The Randomized Quicksort Algorithm

K. Subramani$^1$

$^1$Lane Department of Computer Science and Electrical Engineering
West Virginia University

7 February, 2012
The Randomized Quicksort Algorithm
The Sorting Problem

Problem Statement

The Sorting Problem

Problem Statement

Given an array $A$ of $n$ distinct integers, in the indices $A[1]$ through $A[n]$, permute the elements of $A$, so that
The Sorting Problem

Problem Statement

The Sorting Problem

**Problem Statement**


**Note**

*The assumption of distinctness simplifies the analysis.*
### Problem Statement


### Note

*The assumption of distinctness simplifies the analysis. It has no bearing on the running time.*
The Partition subroutine

Function $\text{PARTITION}(A, p, q)$

1: {We partition the sub-array $A[p, p + 1, \ldots, q]$ about $A[p]$.}
2: for $(i = (p + 1)$ to $q)$ do
5: else
7: Insert $A[i]$ into bucket $U$.
8: end if
9: end if
10: end for
12: Copy the elements of $L$ into the first $|L|$ entries of $A[p \cdot q]$.
14: Copy the elements of $U$ into the entries of $A[(|L| + 2) \cdot q]$.
15: return $(|L| + 1)$. 

Subramani
Sample Analyses
The Partition subroutine

**Function** \( \text{PARTITION}(A, p, q) \)

1. \{ We partition the sub-array \( A[p, p+1, \ldots, q] \) about \( A[p] \). \}
2. \textbf{for} \( i = (p + 1) \textbf{ to } q \) \textbf{do}
3. \hspace{1em} if \( A[i] < A[p] \) \textbf{then}
4. \hspace{2em} Insert \( A[i] \) into bucket \( L \).
5. \hspace{1em} \textbf{else}
6. \hspace{2em} if \( A[i] > A[p] \) \textbf{then}
7. \hspace{3em} Insert \( A[i] \) into bucket \( U \).
8. \hspace{2em} \textbf{end if}
9. \hspace{1em} \textbf{end if}
10. \textbf{end for}
11. Copy \( A[p] \) into \( A[(|L| + 1)] \).
12. Copy the elements of \( L \) into the first \( |L| \) entries of \( A[p \cdot q] \).
13. Copy \( A[p] \) into \( A[(|L| + 1)] \).
14. Copy the elements of \( U \) into the entries of \( A[(|L| + 2) \cdot q] \).
15. \textbf{return} \( (|L| + 1) \).

**Note**

Partitioning an array can be achieved in linear time.
Function **QUICKSORT**(A, p, q)

1: if \( p \geq q \) then
2: return
3: else
4: \( j = \text{PARTITION}(A, p, q) \).
5: Quicksort(A, p, j - 1).
6: Quicksort(A, j + 1, q).
7: end if
The Randomized Quicksort Algorithm

The Quicksort Algorithm

Function \textsc{Quicksort}(A, p, q)

1: if \((p \geq q)\) then
2: \hspace{1em} return
3: \textbf{else}
4: \hspace{1em} \(j = \textsc{Partition}(A, p, q)\).
5: \hspace{1em} \textsc{Quicksort}(A, p, j - 1).
6: \hspace{1em} \textsc{Quicksort}(A, j + 1, q).
7: \textbf{end if}

\textbf{Note}

The main program calls \textsc{Quicksort}(A, 1, n).
Worst-case analysis

Analysis

What is the worst-case input for QUICKSORT()?
Worst-case analysis

Analysis

What is the worst-case input for `QUICKSORT()`? How many comparisons in the worst case?
Worst-case analysis

Analysis

What is the worst-case input for QUICKSORT()? How many comparisons in the worst case? $O(n^2)$. 
### Analysis

What is the worst-case input for `QUICKSORT()`? How many comparisons in the worst case? $O(n^2)$.

