CS F364: Design & Analysis of Algorithms

Lecture-kt07: Median and Order Statistics

Dr. Kamlesh Tiwari,
Assistant Professor,
Department of Computer Science and Information Systems,
BITS Pilani, Rajasthan-333031 INDIA

Feb 04, 2017 (Campus @ BITS-Pilani Jan-May 2017)

Recap:

Lower bound of sorting. Any comparison sort algorithm requires $\Omega(n \log n)$ comparison in the worst case.

Recap:

Lower bound of sorting. Any comparison sort algorithm requires $\Omega(n \log n)$ comparison in the worst case.

- There are $n!$ permutations of n items. Each should be a leaf
- Binary tree of height h has at most 2^h leaves

Recap:

Lower bound of sorting. Any comparison sort algorithm requires $\Omega(n \log n)$ comparison in the worst case.

- There are $n!$ permutations of n items. Each should be a leaf
- Binary tree of height h has at most 2^h leaves

$$n! \leq 2^h \Rightarrow h \geq \log(n!) = \log((n/e)^n) = n\log(n) - n\log(e)$$

$$h = \Omega(n\log(n))$$

Recap:

Lower bound of sorting. Any comparison sort algorithm requires $\Omega(n \log n)$ comparison in the worst case.

- There are $n!$ permutations of n items. Each should be a leaf
- Binary tree of height h has at most 2^h leaves

$$n! \leq 2^h \Rightarrow h \geq \log(n!) = \log((n/e)^n) = n\log(n) - n\log(e)$$

$$h = \Omega(n \log(n))$$

It is still possible to sort in **liner time.** Three examples

1. Counting Sort
2. Radix Sort
3. Bucket Sort

A probability example

1. Consider an array of k items (assume with distinct items)
2. I am randomly picking an item as pivot

A probability example

1. Consider an array of k items (assume with distinct items)
2. I am randomly picking an item as pivot
3. And then I make two array say A1 and A2,

A probability example

1. Consider an array of k items (assume with distinct items)
2. I am randomly picking an item as pivot
3. And then I make two array say A1 and A2,
4. where all the items in A1 are less than or equal to pivot and

A probability example

1. Consider an array of k items (assume with distinct items)
2. I am randomly picking an item as pivot
3. And then I make two array say A1 and A2,
4. where all the items in A1 are less than or equal to pivot and
5. in A2 are the items are larger

A probability example

1. Consider an array of k items (assume with distinct items)
2. I am randomly picking an item as pivot
3. And then I make two array say A1 and A2,
4. where all the items in A1 are less than or equal to pivot and
5. in A2 are the items are larger

What is the probability of getting A1 or A2 of size i. $(1 \leq i \leq k)$

A probability example

1. Consider an array of k items (assume with distinct items)
2. I am randomly picking an item as pivot
3. And then I make two array say A1 and A2,
4. where all the items in A1 are less than or equal to pivot and
5. in A2 are the items are larger

What is the probability of getting A1 or A2 of size i. $(1 \leq i \leq k)$

Answer is :

$$1/k$$

Minimum and Maximum

- The i^{th} order statistic of a set of n elements is the i^{th} smallest element

Minimum and Maximum

- The i^{th} order statistic of a set of n elements is the i^{th} smallest element
- Minimum is the first order statistic and the maximum is the n^{th} order statistic

Minimum and Maximum

- The i^{th} order statistic of a set of n elements is the i^{th} smallest element
- Minimum is the first order statistic and the maximum is the n^{th} order statistic
- A median, is the “halfway point”, if n is odd, it is unique, otherwise there are two medians

Minimum and Maximum

- The i^{th} order statistic of a set of n elements is the i^{th} smallest element
- Minimum is the first order statistic and the maximum is the n^{th} order statistic
- A median, is the "halfway point", if n is odd, it is unique, otherwise there are two medians

We can solve the selection problem in $O(n \log(n))$ time, since we can sort the numbers

Minimum and Maximum

- The i^{th} order statistic of a set of n elements is the i^{th} smallest element
- Minimum is the first order statistic and the maximum is the n^{th} order statistic
- A median, is the "halfway point", if n is odd, it is unique, otherwise there are two medians

We can solve the selection problem in $O(n \log(n))$ time, since we can sort the numbers

