Friday! Feb 16, 2024
Scientific Computing

Announcements

→ Monday Office Hours canceled

* Replacement Tuesday, virtual on Teams time TBD

→ HW 2 due next Friday

Topic 5 - Search Spaces + Brute Force

Most of our problems can be summarized as:

"Out of all ways to do [something]:

(1) Do any of them satisfy certain constraints?

and/or

(2) which one is optimal?"

Greedy algos give very quick solutions

to do [something], usually decent quality, usually not completely optimal.

G.A. do <u>not</u> try all possible solutions — usually they just try a single one.

The <u>search space</u> of a problem is the set of all possible "things" that <u>may or may not</u> satisfy your constraints, <u>may or may not</u> be optimal, but they all have some <u>score</u> that you're trying to maximize or minimize.

The next few lectures:

Methods to check (or rule out) every item in the search space to guaranteed optimal solutions.

After that:

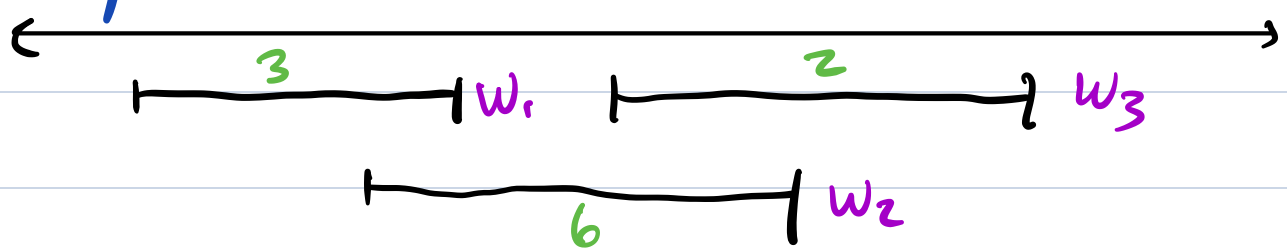
Methods to get very good (not nec. optimal) solutions by wandering around the search space in clever ways.

Most obvious way to get an optimal solution: brute force.

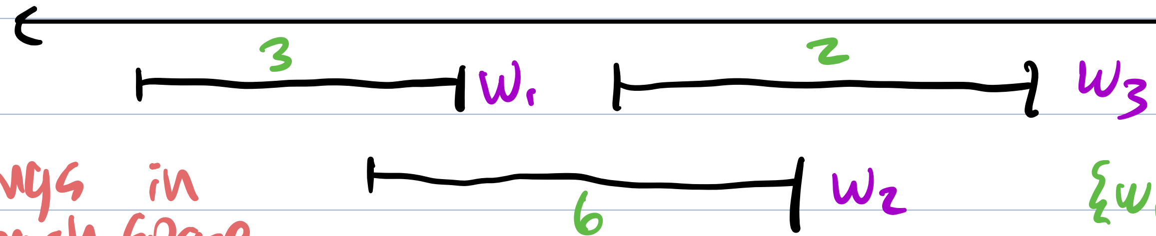
→ Generate every element of the search space.
For each one: check if valid
if so, compute the score
Keep track of the best solution that you've seen so far.

Ex 1: Weighted Interval Scheduling

3 requests



Search space: All subsets of $\{w_1, w_2, w_3\}$



things in search space

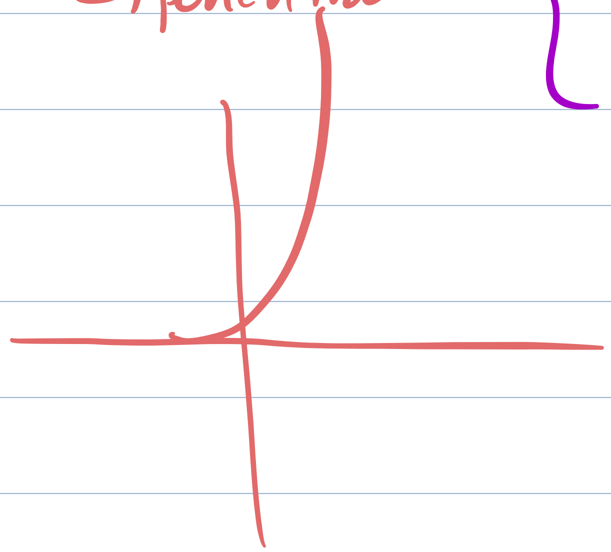
$\{w_1, w_2, w_3\}$

candidate	satisfies constr.	score
$\{\}$	✓	0
$\{w_1\}$	✓	3
$\{w_2\}$	✓	6 *
$\{w_3\}$	✓	2
$\{w_1, w_2\}$	✗	9
$\{w_1, w_3\}$	✓	5
$\{w_2, w_3\}$	✗	8
$\{w_1, w_2, w_3\}$	✗	11

Optimal = the candidate that has the highest score and is valid.

Fact: There are 2^n subsets of a set of size n.

→ Exponential.



2^n means: every time my input data gets bigger, (meeting) the search space doubles in size.

≈ 250 meetings

The # of possible solutions $(= 2^{250})$ is more than the # of atoms in the universe.

Pseudocode:

$R = \{w_1, w_2, w_3\}$

```
R = set of meeting requests
b = 0   ← best_sol = None
for each subset r of R:   loop 2^n times
    if r is valid:
        s = score(r)
        if s > b:
```

$b = s$ ← best_sol = r

~~return b~~

return best_sol

How long does this take? (n meetings)

Looping 2^n times

In each repetition of the loop:

check if it's valid — n steps

compute the score — n steps

$$\approx 2^n \cdot (2n) = O(n \cdot 2^n) \text{ time}$$

↑ big-O notation

Knapsack: Same situation as WIS.

n items

Search space: all subsets of the n items

Size of search space: 2^n

Closest Pair:

Input: n points in the 2D x,y-plane.

Goal: Find the pair of points that is closest (normal Euclidean distance) to each other.

P = set of points = $\{p_1, p_2, \ldots, p_n\}$

Search space:

all pairs of distinct points

$\{(p_1, p_2), ~~(p_2, p_1)~~, (p_1, p_3), \ldots\}$

unordered

The size is $O(n^2)$

Exact size: $\binom{n}{2} = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2}$

$\Rightarrow O(n^2)$