1 Sept. 2023

Cost of Operations on...

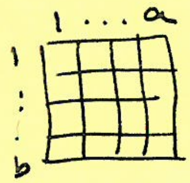
① Arrays

→ $\Theta(1)$ access to a given element by index
- access = read, write

→ $\Theta(n)$ to delete a given element

→ $\Theta(n)$ to resize

→ 2D, 3D, ~~nd~~ arrays:



$a \times b$ matrix
$n = a \cdot b$

② Linked Lists

→ $\Theta(i)$ to access the ith element
- access = read, write

→ If I'm already at an elt, $\Theta(1)$ time to delete it, assuming doubly connected.

→ $\Theta(1)$ to delete /add to front of list

→ to delete third element: $\Theta(1)$

→ 2 common LL we see: FIFO, LIFO
queue stack

Problem: "What" S = a set of objects (e.g., points in the plane)

$n := |S|$

Want to find "the closest pair", that is $(a, b) \in S \times S$ such that $a \neq b$ and $\wedge$ $\text{dist}(a,b) \leq \text{dist}(c,d)$ $\forall\, c,d \in S$, $c \neq d$

Assume: dist (a,b) is $\Theta(1)$ operation.

Solution: "How" We try all pairs of points.

I don't care about the order

```
FIND CLOSEST PAIR (S)
1: pair = ∅; curdist = ∞
2: for s1 ∈ S
3:     for s2 ∈ S
4:         if s1 ≠ s2
5:             if dist(s1, s2) < curdist
6:                 pair ← (s1, s2)
7:                 curdist ← dist(s1, s2)
8:             end if
9:         endif
10:    end for
11: end for
12: return pair
```

Each iter through, lines 4-9 are $\Theta(1)$ time.
With nested for loops, they repeat $\Theta(n^2)$ times.

$$RT = \overset{\text{Line 1}}{\Theta(1)} + \overset{\text{Line 2-11}}{\Theta(n^2)} + \Theta(1) = \Theta(n^2)$$

I can do better!

Let's assume S is an array.

for convenience, could compute on line).

FIND CLOSEST PAIR FAST (S, n)

```
pair ∅ ; curdist = ∞
for i = 1, 2, ..., n-1
    for j = i+1, i+2, ..., n
        if dist (s_i, s_j) < curdist
            pair ← (s_i, s_j)
            curdist ← dist (s_i, s_j)
        endif
    end for
end for
return pair
```

What is the cost of iter i in the outer for loop?

Inner for loop goes $n-(i+1)+1$ times, each time costs $\Theta(1)$

$$\sum_{i=1}^{n-1} n-(i+1)+1 = \sum_{i=1}^{n-1} n-i = (n-1)+(n-2)+\ldots \underbrace{(n-(n-1))}_{=1}$$

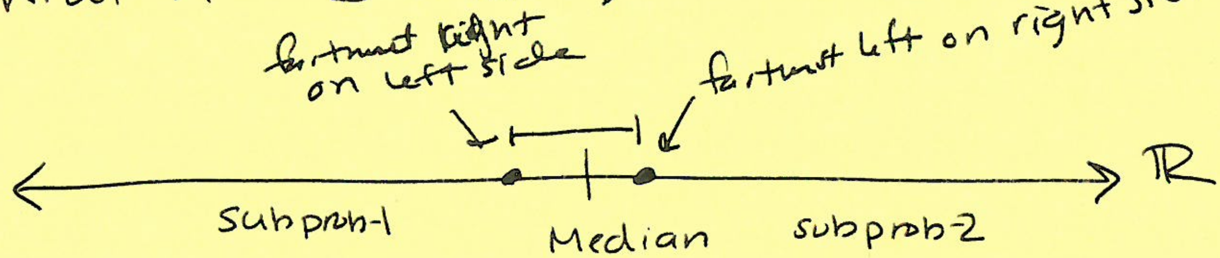
$$= \sum_{i=1}^{n-1} i$$

$$= \frac{(n-1)(n-1+1)}{2}$$

$$= \Theta(n^2)$$

In general, we can't do better, unless we know something about the structure / geometry of our data.

e.g. what if $S \subset \mathbb{R}^1$?



$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n \log n)$$

before divide: need to divide into 2 sets, so need to check every pt $\Theta(n)$

after we've conquered, we have 3 options to consider $\Theta(1)$

Alternatively: ① sort

② linearly walk through, only considering adjacent pairs of points.

Algorithms you present should have the following

① What? State the problem,
independent of the solution.

② How? Give the algorithm, clearly.
Assumed to be efficient in worst case.

③ Why? Why does this work
Proof of Correctness, inc. loop/recursion invariant.

④ How fast? Give the runtime,
with justification.

⑤ Could also consider (sometimes)

- space complexity
- # external messages passed
- specific resource consumption
- degree of predicates used

$a^2 + b$ "easier" than "$a^3 + b + c$"