Linear Time Selection

Selection is the problem of finding the $k$th smallest of a set of $n$ unsorted items, that is, the item that has exactly $k - 1$ items smaller than it.

Of particular interest is the median, defined to be the $\lceil n/2 \rceil$th smallest item.

We will see selection algorithms that use

- at most $4n + O(1)$ comparisons on average
- at most $16n + O(1)$ comparisons in the worst case
Average Linear Time Selection

The following algorithm for finding the \( k \)th smallest member of a set \( S \) of \( n \) distinct values is modelled on quicksort:

\[
\text{function select}(S, k) \\
1. \quad \text{if } S = \{x\} \text{ then return}(x) \text{ else} \\
2. \quad \text{Choose an element } a \text{ from } S \\
3. \quad \text{Let } S_1, S_2 \text{ be the elements of } S \\
\quad \text{ which are respectively } <, > a \\
4. \quad \text{Suppose } |S_1| = j \\
\quad \quad (a \text{ is the } (j+1)\text{st item}) \\
5. \quad \text{if } k = j + 1 \text{ then return}(a) \\
6. \quad \text{else if } k \leq j \text{ then} \\
\quad \quad \text{return}(\text{select}(S_1, k)) \\
7. \quad \text{else return}(\text{select}(S_2, k - j - 1))
\]
The Recurrence

Line 3 is the familiar pivoting operation from quicksort.

Let $T(n)$ be the worst case for all $k$ of the average number of comparisons used by procedure select on an array of $n$ numbers.

Clearly $T(1) = 0$.

Now, immediately after pivoting, $S_1$ contains $j$ values, and $S_2$ contains $n - j - 1$ values, for some $0 \leq j < n$.

Hence, the average number of comparisons for the single recursive call (either Line 6, or Line 7) is at most

$$\frac{1}{n} \sum_{j=0}^{n-1} (\text{either } T(j) \text{ or } T(n - j - 1))$$
But, when is it $T(j)$ and when is it $T(n-j-1)$?

The answer to this question is held in the two if-statements on Lines 5 and 6 of the algorithm.

When $k \leq j < n$ the algorithm recurses on $S_1$, requiring $T(j)$ comparisons.

When $k = j + 1$, the algorithm terminates.

When $k > j + 1$ (that is, $0 \leq j < k - 1$) the algorithm recurses on $S_2$, requiring $T(n-j-1)$ comparisons.

The average time for the recursive calls is therefore at most:

$$\frac{1}{n} \left( \sum_{j=0}^{k-2} T(n-j-1) + \sum_{j=k}^{n-1} T(j) \right).$$
Pivoting in Line 3 takes \( n - 1 \) comparisons, and hence for \( n \geq 2 \),

\[
T(n) \leq \frac{1}{n} \left( \sum_{j=0}^{k-2} T(n-j-1) + \sum_{j=k}^{n-1} T(j) \right) + n - 1
\]

\[
= \frac{1}{n} \left( \sum_{j=n-k+1}^{n-1} T(j) + \sum_{j=k}^{n-1} T(j) \right) + n - 1.
\]

What value of \( k \) maximizes the double sum?

Decrementing \( k \) has the effect of deleting a term from the left sum and inserting a term into the right sum (incrementing it has the opposite effect).

Hence, assuming that \( T(n) \) is a monotone increasing function (which implies that the term added is less than the term deleted), the maximum occurs when both sums are close to identical, that is, \( k = n - k + 1 \), or equivalently, \( k = (n + 1)/2 \).
Solving the Recurrence

Therefore,

\[ T(n) \leq \frac{2}{n} \sum_{j=\lfloor (n+1)/2 \rfloor}^{n-1} T(j) + n - 1. \]

We claim that \( T(n) \leq 4(n - 1) \).

The proof is by induction on \( n \).

The claim is true for \( n = 1 \).

