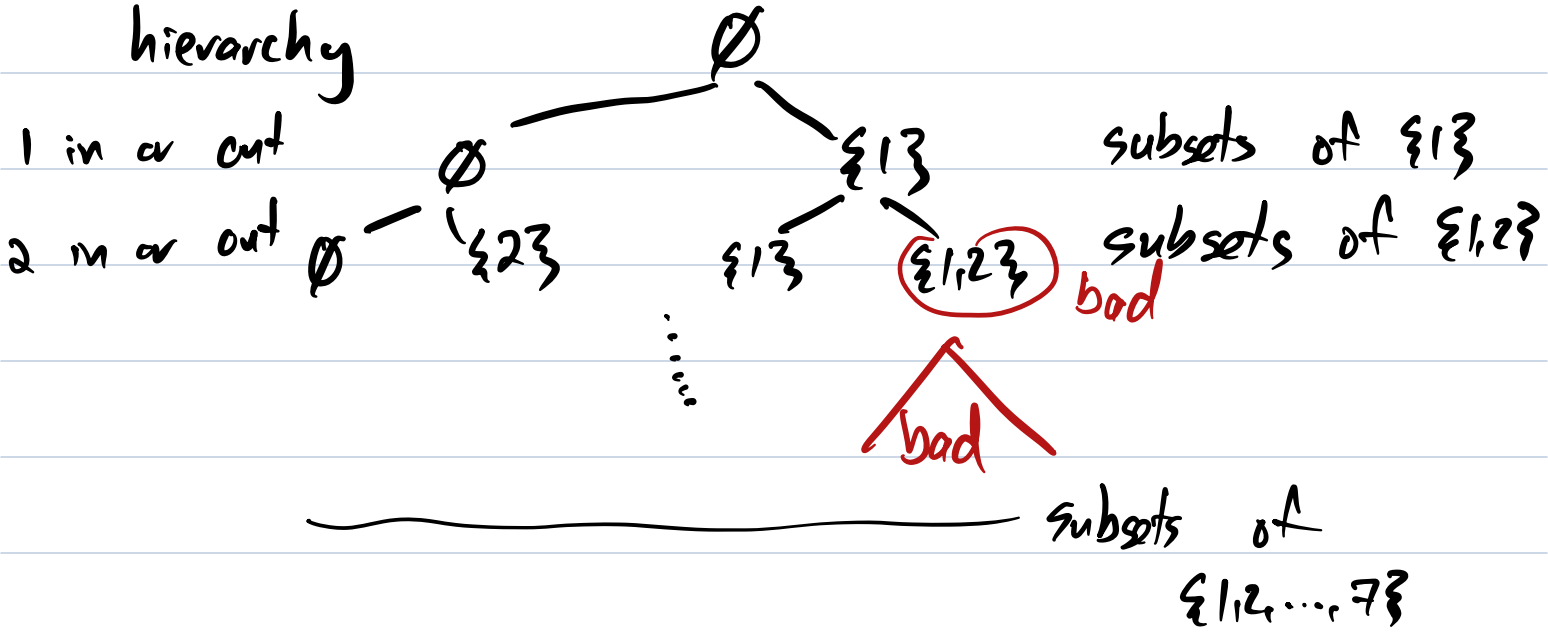
What are we doing?

- Putting a hierarchy on decisions that build the whole search space with the critical property:
  if a partial solution becomes bad, then every solution that branches off of it is also bad

Knapsack with 7 items

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

hierarchy



In code: traverse the tree, and whenever you find a bad candidate, stop traversing that branch.

You can pick the order of the decisions, and this could impact how fast the code is (deciding on heavier items first probably leads to earlier pruning).

Can be much faster than brute force. In bad cases (no pruning) this is just as bad as brute force.

Ex: if you have a super high capacity or very light items, you might have no (or very little pruning).

Ex #2: Sudoku

4	7	1	6	2	3	8	9	5
6	0	8		5	4			
		5			8	7		4
8			4	3	2			
	3			1			4	
			9	8	7			1
1		3	8			4		
			3	4		5		9
				6	9		1	8

Search space:
All ways of filling 1-9 in each blank space.

Size of search space?
$9^{\#\text{ of blanks}}$

Backtracking: Start filling blanks L-R then T-B. Start each cell at "1". If that cell doesn't violate the rules, move to the next cell. If it does violate, add 1 to it. If it reaches 10, clear the cell, go back to the previous one.

See course website for link to Sudoku demo