**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:**

\[ R = \sum_{i} x_{ij} \]

# comparisons b/t i\textsuperscript{th} largest

+ j\textsuperscript{th} largest array elements on an element of sample space

Let \( z_i = \text{i\textsuperscript{th} smallest element of A} \)

\[ 15127 \]

\[ \uparrow \uparrow \uparrow \]

\[ z_2, z_1, z_3 \]

**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 \( k < i,j \)
Back to QuickSort:

\[ R = \sum_{i,j} X_{i,j} \quad \text{# comparisons b/t } i^{\text{th}} \text{ largest } \\
\quad \quad + j^{\text{th}} \text{ largest array elements on element of sample space} \]

Let \( z_i = i^{\text{th}} \) largest element of \( A \)

\[
\begin{bmatrix}
5 & 2 & 7 \\
\uparrow & \uparrow & \uparrow \\
2 & z_i & z_3
\end{bmatrix}
\]

\textbf{Story of } z_i, z_j:

\begin{itemize}
  \item 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)
  \item Something interesting happens when pivot is \( z_k \) with \( i \leq k \leq j \)
\end{itemize}

Because \( X_{i,j} \) is 0 or 1, this means \( X_{i,j} \) is an indicator random variable.

\( k = i \text{ or } j \)

\( z_i, z_j \) 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:

\[ R = \sum_{ij} X_{ij} \]  
\( \) # 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 \( A \)
\[
\begin{array}{c}
5 \\
\uparrow \uparrow \uparrow \\
2_1, 2, 2_3
\end{array}
\]

Calculate Average # of comparisons

\[ E[R] = \sum_{ij} E[X_{ij}] \]

Because indicator random variable

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

= \( Pr(z_i, z_j \text{ are compared}) \)
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 (\( z_i \) or \( z_j \) compared) = \( \Pr(\text{\( z_i \) or \( z_j \) chosen as pivot if an element of \{\( z_i \), ..., \( z_j \}\} is chosen as pivot}) \)

Probability of

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

* 0 comparisons: \( \frac{j-i-1}{j-i+1} = 1 - \frac{2}{j-i+1} \)

\( \Pr[\text{1 comparison}] = \frac{2}{j-i+1} \)
Final Calculation:

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

\[ E[R] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \]

\[ \leq \sum_{i=1}^{n-1} 2 \left( \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n-i} \right) \]

\[ \leq \sum_{i=1}^{n-1} 2 \left( \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n} \right) \quad (\text{added extra positive terms to sum}) \]

\[ \leq 2 \ln(n) \leq \ln(n) \quad \text{Useful fact:} \]

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

Average runtime: \( O(\text{Average # of comparisons}) = O(n \log n) \)
Shortest Paths

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, financial transactions

Idea: explore each distance layer in turn:

\[
\begin{align*}
l[v] &= \infty \quad \forall \ v \in V \\
\text{vis}[v] &= \text{false} \quad \forall \ v \in V \\
A &= \{\} \\
A.\text{add}(s) \\
l[s] &= 0 \\
\text{vis}[s] &= \text{true}
\end{align*}
\]

\[
\text{While (A is not empty) }
\]
\[
\quad - \ v = A.\text{pop} \\
\quad - \text{for } w: \{v,w\} \in E : \\
\quad \quad \text{If } (\text{vis}[w] = \text{false}) : \\
\quad \quad \quad \text{A.\text{add}(w)} \\
\quad \quad \quad \text{vis}[w] = \text{true} \\
\quad \quad \quad l[w] = l[v] + 1
\]

Similar?
Breadth-First Search!

If remove lines involving $l$, have BFS
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?