S. KIMMEL

Goals

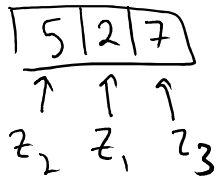
- Analyze average runtime of QuickSort
- Compare & contrast shortest path algorithms

Reflections: hard... how can I help Quiz: no quiz, Spring Symposium extra credit

Back to QuickSort:

$R = \sum_{i,j} X_{i,j}$ ← # comparisons b/t i^{th} largest + j^{th} largest array elements on an element of sample space

Let z_i = i^{th} smallest element of A



Q: Under what circumstances are z_i and z_j never compared?

A) If z_i or z_j is chosen as pivot at some point.

B) If z_k is chosen as pivot, where $k > i, j$

C) If z_k is chosen as pivot, where $i < k < j$

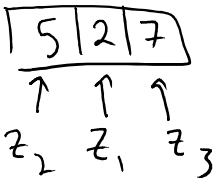
D) If z_k is chosen as pivot, where $k < i, j$

S. KIMMEL

Back to QuickSort:

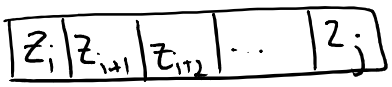
$R = \sum_{i<j} X_{ij}$ ← # comparisons b/t i^{th} largest + j^{th} largest array elements on element of sample space

Let z_i = i^{th} largest element of A



Story of z_i, z_j:

- As long as pivot $= z_k$ with $k > i, j$ or $k < i, j$, z_i, z_j get put together into same subarray for recursion. (No comparisons only)
- Something interesting happens when pivot is z_k with $i \le k \le j$



Because X_{ij} is 0 or 1, this means X_{ij} is an indicator random variable!

$k = i$ or j

z_i, z_j

1 comparison

(No further comparisons because pivot is not included in recursive calls)

$i < k < j$

z_i, z_j

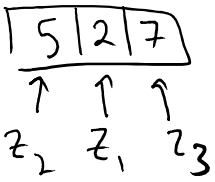
0 comparisons

(No further comparisons because z_i and z_j separated: z_i in A_L, z_j in A_R.)

Back to QuickSort:

$R = \sum_{i<j} X_{ij}$ ← # comparisons b/t i^{th} largest + j^{th} largest array elements on an element of sample space

Let z_i = i^{th} smallest element of A



Calculate Average # of comparisons

$$E[R] = \sum_{i<j} E[X_{ij}]$$

Because indicator random variable

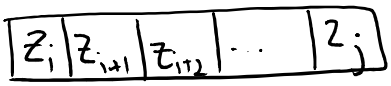
$$E[X_{ij}] = \sum_{s \in S} Pr(s) X_{ij}(s) = \sum_{\substack{s \in S: \\ X_{ij}(s)=1}} Pr(s)$$

definition of expectated value

$$= Pr(z_i, z_j \text{ are compared})$$

definition of event probability

Choice of pivot only has an effect if chosen from



Size of this chunk is $j-i+1$

Eventually, will have to choose a pivot from here.

Probability (z_i z_j compared) = Pr(z_i, or z_j chosen as pivot if an element of $\{z_i, \dots z_j\}$ is chosen as pivot)

Probability of

- 1 comparison: $\frac{2}{j-i+1}$ b/c 2 options (z_i or z_j) out of $j-i+1$ options give 1 comparison.
- 0 comparisons: $\frac{j-i-1}{j-i+1} = 1 - \frac{2}{j-i+1}$

$$\Pr[\text{1 comparison}] = \frac{2}{j-i+1}$$

S. KIMMEL

Final Calculation:

$$\mathbb{E}[R] = \sum_{i,j} \mathbb{E}[X_{i,j}] = \sum_{i,j} \Pr[z_i, z_j \text{ compared}]$$

$$\mathbb{E}[R] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[z_i, z_j \text{ compared}]$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

$$= \sum_{i=1}^{n-1} 2\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \frac{1}{n-i}\right)$$

$$\leq \sum_{i=1}^{n-1} 2\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \frac{1}{n-i} + \cdots + \frac{1}{n}\right)$$ (added extra positive terms to sum)

$$\leq \sum_{i=1}^{n-1} 2\ln(n) \leftarrow$$

$$\leq 2n \ln n$$

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

Useful fact:

$$\sum_{j=1}^{n} \frac{1}{j} \leq \ln(n)$$

Average runtime $= O(\text{Average \# of comparisons})$

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

S. KIMMEL

Shortest Paths

E = set of edges
V = set of vertices

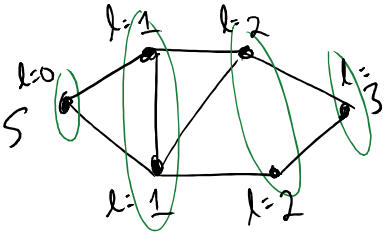
Input: Unweighted, undirected Graph $G=(V,E)$, starting node $s \in V$

Output: Array l, $l[v]$ = length of shortest path from s to v.

($l[v] = \infty$ if no path from s to v)

Applications? Maps, social connections, finacial transactions

Idea: explore each distance layer in turn:



Similar? Breadth-First Search!

```
l[v] = ∞  ∀ v ∈ V
vis[v] = false  ∀ v ∈ V      //make true after visiting
A = {}                       // create queue
A.add(s)                     // initialize with vertex s
l[s] = 0
vis[s] = true
while (A is not empty)
    - v = A.pop
    - for w: {v,w} ∈ E :
        if (vis[w] = false):
            • A.add(w)
            • vis[w] = true
            • l[w] = l[v] + 1.
```

If remove lines involving l, have BFS

S. KIMMEL

Q: Will the algorithm be correct under following conditions? (If not always, provide counter example.)

- directed
- positive weighted edges
- negative weights (uniform)

- If edge weights are positive integers, how could you alter algorithm to make it work?
- What is time complexity of graph search if you store G in adjacency list data structure?