Randomized Algorithms

Md. Saidur Rahman,
Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology,
Dhaka, Bangladesh.

Randomized Algorithms

Randomized Algorithms

A *randomized algorithm* can be defined as one that receives, in addition to its input, a stream of random bits that it can use in the course of its action for the purpose of making random choices.

Randomized Algorithms

Randomized Algorithms

A *randomized algorithm* can be defined as one that receives, in addition to its input, a stream of random bits that it can use in the course of its action for the purpose of making random choices.

Randomized Algorithms

Randomized Algorithms

A *randomized algorithm* can be defined as one that receives, in addition to its input, a stream of random bits that it can use in the course of its action for the purpose of making random choices.

- may give different results when applied to the same input in different run

Randomized Algorithms

Randomized Algorithms

A *randomized algorithm* can be defined as one that receives, in addition to its input, a stream of random bits that it can use in the course of its action for the purpose of making random choices.

- may give different results when applied to the same input in different run
- time and space requirement is smaller than deterministic algorithms.

Randomized Algorithms

Randomized Algorithms

A *randomized algorithm* can be defined as one that receives, in addition to its input, a stream of random bits that it can use in the course of its action for the purpose of making random choices.

- may give different results when applied to the same input in different run
- time and space requirement is smaller than deterministic algorithms.
- extremely simple to implement

Randomized Algorithms

kth Largest Element Selection Problem

Given a set of n numbers S and a number k between 1 and n, select the kth largest element in S.

Randomized Algorithms

kth Largest Element Selection Problem

Given a set of n numbers S and a number k between 1 and n, select the kth largest element in S.

kth Largest Element Selection Problem

Given a set of n numbers S and a number k between 1 and n, select the kth largest element in S.

Idea

We choose an element $a_i \in S$ as the splitter and form sets $S^- = \{a_j : a_j < a_i\}$ and $S^+ = \{a_j : a_j > a_i\}$. We then can determine which of S^- or S^+ contains the largest element, and iterate only on this one.

Algorithm

Select(S, k)

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**
  The splitter a_i was in fact the desired answer

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**
  The splitter a_i was in fact the desired answer
else if $|S^-| \geq k - 1$ **then**

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**
  The splitter a_i was in fact the desired answer
else if $|S^-| \geq k - 1$ **then**
  The kth largest element lies in S^-

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**
  The splitter a_i was in fact the desired answer
else if $|S^-| \geq k - 1$ **then**
  The kth largest element lies in S^-
  Recursively call Select(S^-, k)

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**
  The splitter a_i was in fact the desired answer
else if $|S^-| \geq k - 1$ **then**
  The kth largest element lies in S^-
  Recursively call Select(S^-, k)
else

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**
  The splitter a_i was in fact the desired answer
else if $|S^-| \geq k - 1$ **then**
  The kth largest element lies in S^-
  Recursively call Select(S^-, k)
else
  Suppose $|S^-| = l < k - 1$

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**
  The splitter a_i was in fact the desired answer
else if $|S^-| \geq k - 1$ **then**
  The kth largest element lies in S^-
  Recursively call Select(S^-, k)
else
  Suppose $|S^-| = l < k - 1$
  The kth largest element lies in S^+

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**
  The splitter a_i was in fact the desired answer
else if $|S^-| \geq k - 1$ **then**
  The kth largest element lies in S^-
  Recursively call Select(S^-, k)
else
  Suppose $|S^-| = l < k - 1$
  The kth largest element lies in S^+
  recursively call Select($S^+, k - 1 - l$)

Algorithm

Select(S, k)
Choose a splitter $a_i \in S$
for each element a_j of S **do**
  put a_j in S^- if $a_j < a_i$
  put a_j in S^+ if $a_j > a_i$
end for
if $|S^-| = k - 1$ **then**
  The splitter a_i was in fact the desired answer
else if $|S^-| \geq k - 1$ **then**
  The kth largest element lies in S^-
  Recursively call Select(S^-, k)
else
  Suppose $|S^-| = l < k - 1$
  The kth largest element lies in S^+
  recursively call Select($S^+, k - 1 - l$)
end if

Correctness of Select(S, k)

Lemma

Regardless of how the splitter is chosen, the algorithm Select(S, k) returns the kth largest element.

