CS 4800: Algorithms & Data

Lecture 5

January 24, 2017

Median

Problem: given a list of n elements, find the element of rank n/2 (half are larger, half are smaller)

can generalize to i

first solution: sort and pluck.

$$\Theta(n \log n)$$

Problem: given a list of n elements, find the element of rank i.

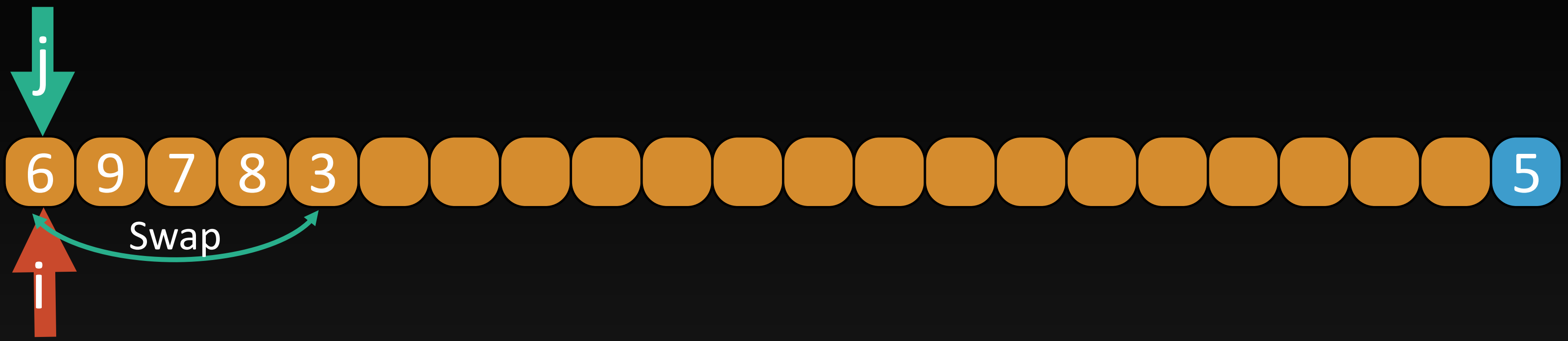
key insight:

Only need partial ordering

- Pick an element p
- Partition the list using p as pivot
- Recurse on the side containing i^{th} element

Partition a list

Partition a list



- i ← l //A[l..i-1] will be the elements < p
- For $j \leftarrow l$ to $r - 1$
  - If $A[j] < p$ then
    - Swap A[i] and A[j]
    - i ← $i + 1$
- Swap A[i] and A[r]

Select algorithm

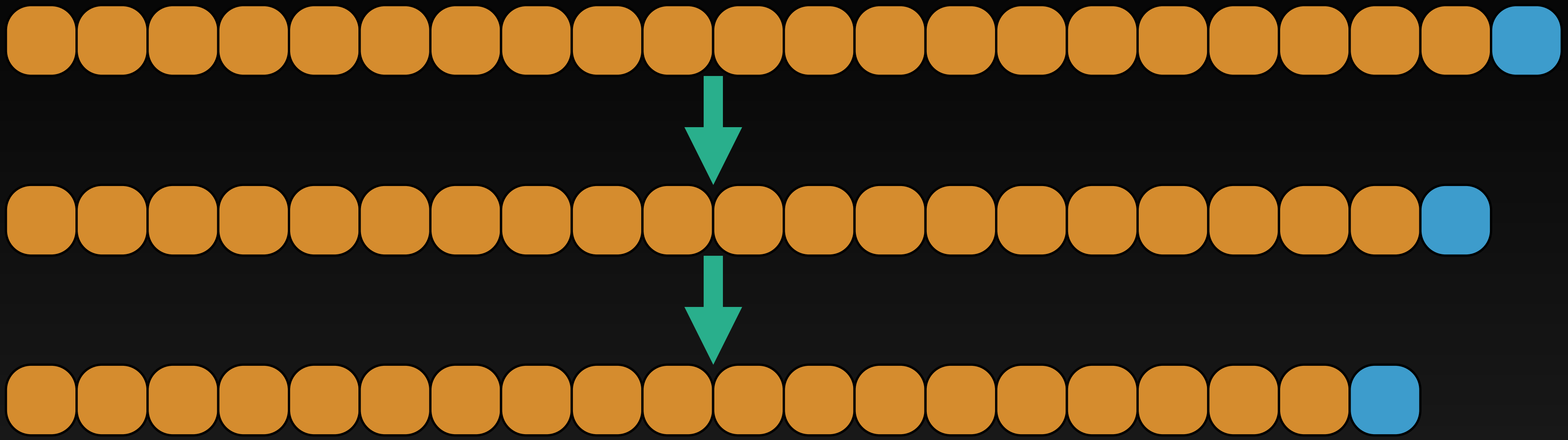
select(A[1, ..., n], i)