1. But we are interested in better algorithms.

Minimum and Maximum

- The i^{th} order statistic of a set of n elements is the i^{th} smallest element
- Minimum is the first order statistic and the maximum is the n^{th} order statistic
- A median, is the "halfway point", if n is odd, it is unique, otherwise there are two medians

We can solve the selection problem in $O(n \log(n))$ time, since we can sort the numbers

1. But we are interested in better algorithms.
2. On what basis?

Selection in expected linear time

Algorithm 1: Minimum(A)

1 $min = A[1]$;

Selection in expected linear time

Algorithm 2: Minimum(A)

```
1 min = A[1];
2 for i = 2 to A.length do
```

Selection in expected linear time

Algorithm 3: Minimum(A)

```
1 min = A[1];
2 for i = 2 to A.length do
3 |   if min > A[i] then
4 |   |   min = A[i];
```

Selection in expected linear time

Algorithm 4: Minimum(A)

```
1 min = A[1];
2 for i = 2 to A.length do
3     if min > A[i] then
4         min = A[i];
5 return min
```

Selection in expected linear time

Algorithm 5: Minimum(A)

```
1 min = A[1];
2 for i = 2 to A.length do
3     if min > A[i] then
4         min = A[i];
5 return min
```

1. What is running time?

Selection in expected linear time

Algorithm 6: Minimum(A)

```
1 min = A[1];
2 for i = 2 to A.length do
3     if min > A[i] then
4         min = A[i];
5 return min
```

1. What is running time? $O(n)$
2. How to get maximum?

Selection in expected linear time

Algorithm 7: Minimum(A)

```
1 min = A[1];
2 for i = 2 to A.length do
3     if min > A[i] then
4         min = A[i];
5 return min
```

1. What is running time? $O(n)$
2. How to get maximum?
3. How to get Minimum as well as Maximum both at the same time?

Selection in expected linear time

Algorithm 8: Minimum(A)

```
1 min = A[1];
2 for i = 2 to A.length do
3     if min > A[i] then
4         min = A[i];
5 return min
```

1. What is running time? $O(n)$
2. How to get maximum?
3. How to get Minimum as well as Maximum both at the same time?
4. What about the second maximum item?

Selection in expected linear time

Algorithm 9: Minimum(A)

```
1 min = A[1];
2 for i = 2 to A.length do
3     if min > A[i] then
4         min = A[i];
5 return min
```

1. What is running time? $O(n)$
2. How to get maximum?
3. How to get Minimum as well as Maximum both at the same time?
4. What about the second maximum item?

How to find i^{th} item?

Selection in expected linear time

Algorithm 10: Randomized-Select(A, p, r, i)

```
1 if p = r then
2     return A[p]
```

Selection in expected linear time

Algorithm 11: Randomized-Select(A, p, r, i)

```
1 if p = r then
2     return A[p]
3 q = Randomized-Partition(A, p, r)
```

Selection in expected linear time

Algorithm 12: Randomized-Select(A, p, r, i)

```
1 if p = r then
2     return A[p]
3 q = Randomized-Partition(A, p, r)
4 k = q - p + 1
```

Selection in expected linear time

Algorithm 13: Randomized-Select(A, p, r, i)

```
1 if p = r then
2     return A[p]
3 q = Randomized-Partition(A, p, r)
4 k = q - p + 1
5 if i = k then
6     return A[p]
7 else if i < k then
8     return Randomized-Select(A, p, q-1, i)
```

Selection in expected linear time

Algorithm 14: Randomized-Select(A, p, r, i)

```
1  if p = r then
2      return A[p]
3  q = Randomized-Partition(A, p, r)
4  k = q - p + 1
5  if i = k then
6      return A[p]
7  else if i < k then
8      return Randomized-Select(A, p, q-1, i)
9  else
10     return Randomized-Select(A, q+1, r, i-k)
```

Selection in expected linear time

Algorithm 15: Randomized-Select(A, p, r, i)

```
1  if p = r then
2      return A[p]
3  q = Randomized-Partition(A, p, r)
4  k = q - p + 1
5  if i = k then
6      return A[p]
7  else if i < k then
8      return Randomized-Select(A, p, q-1, i)
9  else
10     return Randomized-Select(A, q+1, r, i-k)
```

Running time analysis.