Correctness of Select(S, k)

Lemma

Regardless of how the splitter is chosen, the algorithm Select(S, k) returns the kth largest element.

Correctness of Select(S, k)

Lemma

Regardless of how the splitter is chosen, the algorithm Select(S, k) returns the kth largest element.

Proof

The algorithm will terminate since it is always called recursively on a strictly smaller set.

Correctness of Select(S, k)

Lemma

Regardless of how the splitter is chosen, the algorithm Select(S, k) returns the kth largest element.

Proof

The algorithm will terminate since it is always called recursively on a strictly smaller set.

Correctness of Select(S, k)

Lemma

Regardless of how the splitter is chosen, the algorithm Select(S, k) returns the kth largest element.

Proof

The algorithm will terminate since it is always called recursively on a strictly smaller set. If $|S| = 1$ we must have $k = 1$. By induction it can be shown that correct answer will be returned when $|S| > 1$.

Time Complexity of Select(S, k)

Time Complexity of Select(S, k) depends on the way we choose the Splitter.

Time Complexity of Select(S, k)

Time Complexity of Select(S, k) depends on the way we choose the Splitter.

Time Complexity of Select(S, k)

Time Complexity of Select(S, k) depends on the way we choose the Splitter.

S^- and S^+ are approximately equal in size

$T(n) \leq T(n/2) + cn$

Time Complexity of Select(S, k)

Time Complexity of Select(S, k) depends on the way we choose the Splitter.

S^- and S^+ are approximately equal in size

$T(n) \leq T(n/2) + cn$

Time Complexity of Select(S, k)

Time Complexity of Select(S, k) depends on the way we choose the Splitter.

S^- and S^+ are approximately equal in size

$T(n) \leq T(n/2) + cn$
$T(n) = O(n)$

Time Complexity of Select(S, k)

Time Complexity of Select(S, k) depends on the way we choose the Splitter.

S^- and S^+ are approximately equal in size

$T(n) \leq T(n/2) + cn$
$T(n) = O(n)$

Choose the minimum element as the splitter

$$T(n) \leq cn + c(n-1) + c(n-2) + \cdots = \frac{cn(n+1)}{2} = \Theta(n^2)$$

Time Complexity of Select(S, k)

Well-Centered Splitter

If we had a way to choose splitter a_i such that there were at least ϵn elements both larger and smaller than a_i, for any fixed consant $\epsilon > 0$, then the size of the sets in the recursive call would shrink by a factor of at least $(1 - \epsilon)$ each time. Then

$$T(n) \leq T((1-\epsilon)n) + cn \leq \frac{1}{\epsilon}.cn$$

Randomized Select(S, k)

Random Splitter

Choose a splitter $a_i \in S$ uniformly at random

Analysis of Randomized Select(S, k)

Phase of the randomized algorithm

We will say that the algorithm is in *phase j* when the size of the set under consideration is at most $n(\frac{3}{4})^j$ but greater than $n(\frac{3}{4})^{j+1}$.

Analysis of Randomized Select(S, k)

Phase of the randomized algorithm

We will say that the algorithm is in *phase* j when the size of the set under consideration is at most $n(\frac{3}{4})^j$ but greater than $n(\frac{3}{4})^{j+1}$.

Analysis of Randomized Select(S, k)

Phase of the randomized algorithm

We will say that the algorithm is in *phase* j when the size of the set under consideration is at most $n(\frac{3}{4})^j$ but greater than $n(\frac{3}{4})^{j+1}$.

Central Element

An element of the set under consideration is *central* if at least a quarter of the elements are smaller than it and at least a quarter of the elements are larger than it.

Analysis of Randomized Select(S, k)

Observations

Analysis of Randomized Select(S, k)

Observations

- if a central element is choosen as a splitter, then at least a quarter of the set will be thrown away.

Analysis of Randomized Select(S, k)

Observations

- if a central element is choosen as a splitter, then at least a quarter of the set will be thrown away.

Analysis of Randomized Select(S, k)

Observations

- if a central element is choosen as a splitter, then at least a quarter of the set will be thrown away.
- the set will shrink by a factor of $\frac{3}{4}$ or better, and the current phase will come to an end.