### Intuition for randomized case

What sort of assumptions are reasonable in analysis?
Randomized Quicksort
Randomized Quicksort

Function \textsc{Randomized-Quicksort}(A, p, q)

1: if \((p \geq q)\) then
2: \hspace{1em} return
3: else
4: \hspace{1em} Choose a number, say \(r\), uniformly and at random from the set \(\{p, p+1, \ldots, q\}\).
5: \hspace{1em} Swap \(A[p]\) and \(A[r]\).
6: \hspace{1em} \(j = \text{Partition}(A, p, q)\).
7: \hspace{1em} Quicksort(A, p, j − 1).
8: \hspace{1em} Quicksort(A, j + 1, q).
9: end if
Function RANDOMIZED-QUICKSORT(A, p, q)
1: if (p ≥ q) then
2:   return
3: else
4:   Choose a number, say r, uniformly and at random from the set \{p, p + 1, \ldots, q\}.
5:   Swap A[p] and A[r].
6:   j = PARTITION(A, p, q).
7:   Quicksort(A, p, j − 1).
8:   Quicksort(A, j + 1, q).
9: end if

Note

Worst case running time?
The Randomized Quicksort Algorithm

Randomized Quicksort

Function $\text{RANDOMIZED-QUICKSORT}(A, p, q)$

1: if $(p \geq q)$ then
2: return
3: else
4: Choose a number, say $r$, uniformly and at random from the set \{p, p+1, \ldots, q\}.
6: $j = \text{PARTITION}(A, p, q)$.
7: Quicksort($A, p, j-1$).
8: Quicksort($A, j+1, q$).
9: end if

Note

Worst case running time? $O(n^2)$!
**Randomized Quicksort**

**Function** `RANDOMIZED-QUICKSORT(A, p, q)`

1. if \((p \geq q)\) then
2. return
3. else
4. Choose a number, say \(r\), uniformly and at random from the set \(\{p, p+1, \ldots, q\}\).
5. Swap \(A[p]\) and \(A[r]\).
6. \(j =\text{PARTITION}(A, p, q)\).
7. \(\text{Quicksort}(A, p, j-1)\).
8. \(\text{Quicksort}(A, j+1, q)\).
9. end if

**Note**

*Worst case running time? \(O(n^2)\)! However, for a randomized algorithm we are not interested in worst-case running time, but in expected running time.*
Decision Tree Analysis
The Randomized Quicksort Algorithm

Decision Tree Analysis

**Decision Tree**

The operation of `RANDOMIZED QUICKSORT()` can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node.
The operation of \textsc{Randomized Quicksort()} can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree.
The operation of `RANDOMIZED QUICKSORT()` can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order.
The operation of **RANDOMIZED QUICKSORT()** can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree?
The operation of RANDOMIZED QUICKSORT() can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. 
The operation of \texttt{RANDOMIZED QUICKSORT()} can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. 
The Randomized Quicksort Algorithm

Decision Tree Analysis

The operation of **RANDOMIZED QUICKSORT()** can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.
The operation of \textsc{Randomized Quicksort}() can be thought of as a binary tree, say \( T \), with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of \( T \) focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? \( O(n) \). Let \( h \) denote the height of \( T \). Observe that \( h \) is a random variable and we are interested in its expected value. The rank of an element of \( A \) is its position in \( A \), when \( A \) has been sorted.
The operation of \textsc{Randomized Quicksort()} can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.

The rank of an element of $A$ is its position in $A$, when $A$ has been sorted. When you pick an element at random, what is the probability that the rank of the element chosen is between $\frac{1}{4} \cdot n$ and $\frac{3}{4} \cdot n$, where $n$ is the number of elements in the array?
Decision Tree Analysis

Decision Tree

The operation of RANDOMIZED QUICKSORT() can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.

The rank of an element of $A$ is its position in $A$, when $A$ has been sorted.

When you pick an element at random, what is the probability that the rank of the element chosen is between $\frac{1}{4} \cdot n$ and $\frac{3}{4} \cdot n$, where $n$ is the number of elements in the array? $\frac{1}{2}$.

Consider the tree $T$. 
The operation of **RANDOMIZED QUICKSORT()** can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.

The rank of an element of $A$ is its position in $A$, when $A$ has been sorted.

When you pick an element at random, what is the probability that the rank of the element chosen is between $\frac{1}{4} \cdot n$ and $\frac{3}{4} \cdot n$, where $n$ is the number of elements in the array? $\frac{1}{2}$.