- Take a indicator random variable X_k as $X_k = 1$ if sub array [p..q] has exactly k elements
- $E[X_k] = 1/k$

Analysis

Assume that the i^{th} element is always on the side of the partition with the greater number of elements.

$$T(n) \leq \sum_{k=1}^{n} X_k.(T(max(k-1, n-k)) + O(n))$$

Analysis

Assume that the i^{th} element is always on the side of the partition with the greater number of elements.

$$\begin{aligned} T(n) &\leq \sum_{k=1}^{n} X_k.(T(max(k-1, n-k)) + O(n)) \\ E[T(n)] &\leq \sum_{k=1}^{n} E[X_k].E[T(max(k-1, n-k))] + O(n) \end{aligned}$$

Analysis

Assume that the i^{th} element is always on the side of the partition with the greater number of elements.

$$
\begin{aligned}
T(n) &\leq \sum_{k=1}^{n} X_k.(T(max(k-1, n-k)) + O(n)) \\
E[T(n)] &\leq \sum_{k=1}^{n} E[X_k].E[T(max(k-1, n-k))] + O(n) \\
&\leq \sum_{k=1}^{n} \frac{1}{n}.E[T(max(k-1, n-k))] + O(n)
\end{aligned}
$$

Analysis

Assume that the i^{th} element is always on the side of the partition with the greater number of elements.

$$
\begin{aligned}
T(n) &\leq \sum_{k=1}^{n} X_k.(T(max(k-1, n-k)) + O(n)) \\
E[T(n)] &\leq \sum_{k=1}^{n} E[X_k].E[T(max(k-1, n-k))] + O(n) \\
&\leq \sum_{k=1}^{n} \frac{1}{n}.E[T(max(k-1, n-k))] + O(n) \\
&\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + O(n)
\end{aligned}
$$

Analysis

Assume that the i^{th} element is always on the side of the partition with the greater number of elements.

$$
\begin{aligned}
T(n) &\leq \sum_{k=1}^{n} X_k.(T(max(k-1, n-k)) + O(n)) \\
E[T(n)] &\leq \sum_{k=1}^{n} E[X_k].E[T(max(k-1, n-k))] + O(n) \\
&\leq \sum_{k=1}^{n} \frac{1}{n}.E[T(max(k-1, n-k))] + O(n) \\
&\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + O(n)
\end{aligned}
$$

We show $E[T(n)] = O(n)$

This means $E[T(n)] \leq cn$

Analysis

$$E[T(n)] \leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + O(n)$$

Analysis

$$
\begin{aligned}
E[T(n)] &\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + O(n) \\
&\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + an
\end{aligned}
$$

Analysis

$$
\begin{aligned}
E[T(n)] &\leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + O(n) \\
&\leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + an \\
&= \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k\right) + an
\end{aligned}
$$

Analysis

$$
\begin{aligned}
E[T(n)] &\leq \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} E[T(k)] + O(n) \\
&\leq \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} ck + an \\
&= \frac{2c}{n}\Big(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2\rfloor-1} k\Big) + an \\
&= \frac{2c}{n}\Big(\frac{n(n-1)}{2} - \frac{(\lfloor n/2\rfloor - 1)(\lfloor n/2\rfloor - 2)}{2}\Big) + an
\end{aligned}
$$

Analysis

$$
\begin{aligned}
E[T(n)] &\leq \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} E[T(k)] + O(n) \\
&\leq \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} ck + an \\
&= \frac{2c}{n}\Big(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2\rfloor-1} k\Big) + an \\
&= \frac{2c}{n}\Big(\frac{n(n-1)}{2} - \frac{(\lfloor n/2\rfloor - 1)(\lfloor n/2\rfloor - 2)}{2}\Big) + an \\
&\leq cn - (\frac{cn}{4} - \frac{c}{2} - an) \leq cn
\end{aligned}
$$

Analysis

$$
\begin{aligned}
E[T(n)] &\leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + O(n) \\
&\leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + an \\
&= \frac{2c}{n}\Big(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k\Big) + an \\
&= \frac{2c}{n}\Big(\frac{n(n-1)}{2} - \frac{(\lfloor n/2 \rfloor - 1)(\lfloor n/2 \rfloor - 2)}{2}\Big) + an \\
&\leq cn - (\frac{cn}{4} - \frac{c}{2} - an) \leq cn
\end{aligned}
$$