- Handle base case n=1
- pivot = a[n]
- Partition about pivot, resulting in pivot at position r
- If i = r, return pivot
- If i < r, select(A[1,...,r-1], i)
- If i > r, select(A[r+1, ..., n], i-r)

Assume equal partition every time

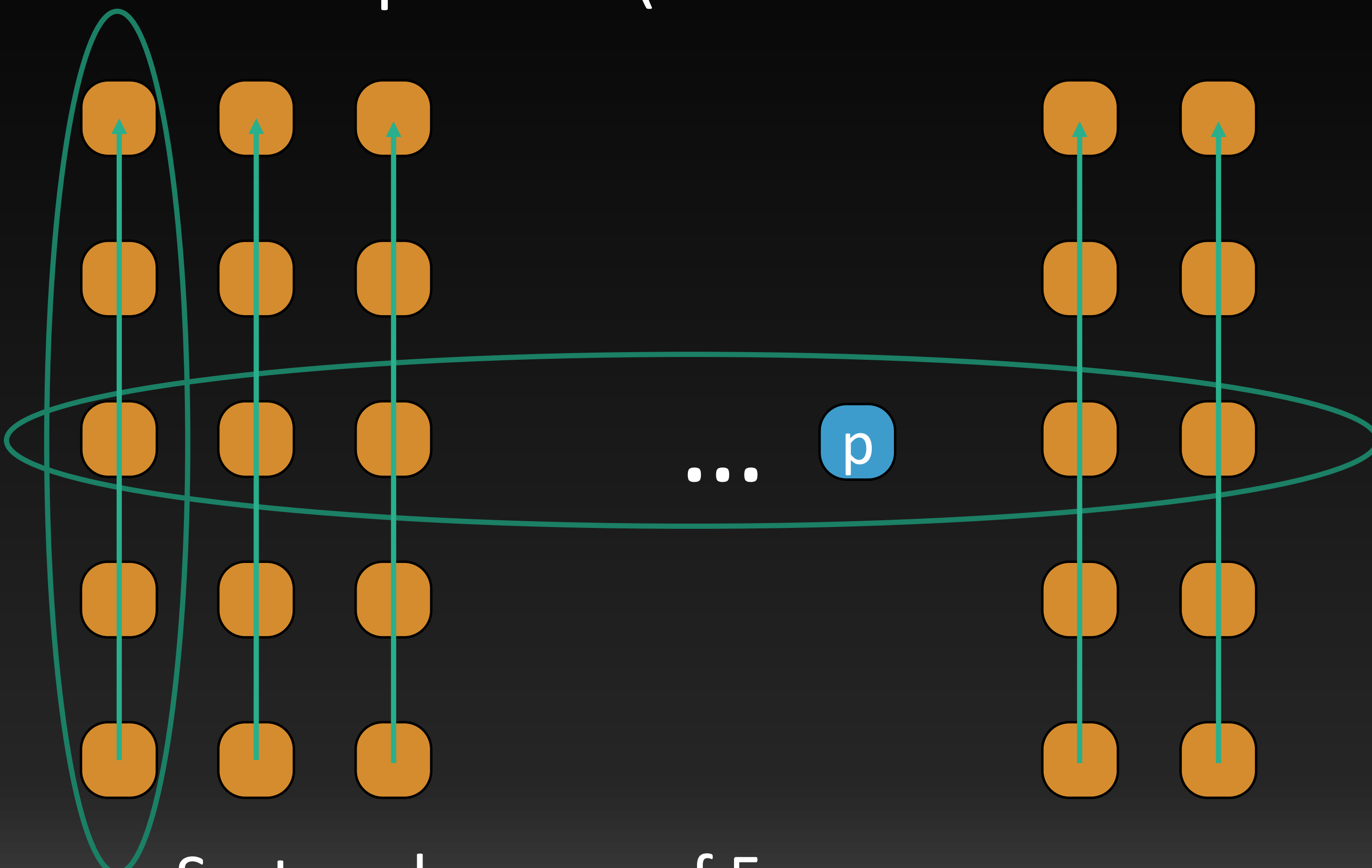
$$T(n) = T(n/2) + O(n)$$

Bad pivots? e.g. sorted array



$$\Omega(n^2)$$

Good pivot (median of medians)