Consider the tree $T$. We define an internal node $o$ of the tree to be **good**, if both its children have at most $\frac{3}{4} \cdot |o|$ nodes, where $|o|$ denotes the number of elements in the node $o$.  

### Decision Tree
The operation of **RANDOMIZED QUICKSORT()** can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.

The rank of an element of $A$ is its position in $A$, when $A$ has been sorted.

When you pick an element at random, what is the probability that the rank of the element chosen is between $\frac{1}{4} \cdot n$ and $\frac{3}{4} \cdot n$, where $n$ is the number of elements in the array? $\frac{1}{2}$.

Consider the tree $T$. We define an internal node $o$ of the tree to be **good**, if both its children have at most $\frac{3}{4} \cdot |o|$ nodes, where $|o|$ denotes the number of elements in the node $o$. 

### Decision Tree

![Decision Tree Diagram](image-url)
The operation of \texttt{RANDOMIZED QUICKSORT()} can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.

The rank of an element of $A$ is its position in $A$, when $A$ has been sorted.

When you pick an element at random, what is the probability that the rank of the element chosen is between $\frac{1}{4} \cdot n$ and $\frac{3}{4} \cdot n$, where $n$ is the number of elements in the array? $\frac{1}{2}$.

Consider the tree $T$. We define an internal node $o$ of the tree to be \textbf{good}, if both its children have at most $\frac{3}{4} \cdot |o|$ nodes, where $|o|$ denotes the number of elements in the node $o$. Given an internal node $o$ of $T$, what is the probability that it is \textbf{good}?
The operation of \texttt{RANDOMIZED QUICKSORT()} can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.

The rank of an element of $A$ is its position in $A$, when $A$ has been sorted.

When you pick an element at random, what is the probability that the rank of the element chosen is between $\frac{1}{4} \cdot n$ and $\frac{3}{4} \cdot n$, where $n$ is the number of elements in the array? $\frac{1}{2}$.

Consider the tree $T$. We define an internal node $o$ of the tree to be \texttt{good}, if both its children have at most $\frac{3}{4} \cdot |o|$ nodes, where $|o|$ denotes the number of elements in the node $o$. Given an internal node $o$ of $T$, what is the probability that it is \texttt{good}? At least $\frac{1}{2}$!
The operation of \textsc{Randomized Quicksort()} can be thought of as a binary tree, say \( T \), with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of \( T \) focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? \( O(n) \). Let \( h \) denote the height of \( T \). Observe that \( h \) is a random variable and we are interested in its expected value.

The rank of an element of \( A \) is its position in \( A \), when \( A \) has been sorted.

When you pick an element at random, what is the probability that the rank of the element chosen is between \( \frac{1}{4} \cdot n \) and \( \frac{3}{4} \cdot n \), where \( n \) is the number of elements in the array? \( \frac{1}{2} \).

Consider the tree \( T \). We define an internal node \( o \) of the tree to be \textbf{good}, if both its children have at most \( \frac{3}{4} \cdot |o| \) nodes, where \( |o| \) denotes the number of elements in the node \( o \). Given an internal node \( o \) of \( T \), what is the probability that it is \textbf{good}? At least \( \frac{1}{2} \)!

Consider a root to leaf path in \( T \).
The operation of `RANDOMIZED QUICKSORT()` can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.

The rank of an element of $A$ is its position in $A$, when $A$ has been sorted.

When you pick an element at random, what is the probability that the rank of the element chosen is between $\frac{1}{4} \cdot n$ and $\frac{3}{4} \cdot n$, where $n$ is the number of elements in the array? $\frac{1}{2}$.

Consider the tree $T$. We define an internal node $o$ of the tree to be **good**, if both its children have at most $\frac{3}{4} \cdot |o|$ nodes, where $|o|$ denotes the number of elements in the node $o$. Given an internal node $o$ of $T$, what is the probability that it is **good**? At least $\frac{1}{2}$!

Consider a root to leaf path in $T$. How many **good** nodes can exist on such a path?
Decision Tree Analysis

The operation of `RANDOMIZED QUICKSORT()` can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.

