Computer Science 112
Data Structures

Lecture 24:
Quicksort

Coming Events

- **Next lecture, Monday April 27: Review**
- **Thursday, April 30: continuation of review if needed**
- **Monday, April 4: No lecture**
- **Monday, May 11, 4-7 pm final exam**
  - **rooms to be announced**

Review: Cost of Dijkstra's Algorithm

What are the operations to consider?

- **Picking the min-distance vertices from the fringe**
- **Adding vertices to the Tree**
- **Updating neighbors**
  - **Adding vertices to the Fringe**
  - **Updating links when needed**

Data Structures

- **For graph:**
  - adjacency matrix
  - adjacency list
- **For fringe:**
  - unordered array or linked list
  - ordered array or linked list
  - min heap

plus tree/fringe/neither marked on node

Tree as Adjacency list, Fringe as Unordered linked list

- Picking and removing min-distance vertices from the fringe
  - Worst case fringe is all vertices that are not in tree
    $(n-1) + (n-2) + \ldots + 1 = O(n^2)$
- Adding vertices to the Tree
  - $O(n)$
- Updating neighbors
  - Adding vertices to the Fringe: $O(n)$
  - Checking and Updating links: $O(n+e)$
- Total: $O(n^2) + O(n) + O(n+e) = O(n^2)$

Tree as Adjacency list, Fringe as min-heap

- Picking the min-distance vertices from the fringe
  - Worst case fringe is all vertices that are not in tree

    $\log(n-1) + \log(n-2) + \ldots + \log(1) = O(n \log n)$
- Adding vertices to the Tree
  - $O(n)$
- Updating neighbors
  - Adding vertices to the Fringe: $O(n \log n)$
  - Checking and Updating links: $O(n + (e \log n))$
- Total:

  $O(n \log n) + O(n + (e \log n))$

  $= O((n+e) \log n)$

Review: Quicksort

- **Quicksort:**
  - **Partition**
    - **Split data into two groups, all in one group < any in other group**
  - **sort groups separately**
    - **use quicksort recursively**
  - **append**
    - **if partition & sort are in-place there is nothing to do here**

Quick Sort

Unsorted:	4	5	1	8	3	9	2
pivot=4							
Partition:	$\boxed{3 \quad 2 \quad 1}$			4	$\boxed{8 \quad 9 \quad 5}$		
Sort Groups:	$\boxed{1 \quad 2 \quad 3}$			4	$\boxed{5 \quad 8 \quad 9}$		
Result:	1	2	3	4	5	8	9

Partition

- **Choose a "pivot" value from data**
  - **simplest: choose first data value, A[lo]**

Partition

- Use 2 pointers: left and right
  - move left from lo+1 up until A[left] > pivot
  - move right from hi down until A[right]<pivot
  - Swap numbers in A[left] and A[right]
  - Repeat until left>=right

Partition

30	25	24	53	47	16	19	20	35	28	41
30	25	24	53	47	16	19	20	35	28	41
30	25	24	28	47	16	19	20	35	53	41
30	25	24	28	20	16	19	47	35	53	41
19	25	24	28	20	16	30	47	35	53	41