Recurse to find median of medians

Sort each group of 5

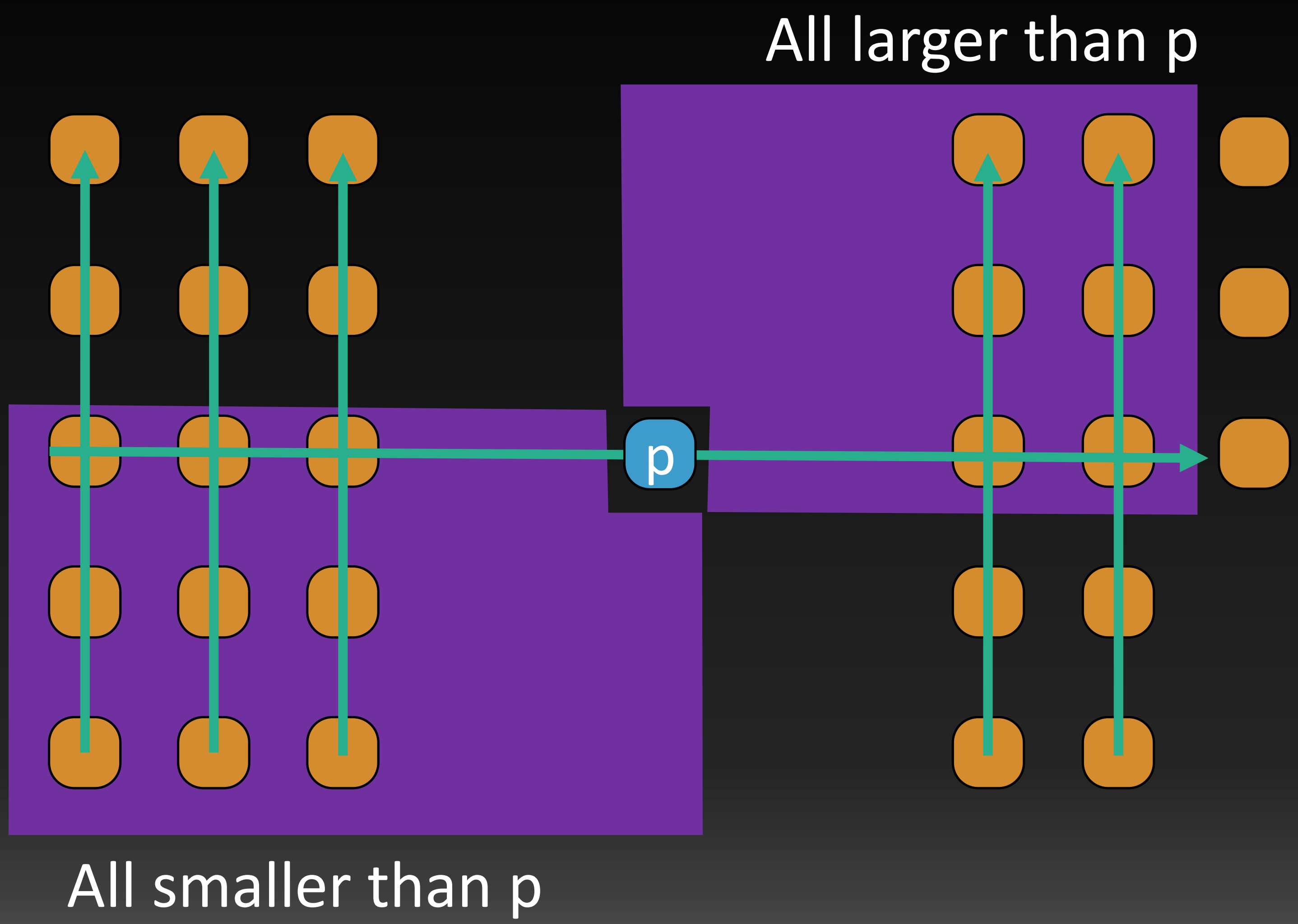
Time to find pivot