The rank of an element of $A$ is its position in $A$, when $A$ has been sorted.

When you pick an element at random, what is the probability that the rank of the element chosen is between $\frac{1}{4} \cdot n$ and $\frac{3}{4} \cdot n$, where $n$ is the number of elements in the array? $\frac{1}{2}$.

Consider the tree $T$. We define an internal node $o$ of the tree to be good, if both its children have at most $\frac{3}{4} \cdot |o|$ nodes, where $|o|$ denotes the number of elements in the node $o$. Given an internal node $o$ of $T$, what is the probability that it is good? At least $\frac{1}{2}$!

Consider a root to leaf path in $T$. How many good nodes can exist on such a path? At most $r = \log_4 3 \cdot n$. 
Decision Tree Analysis

Decision Tree

The operation of RANDOMIZED QUICKSORT() can be thought of as a binary tree, say $T$, with a pivot being chosen at each internal node. The elements in the node which are less than the pivot are shunted to the left subtree and the rest of the elements (excluding the pivot) are shunted to the right subtree. An in-order traversal of $T$ focusing on the pivots, gives the sorted order. What is the work done at each level of the tree? $O(n)$. Let $h$ denote the height of $T$. Observe that $h$ is a random variable and we are interested in its expected value.

The rank of an element of $A$ is its position in $A$, when $A$ has been sorted.

When you pick an element at random, what is the probability that the rank of the element chosen is between $\frac{1}{4} \cdot n$ and $\frac{3}{4} \cdot n$, where $n$ is the number of elements in the array? $\frac{1}{2}$.

Consider the tree $T$. We define an internal node $o$ of the tree to be good, if both its children have at most $\frac{3}{4} \cdot |o|$ nodes, where $|o|$ denotes the number of elements in the node $o$. Given an internal node $o$ of $T$, what is the probability that it is good? At least $\frac{1}{2}$!

Consider a root to leaf path in $T$. How many good nodes can exist on such a path? At most $r = \log_{\frac{3}{4}} n$. What is the expected number of nodes on a root to leaf path before you see $r$ good nodes?
Lemma

Consider a coin for which the probability of “heads” turning up on a toss is $p$. 
Lemma

Consider a coin for which the probability of “heads” turning up on a toss is $p$. What is the expected number of tosses to obtain $k$ heads?
Lemma

Consider a coin for which the probability of “heads” turning up on a toss is \( p \). What is the expected number of tosses to obtain \( k \) heads? \( \frac{k}{p} \).
Lemma

Consider a coin for which the probability of “heads” turning up on a toss is $p$. What is the expected number of tosses to obtain $k$ heads? $\frac{k}{p}$.

Decision Tree (contd.)

Thus the expected number of nodes on a root to leaf path is
Lemma

Consider a coin for which the probability of “heads” turning up on a toss is $p$. What is the expected number of tosses to obtain $k$ heads? $\frac{k}{p}$.

Decision Tree (contd.)

Thus the expected number of nodes on a root to leaf path is $\frac{r}{2} = $
Lemma

Consider a coin for which the probability of “heads” turning up on a toss is $p$. What is the expected number of tosses to obtain $k$ heads? $\frac{k}{p}$.

Decision Tree (contd.)

Thus the expected number of nodes on a root to leaf path is $\frac{r}{2} = 2 \cdot r$. 
Lemma

Consider a coin for which the probability of "heads" turning up on a toss is $p$. What is the expected number of tosses to obtain $k$ heads? $\frac{k}{p}$.

Decision Tree (contd.)

Thus the expected number of nodes on a root to leaf path is $\frac{r}{2} = 2 \cdot r = 2 \cdot \log_{\frac{4}{3}} n$. 
Lemma

Consider a coin for which the probability of “heads” turning up on a toss is $p$. What is the expected number of tosses to obtain $k$ heads? $rac{k}{p}$.

Decision Tree (contd.)

Thus the expected number of nodes on a root to leaf path is \( \frac{r}{2} = 2 \cdot r = 2 \cdot \log_{\frac{4}{3}} n \). However, this is the expected height of $T$, i.e., $E[h]$. 
Lemma

Consider a coin for which the probability of “heads” turning up on a toss is $p$. What is the expected number of tosses to obtain $k$ heads? \( \frac{k}{p} \).