Quicksort

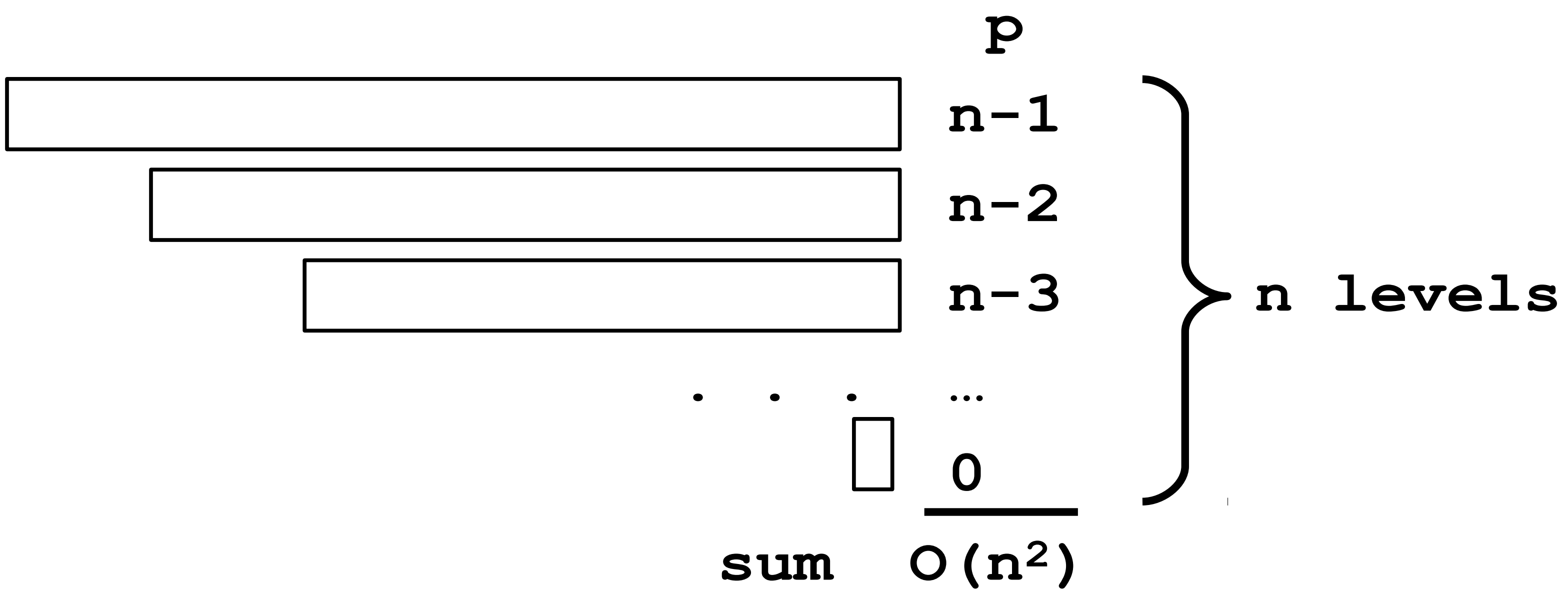
- **How sort regions left & right of pivot?**
  - **Quicksort! (unless $<$ 3 numbers in region)**

Quick Sort

30	25	24	53	47	16	19	20	35	28	41
19	25	24	28	20	16	30	47	35	53	41
16	19	24	28	20	25		41	35	47	53
		20	24	28	25		35	41		
				25	28					

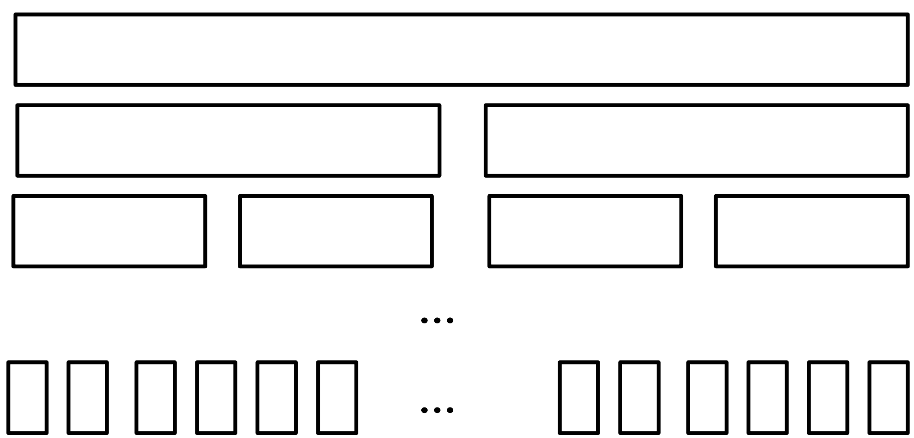
Worst Case Complexity

- **Partition takes O(p) time where p is the number of numbers to partition**
- **<u>Worst</u> case: each pivot is smallest of the numbers: results in regions of 0 and p-1 numbers**



Best Case Complexity

- **<u>Best</u> case: each pivot is the median of the numbers: results in two groups of p/2**



```
 p
1* n   =n  ⎫
2* n/2=n   ⎪
4* n/4=n   ⎬  log n
 .  .  .   ⎪  levels
n* n/n=n   ⎭
sum   O(n log n)
```

```
x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x
x x x x x x x x x x x x x
x x x x x x x x x x x x
x x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x
x x x x x x x x
x x x x x x x
x x x x x x
x x x x x
x x x x
x x x
x x
x
```

```
x x x x x x x x x x x x x x x
x x x x x x x   x x x x x x x
x x x   x x x   x x x   x x x
x   x   x   x   x   x   x   x
```

x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x

x x x x

x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x

x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
x x x x x x x x x x

Average Case Complexity

- **<u>Average</u> case: O(n log n), like best case**
- **In practice, on the average, for large arrays, quick sort is the fastest (in terms of real time) sort**

Improvements to quicksort

- **Choose a better pivot:**
  - **median of a[lo], a[hi], a[(lo+hi)/2]**
  - **makes worst case less likely**
- **If region is < 10 numbers long use insertion sort**
- **When recurring on regions, do smaller region first**

You do a quicksort

- **Use median to choose pivot, but do not use insertion sort on small regions**

31 12 47 33 41 25 40