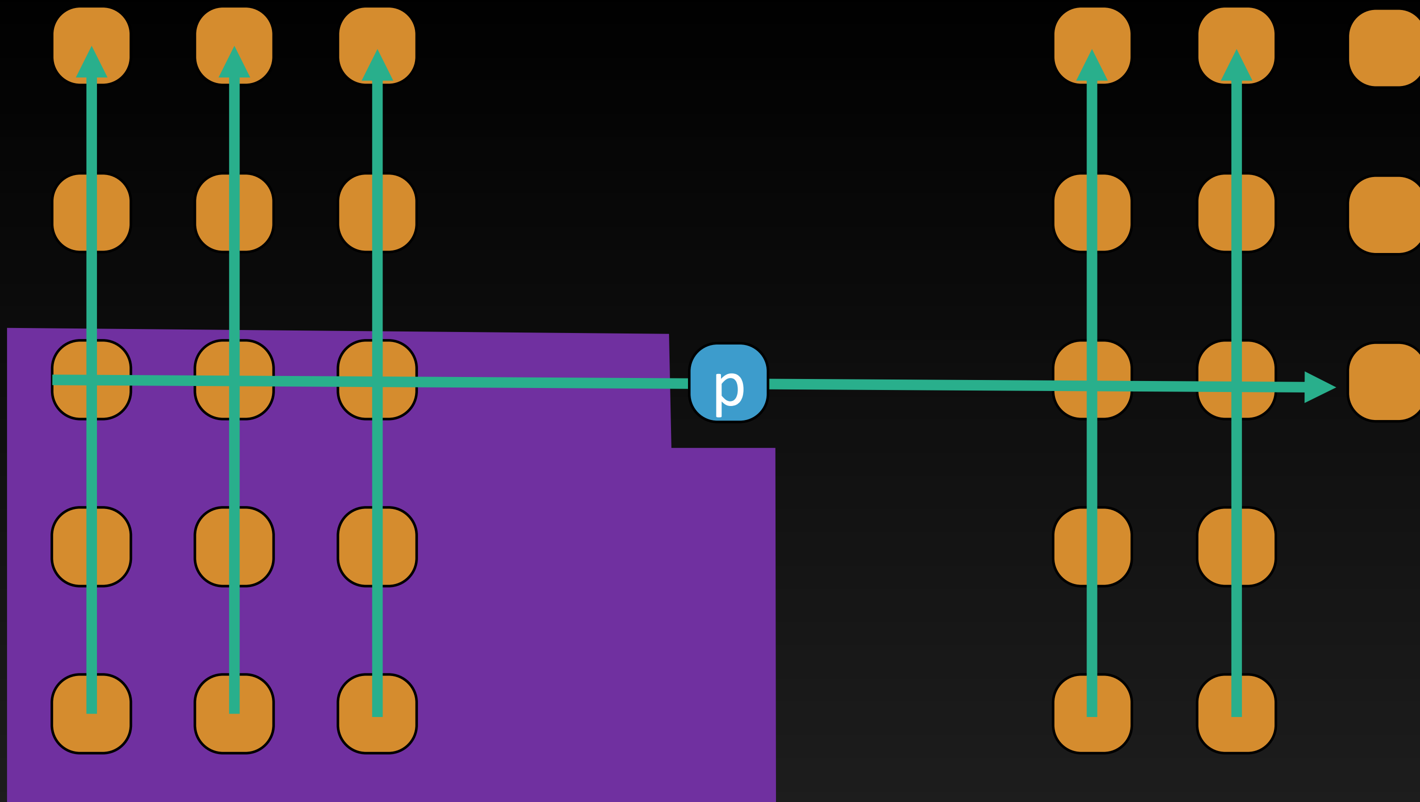
- $P(n)$: time to find pivot
- $S(n)$: time to select
- $P(n) = S(n/5) + O(n)$