Decision Tree (contd.)

Thus the expected number of nodes on a root to leaf path is \( \frac{r}{2} = 2 \cdot r = 2 \cdot \log_3 4 \cdot n \). However, this is the expected height of $T$, i.e., $E[h]$. Therefore, the expected work undertaken by the algorithm
Lemma

Consider a coin for which the probability of “heads” turning up on a toss is $p$. What is the expected number of tosses to obtain $k$ heads? $\frac{k}{p}$.

Decision Tree (contd.)

Thus the expected number of nodes on a root to leaf path is $\frac{1}{2} = 2 \cdot r = 2 \cdot \log_{\frac{3}{2}} n$. However, this is the expected height of $T$, i.e., $E[h]$. Therefore, the expected work undertaken by the algorithm

$$E[h] \times \text{work done per level} = O(n \cdot \log n).$$
Indicator Variable Analysis
Definition

A random variable is an indicator variable, if it assumes the value 1, for the occurrence of some event,
### Definition

A random variable is an indicator variable, if it assumes the value 1, for the occurrence of some event, and 0 otherwise.
Definition

A random variable is an indicator variable, if it assumes the value 1, for the occurrence of some event, and 0 otherwise.

Note

We recall that the rank of an array element is its position in the sorted array.
Definition

A random variable is an indicator variable, if it assumes the value 1, for the occurrence of some event, and 0 otherwise.

Note

We recall that the rank of an array element is its position in the sorted array. Every element of \(A\) has a unique rank in the set \(\{1, 2, \ldots, n\}\).
Definition

A random variable is an indicator variable, if it assumes the value 1, for the occurrence of some event, and 0 otherwise.

Note

We recall that the rank of an array element is its position in the sorted array. Every element of \( A \) has a unique rank in the set \( \{1, 2, \ldots, n\} \).

Analysis

Let \( S(i) \) denote the element in \( A \), whose rank \( i \).
Definition

A random variable is an indicator variable, if it assumes the value 1, for the occurrence of some event, and 0 otherwise.

Note

We recall that the rank of an array element is its position in the sorted array. Every element of $A$ has a unique rank in the set $\{1,2,\ldots,n\}$.

Analysis

Let $S(i)$ denote the element in $A$, whose rank $i$. We wish to compute the number of comparisons between $A[i]$ and the other elements of $A$, for each $i = 1,2\ldots n$. 

Indicator Variable Analysis

Definition

A random variable is an indicator variable, if it assumes the value 1, for the occurrence of some event, and 0 otherwise.

Note

We recall that the rank of an array element is its position in the sorted array. Every element of $A$ has a unique rank in the set $\{1, 2, \ldots, n\}$.

Analysis

Let $S(i)$ denote the element in $A$, whose rank $i$. We wish to compute the number of comparisons between $A[i]$ and the other elements of $A$, for each $i = 1, 2 \ldots n$. Instead, we will compute the number of comparisons between $S(i)$ and the elements of other ranks, for each $i = 1, 2, \ldots, n$. 
## Indicator Variable Analysis

### Definition

A random variable is an indicator variable, if it assumes the value 1, for the occurrence of some event, and 0 otherwise.

### Note

*We recall that the rank of an array element is its position in the sorted array. Every element of $A$ has a unique rank in the set $\{1, 2, \ldots, n\}$.*

### Analysis

Let $S(i)$ denote the element in $A$, whose rank $i$. We wish to compute the number of comparisons between $A[i]$ and the other elements of $A$, for each $i = 1, 2 \ldots n$. Instead, we will compute the number of comparisons between $S(i)$ and the elements of other ranks, for each $i = 1, 2, \ldots, n$. Are the two computations equivalent?
Indicator Variable Analysis (contd.)
Let $X_{ij}$ denote an indicator random variable, defined as follows:

$$X_{ij} = \begin{cases} 
1, & \text{if } S(i) \text{ and } S(j) \text{ are compared during the course of the algorithm} \\
0, & \text{otherwise} 
\end{cases}$$