Analysis of Randomized Select(S, k)

Observations

- if a central element is choosen as a splitter, then at least a quarter of the set will be thrown away.
- the set will shrink by a factor of $\frac{3}{4}$ or better, and the current phase will come to an end.

Analysis of Randomized Select(S, k)

Observations

- if a central element is choosen as a splitter, then at least a quarter of the set will be thrown away.
- the set will shrink by a factor of $\frac{3}{4}$ or better, and the current phase will come to an end.
- half of the elements of the set are central. So the probability of choosing a central element is $\frac{1}{2}$.

Analysis of Randomized Select(S, k)

Observations

- if a central element is choosen as a splitter, then at least a quarter of the set will be thrown away.
- the set will shrink by a factor of $\frac{3}{4}$ or better, and the current phase will come to an end.
- half of the elements of the set are central. So the probability of choosing a central element is $\frac{1}{2}$.

Analysis of Randomized Select(S, k)

Observations

- if a central element is choosen as a splitter, then at least a quarter of the set will be thrown away.
- the set will shrink by a factor of $\frac{3}{4}$ or better, and the current phase will come to an end.
- half of the elements of the set are central. So the probability of choosing a central element is $\frac{1}{2}$.
- expected number of iterations before a central element found is at most 2.

Analysis of Randomized Select(S, k)

Observations

- if a central element is choosen as a splitter, then at least a quarter of the set will be thrown away.
- the set will shrink by a factor of $\frac{3}{4}$ or better, and the current phase will come to an end.
- half of the elements of the set are central. So the probability of choosing a central element is $\frac{1}{2}$.
- expected number of iterations before a central element found is at most 2.

Analysis of Randomized Select(S, k)

Observations

- if a central element is choosen as a splitter, then at least a quarter of the set will be thrown away.
- the set will shrink by a factor of $\frac{3}{4}$ or better, and the current phase will come to an end.
- half of the elements of the set are central. So the probability of choosing a central element is $\frac{1}{2}$.
- expected number of iterations before a central element found is at most 2.
- expected number of iterations in phase j is at most 2.

Analysis of Randomized Select(S, k)

Let X be a random variable equal to the number of steps taken by the algorithm. Then

Analysis of Randomized Select(S, k)

Let X be a random variable equal to the number of steps taken by the algorithm. Then

$$X = X_0 + X_1 + X_2 + \cdots ,$$

where X_j is the expected number of steps spent in phase j.
When the algorithm is in phase j, the set has size at most $n(\frac{3}{4})^j$.

Analysis of Randomized Select(S, k)

Let X be a random variable equal to the number of steps taken by the algorithm. Then

$$X = X_0 + X_1 + X_2 + \cdots ,$$

where X_j is the expected number of steps spent in phase j. When the algorithm is in phase j, the set has size at most $n(\frac{3}{4})^j$. The number of steps required for one iteration in phase j is at most $cn(\frac{3}{4})^j$ for some constant c.

Analysis of Randomized Select(S, k)

Let X be a random variable equal to the number of steps taken by the algorithm. Then

$$X = X_0 + X_1 + X_2 + \cdots ,$$

where X_j is the expected number of steps spent in phase j.
When the algorithm is in phase j, the set has size at most $n(\frac{3}{4})^j$.
The number of steps required for one iteration in phase j is at most $cn(\frac{3}{4})^j$ for some constant c.
Expected number of iterations in phase j is at most 2. Thus $E[X_j] \leq 2cn(\frac{3}{4})^j$.

Analysis of Randomized Select(S, k)

Let X be a random variable equal to the number of steps taken by the algorithm. Then

$$X = X_0 + X_1 + X_2 + \cdots ,$$

where X_j is the expected number of steps spent in phase j.
When the algorithm is in phase j, the set has size at most $n(\frac{3}{4})^j$.
The number of steps required for one iteration in phase j is at most $cn(\frac{3}{4})^j$ for some constant c.
Expected number of iterations in phase j is at most 2. Thus $E[X_j] \leq 2cn(\frac{3}{4})^j$.
Thus total expected running time of the algorihm

$$E[X] = \sum_j e[X_j] \leq \sum_j 2cn(\frac{3}{4})^j = 2cn \sum_j (\frac{3}{4})^j \leq 8cn.$$