select(A[1, ..., n],i)

- Handle base cases n < 15
- $pivot \leftarrow pivot(A[1, \dots, n])$
- Partition about pivot, resulting in pivot at position r
- If i = r, return pivot
- If i < r, select(A[1,…,r-1], i)
- If i > r, select(A[r+1, ..., n], i-r)

Quality of pivot





All smaller than p

$\lfloor n/5 \rfloor$ groups of 5 $\quad \geq n/5$-1 groups

$\geq \left\lceil (\frac{n}{5} - 1)/2 \right\rceil$ groups to the left of p (including p’s group) $\quad \geq (\frac{n}{5} - 1)/2$

$\geq \frac{3\left(\frac{n}{5}-1\right)}{2}$ elements $\leq p$ (3 elements per group) $\quad \geq \frac{3n}{10} - \frac{3}{2}$ elements

How many elts smaller than pivot?

- The rank of pivot $\geq \frac{3n}{10} - \frac{3}{2}$
- Similarly, the number of elements $\geq$ pivot is at least $\frac{3n}{10} - \frac{3}{2}$
- On recursive call, at most $n - \left(\frac{3n}{10} - \frac{3}{2}\right) = \frac{7n}{10} + \frac{3}{2}$ elements remain

Running time

- $S(n) = S\left(\frac{n}{5}\right) + S\left(\frac{7n}{10} + \frac{3}{2}\right) + O(n)$

Discussion problem

- $S(n) = S\left(\frac{n}{5}\right) + S\left(\frac{7n}{10} + \frac{3}{2}\right) + c \cdot n$
- Prove by induction that $S(n) = O(n)$

- $S(n) = S\left(\frac{n}{5}\right) + S\left(\frac{7n}{10} + \frac{3}{2}\right) + c \cdot n$
- Prove by induction that $S(n) = O(n)$
- Hypothesis : $S(n) \leq d \cdot n$ for constant d
- Base case $n \leq 30$: can pick d large enough for all base cases
- Inductive case: assume true for $n < k$, will prove for $n = k > 30$
- $S(k) = S\left(\frac{k}{5}\right) + S\left(\frac{7k}{10} + \frac{3}{2}\right) + c \cdot k$
- By assumption, $S(k) \leq d \cdot \frac{k}{5} + d \cdot \left(\frac{7k}{10} + \frac{3}{2}\right) + c \cdot k$
- $S(k) \leq d \cdot \left(\frac{9k}{10} + \frac{3}{2}\right) + c \cdot k$

$$\leq d \cdot \left(\frac{19k}{20}\right) + c \cdot k$$

- The RHS is $\leq dk$ if d > 20c (which we can ensure by picking d as large as need)