Let $X$ denote the total number of comparisons made by the algorithm.
Let $X_{ij}$ denote an indicator random variable, defined as follows:

$$X_{ij} = \begin{cases} 
1, & \text{if } S(i) \text{ and } S(j) \text{ are compared during the course of the algorithm} \\
0, & \text{otherwise}
\end{cases}$$

Let $X$ denote the total number of comparisons made by the algorithm. Clearly,

$$X = \frac{n(n-1)}{2}$$
Let $X_{ij}$ denote an indicator random variable, defined as follows:

\[
X_{ij} = \begin{cases} 
1, & \text{if } S(i) \text{ and } S(j) \text{ are compared during the course of the algorithm} \\
0, & \text{otherwise}
\end{cases}
\]

Let $X$ denote the total number of comparisons made by the algorithm. Clearly,

\[
X = \sum_{i=1}^{n-1} \sum_{j>i} X_{ij}
\]
Let $X_{ij}$ denote an indicator random variable, defined as follows:

$$X_{ij} = \begin{cases} 
1, & \text{if } S(i) \text{ and } S(j) \text{ are compared during the course of the algorithm} \\
0, & \text{otherwise}
\end{cases}$$

Let $X$ denote the total number of comparisons made by the algorithm. Clearly,

$$X = \sum_{i=1}^{n-1} \sum_{j>i} X_{ij}$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$


Let $X_{ij}$ denote an indicator random variable, defined as follows:

$$X_{ij} = \begin{cases} 
1, & \text{if } S(i) \text{ and } S(j) \text{ are compared during the course of the algorithm} \\
0, & \text{otherwise}
\end{cases}$$

Let $X$ denote the total number of comparisons made by the algorithm. Clearly,

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

How to compute $X$?
Let $X_{ij}$ denote an indicator random variable, defined as follows:

$$X_{ij} = \begin{cases} 1, & \text{if } S(i) \text{ and } S(j) \text{ are compared during the course of the algorithm} \\ 0, & \text{otherwise} \end{cases}$$

Let $X$ denote the total number of comparisons made by the algorithm. Clearly,

$$X = \sum_{i=1}^{n-1} \sum_{j>i} X_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

How to compute $X$? We are not interested in $X$, but in $E[X]$!
Let $X_{ij}$ denote an indicator random variable, defined as follows:

$$X_{ij} = \begin{cases} 
1, & \text{if } S(i) \text{ and } S(j) \text{ are compared during the course of the algorithm} \\
0, & \text{otherwise}
\end{cases}$$

Let $X$ denote the total number of comparisons made by the algorithm. Clearly,

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

How to compute $X$? We are not interested in $X$, but in $E[X]$! Observe that,

$$E[X] = E\left[ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \right]$$
Indicator Variable Analysis (contd.)
Analysis (contd.)

\[ E[X] = \]
Indicator Variable Analysis (contd.)

Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]
The Randomized Quicksort Algorithm

Indicator Variable Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared.
The Randomized Quicksort Algorithm

Indicator Variable Analysis (contd.)

Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = \)
Indicator Variable Analysis (contd.)

Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = p_{ij} \).
Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = p_{ij} \). How to compute \( p_{ij} \)?
Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = p_{ij} \). How to compute \( p_{ij} \)? Let \( S_{ij} = \{S(i), S(i+1), \ldots S(j)\} \).
The Randomized Quicksort Algorithm

Indicator Variable Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = p_{ij} \). How to compute \( p_{ij} \)? Let \( S_{ij} = \{S(i), S(i+1), \ldots S(j)\} \). \( S(i) \) and \( S(j) \) will be compared only if, either one of them is picked before the other elements in \( S_{ij} \)!
The Randomized Quicksort Algorithm

Indicator Variable Analysis (contd.)

Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = p_{ij} \). How to compute \( p_{ij} \)? Let \( S_{ij} = \{S(i), S(i+1), \ldots S(j)\} \). \( S(i) \) and \( S(j) \) will be compared only if, either one of them is picked before the other elements in \( S_{ij} \)! Since all choices are made uniformly and at random, the probability of either \( S(i) \) or \( S(j) \) being picked before the other elements in \( S_{ij} \) is exactly...
The Randomized Quicksort Algorithm

Indicator Variable Analysis (contd.)

Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = p_{ij} \). How to compute \( p_{ij} \)? Let \( S_{ij} = \{ S(i), S(i+1), \ldots S(j) \} \). \( S(i) \) and \( S(j) \) will be compared only if, either one of them is picked before the other elements in \( S_{ij} \)! Since all choices are made uniformly and at random, the probability of either \( S(i) \) or \( S(j) \) being picked before the other elements in \( S_{ij} \) is exactly \( \frac{2}{j-i+1} \).
The Randomized Quicksort Algorithm

Indicator Variable Analysis (contd.)

Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = p_{ij} \). How to compute \( p_{ij} \)? Let \( S_{ij} = \{ S(i), S(i+1), \ldots, S(j) \} \). \( S(i) \) and \( S(j) \) will be compared only if, either one of them is picked before the other elements in \( S_{ij} \). Since all choices are made uniformly and at random, the probability of either \( S(i) \) or \( S(j) \) being picked before the other elements in \( S_{ij} \) is exactly \( \frac{2}{j-i+1} \). Therefore,
Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = p_{ij} \). How to compute \( p_{ij} \)? Let \( S_{ij} = \{ S(i), S(i+1), \ldots, S(j) \} \). \( S(i) \) and \( S(j) \) will be compared only if, either one of them is picked before the other elements in \( S_{ij} \)! Since all choices are made uniformly and at random, the probability of either \( S(i) \) or \( S(j) \) being picked before the other elements in \( S_{ij} \) is exactly \( \frac{2}{j-i+1} \).

Therefore,

\[
E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]
\]
Let $p_{ij}$ denote the probability that $S(i)$ and $S(j)$ are compared. Clearly, $E[X_{ij}] = p_{ij}$. How to compute $p_{ij}$? Let $S_{ij} = \{ S(i), S(i+1), \ldots S(j) \}$. $S(i)$ and $S(j)$ will be compared only if, either one of them is picked before the other elements in $S_{ij}$! Since all choices are made uniformly and at random, the probability of either $S(i)$ or $S(j)$ being picked before the other elements in $S_{ij}$ is exactly $\frac{2}{j-i+1}$.

Therefore,

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$
Indicator Variable Analysis (contd.)

Analysis (contd.)

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

Let \( p_{ij} \) denote the probability that \( S(i) \) and \( S(j) \) are compared. Clearly, \( E[X_{ij}] = p_{ij} \). How to compute \( p_{ij} \)? Let \( S_{ij} = \{ S(i), S(i+1), \ldots, S(j) \} \). \( S(i) \) and \( S(j) \) will be compared only if, either one of them is picked before the other elements in \( S_{ij} \)! Since all choices are made uniformly and at random, the probability of either \( S(i) \) or \( S(j) \) being picked before the other elements in \( S_{ij} \) is exactly \( \frac{2}{j-i+1} \). Therefore,

\[
E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]
= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}
= \sum_{i=1}^{n-1} \frac{n-i+1}{k} \frac{2}{k}
\]
Indicator Variable Analysis (contd.)
Indicator Variable Analysis (contd.)

Subramani Sample Analyses

Analysis (contd.)

\[
E[X] \leq \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k}
\]
Analysis (contd.)

\[ E[X] \leq \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} \]

\[ = 2 \cdot \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} \]
Analysis (contd.)

\[ E[X] \leq \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} \]

\[= 2 \cdot \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} \]

\[= 2 \cdot \sum_{i=1}^{n} H_n \]
Indicator Variable Analysis (contd.)

\[
E[X] \leq \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k}
\]

\[
= 2 \cdot \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k}
\]

\[
= 2 \cdot \sum_{i=1}^{n} H_n
\]

\[
= 2 \cdot n \cdot H_n
\]
Indicator Variable Analysis (contd.)

\[ E[X] \leq \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} \]

\[ = 2 \cdot \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} \]

\[ = 2 \cdot \sum_{i=1}^{n} H_n \]

\[ = 2 \cdot n \cdot H_n \]

\[ \in O(n \cdot \log n) \]