Monday, Feb 13, 2023
Lecture #12
MSSC 6000

Announcements

* HW 2 due Wednesday, Feb 22, 11:59pm
* Office Hours today, 1pm-2pm, CU 307

Lecture 5 - Search Space and Brute Force

Most of our problems can be summarized as:

"Out of all ways to do [blank]:
(1) Do any of them satisfy a list of constraints
and/or
(2) Which one is optimal?"

Greedy algos. give us a quick way to get a [blank] that might be decent.

but in most cases is not guaranteed
to be optimal.

They don't check every [blank]. Usually they only check a single one.

The <u>search space</u> of a problem is the set of all possible "things" that may or may not satisfy the contraints, and that may have a <u>score</u> that you want to minimize or maximize.

The next few lectures: ways to find optimal solutions by either checking the entire search space or (usually) not.

Most obvious: <u>brute force</u>

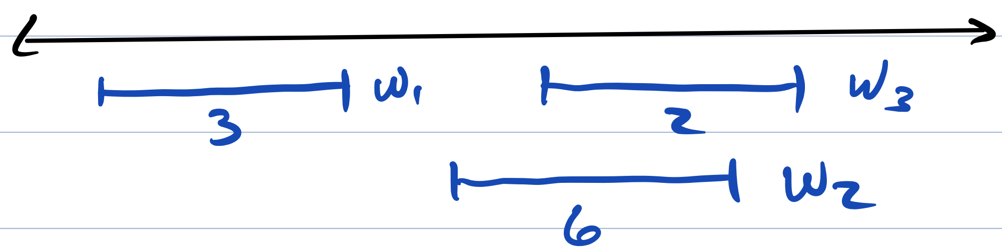
generate every single element in the search space one-by-one

- check if it satisfies the constr.

- if so, check its score

Ex 1: Weighted Interval Scheduling

3 requests



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

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 sol: a solution that satisfies constr. and has highest poss. score

Question: If you have a set of size n, how many subsets does it have?

2^n

Pseudocode

```
R = set of requests                      n = |R|
b = 0
for each subset r of R:                  loops 2^n times
    if r satisfies constraints:          have to go through each element of r to do these steps
        s = score(r)
        if s > b:
            b = s
return b
```

How long does this take to run?

Runtime is $\underbrace{O(n \cdot 2^n)}_{\text{"big-O notation"}}$ roughly $n \cdot 2^n$ operations for the code to finish

$3 \cdot n \cdot 2^n$ is also $O(n \cdot 2^n)$

big-O: ignore constant multipliers
ignore smaller terms

$5 \cdot n \cdot 2^n + \underbrace{7 \cdot n^2}_{\text{insignificant as } n \to \infty}$ is also $O(n \cdot 2^n)$

Knapsack Problem: same situation
n items
search space = all subsets of n items
size is 2^n

Closest Pair Problem:
Input: n points in the xy-plane
Goal: Find the pair that is closest
(normal Euclidean distance)
Search Space = all unordered pairs of points

Suppose our points are $\{p_1, p_2, p_3, p_4\}$. 6

The search space is

$$\{ \{p_1, p_2\}, \{p_1, p_3\}, \{p_1, p_4\}, \{p_2, p_3\}, \{p_2, p_4\}, \{p_3, p_4\} \}$$

6 pairs

If we have n points, what's the size of our search space?

$$\cancel{(n-1)! = (n-1)\cdot(n-2)\cdot(n-3)\cdots 3\cdot 2\cdot 1}$$

"n choose 2" = the # of ways to pick 2 things out of n, when the order doesn't matter

Notation: $\binom{n}{2}$

Formula: $\binom{n}{2} = \dfrac{n\cdot(n-1)}{2}$.

$$= \frac{n^2}{2} - \cancel{\frac{n}{2}}$$

$$= \frac{1}{2}\cdot n^2 = O(n^2)$$

(polynomial-time, not exp. time)

100 points

$$\binom{100}{2} = \frac{100\cdot 99}{2} = 50\cdot 99 = \ldots$$

* Surprising: there is a way to do this in $O(n \cdot \log(n))$ time. (next lecture)

Gamestop Problem from HW2:

How can we think of each possible solution (whether or not the constraints are satisfied)?

n people, 60 transaction slots

A candidate is an assignment of 60 of those n people into 60 transaction slots. How many ways can that be done?

Slot 1: n people
Slot 2: $n-1$ people
Slot 3: $n-2$ people
⋮
Slot 60: $n-59$ people

Search space: All ordered lists of 60 people out of n (8)

Size: $n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-59)$
$= n^{60} + ?\, n^{59} + ?\, n^{58} + \ldots$
$O(n^{60})$

Good news: polynomial, not exponential
Bad news: huge power of n

NFL schedules: search space for 1 week
= all ways of putting 32 teams in pairs
$= 31 \cdot 29 \cdot 27 \cdot 25 \cdot \ldots \cdot 5 \cdot 3 \cdot 1$

17 weeks: this # to the 17th power
$\approx 6.5 \times 10^{296}$

10^{80}