Choose $c > a/4$ then $n_0 = 2c/(c - 4a)$ then $\boxed{E[T(n)] = O(n)}$

Analysis

$$
\begin{aligned}
E[T(n)] &\leq \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} E[T(k)] + O(n) \\
&\leq \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} ck + an \\
&= \frac{2c}{n}\Big(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2\rfloor-1} k\Big) + an \\
&= \frac{2c}{n}\Big(\frac{n(n-1)}{2} - \frac{(\lfloor n/2\rfloor - 1)(\lfloor n/2\rfloor - 2)}{2}\Big) + an \\
&\leq cn - (\frac{cn}{4} - \frac{c}{2} - an) \leq cn
\end{aligned}
$$

Choose $c > a/4$ then $n_0 = 2c/(c-4a)$ then $\boxed{E[T(n)] = O(n)}$

What is worst case running time

Analysis

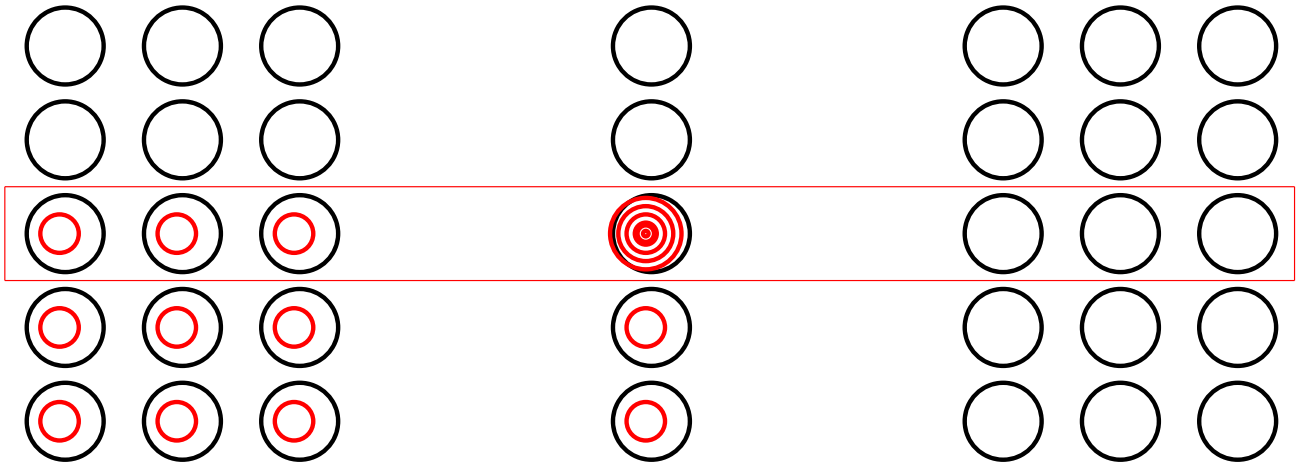
$$
\begin{aligned}
E[T(n)] &\leq \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} E[T(k)] + O(n) \\
&\leq \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} ck + an \\
&= \frac{2c}{n}\Big(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2\rfloor-1} k\Big) + an \\
&= \frac{2c}{n}\Big(\frac{n(n-1)}{2} - \frac{(\lfloor n/2\rfloor - 1)(\lfloor n/2\rfloor - 2)}{2}\Big) + an \\
&\leq cn - (\frac{cn}{4} - \frac{c}{2} - an) \leq cn
\end{aligned}
$$

Choose $c > a/4$ then $n_0 = 2c/(c-4a)$ then $\boxed{E[T(n)] = O(n)}$

What is worst case running time $\boxed{O(n^2)}$ ☹

Median of Medians

1. Divides the array in group of 5 items to find their median by sorting. Produce sub-array containing $n/5$ medians in $O(n)$ time.
2. Recursively apply same procedure to get Median-of-Medians
3. Time taken is $T(n) = T(n/5) + O(n)$ which solves to $T(n) = O(n)$.



1. Median-of-Medians is greater then at least $3n/10$ items.
2. Discard a fraction of items (that is at least $3n/10$) from the search space by spending $O(n)$ additional time.
3. Auxiliary array is used

Selection in expected linear time

Algorithm 16: momRankI(A, k)

```
1 j=Median-of-Medians(A)
2 if j = k then
3     return A[j]
4 if k < j then
5     return momRankI( A_<, k)
6 else
7     return momRankI( A_>, k-j)
```

$T(n) = T(7n/10) + O(n)$ that solves to $O(n)$.

Median of Medians takes $O(n)$ time even in worst case. ☺

Thank You!

Thank you very much for your attention!

Queries ?