Now suppose that \( n > 1 \) and \( T(j) \leq 4(j - 1) \) for all \( j < n \).
Then,

\[ T(n) \leq \frac{2}{n} \left( \sum_{j=\lfloor (n+1)/2 \rfloor}^{n-1} T(j) \right) + n - 1 \]

\[ \leq \frac{8}{n} \left( \sum_{j=\lfloor (n+1)/2 \rfloor}^{n-1} (j - 1) \right) + n - 1 \]

(by the induction hypothesis)

\[ = \frac{8}{n} \left( \sum_{j=\lfloor (n-1)/2 \rfloor}^{n-2} j \right) + n - 1 \]

\[ = \frac{8}{n} \left( \sum_{j=1}^{n-2} j - \sum_{j=1}^{\lfloor (n-3)/2 \rfloor} j \right) + n - 1 \]

\[ = \frac{8}{n} \left( \frac{(n-1)(n-2)}{2} - \frac{1}{2} \cdot \frac{n-3}{2} \cdot \frac{n-1}{2} \right) + n - 1 \]
There are two cases to consider, depending on whether $n$ is odd or even.

**Case 1.** $n$ is odd.

\[
\begin{align*}
T(n) &\leq \frac{8}{n} \left( \frac{(n - 1)(n - 2)}{2} - \frac{(n - 3)(n - 1)}{8} \right) + n - 1 \\
&= \frac{1}{n} (4n^2 - 12n + 8 - n^2 + 4n - 3) + n - 1 \\
&= 3n - 8 + \frac{5}{n} + n - 1 \\
&\leq 4n - 4 \quad (\text{since } n > 1).
\end{align*}
\]
Case 2. \( n \) is even.

\[
T(n) 
\leq \frac{8}{n} \left( \frac{(n-1)(n-2)}{2} - \frac{(n-4)(n-2)}{8} \right) + n - 1 
\]

\[
= \frac{1}{n} \left( 4n^2 - 12n + 8 - n^2 + 6n - 8 \right) + n - 1 
\]

\[
= 3n - 6 + n - 1 
\]

\[
= 4n - 7. 
\]

Therefore, for all \( n \geq 1 \), \( T(n) \leq 4n - 4 \).

The worst case number of comparisons used by this algorithm is identical to the worst case of quicksort, which is \( \Omega(n^2) \).
Worst Case Linear Time Selection

If we can find a pivot value that is guaranteed to be close to the median, that is, at worst the \( n/d \)th smallest or largest value for some \( d \geq 3 \), then the algorithm in the previous section would run in linear time, using an argument similar to the one that we used for quicksort.

If you can’t see this right now, the details are coming.

First, we need a useful result about the median.
Lemma

If $m$ is the median of $n$ values, then there are at least $n/2 - 1$ values larger than $m$, and at least $(n - 1)/2$ values smaller than $m$.

Proof. Let $L(n)$ be the number of values larger than, and $S(n)$ be the number of values smaller than the median of $n$ values.

The claim is that $L(n) \geq n/2 - 1$, and $S(n) \geq (n - 1)/2$.

Proof by induction on $n \geq 1$.

The claim is certainly true for $n = 1$ (since $S(1) = L(1) = 0$), and for $n = 2$ (since $S(2) = 1$ and $L(2) = 0$).
Now suppose that $n > 2$, and that $L(n - 2) \geq (n - 2)/2 - 1 = n/2 - 2$, $S(n - 2) \geq (n - 3)/2$. Then,

\[
L(n) = L(n - 2) + 1 \\
\geq (n/2 - 2) + 1 \\
= n/2 - 1,
\]

and

\[
S(n) = S(n - 2) + 1 \\
\geq (n - 3)/2 + 1 \\
= (n - 1)/2.
\]

**End of Proof.**
Diagrams

To help make things clearer, we will use the following diagramming technique. The numbers will be represented by dots, and a line will be drawn between two dots when they have been compared by the algorithm. The higher of two dots joined by a line will be larger than the lower one.
The Algorithm

If $n < 56$, we will simply sort the $n$ values using mergesort and return the $k$th one. Otherwise, suppose $n \geq 56$.

The algorithm will be broken into four steps. Suppose it uses $T(n)$ comparisons in the worst case.
Step 1

Break the $n$ values into $\lfloor n/7 \rfloor$ groups of 7 values, and at most one group that has the remaining $n \mod 7$ values, if any.

The latter group is a special one which we will call the overflow group.

Sort each of the $\lfloor n/7 \rfloor$ non-overflow groups independently. We will call each sorted group a chain.

It takes 14 comparisons to sort each chain using mergesort and hence $14 \lfloor n/7 \rfloor$ comparisons in all for Step 1.
Step 2

Find the median $x$ of the $\lfloor n/7 \rfloor$ medians of the chains found in Step 1. $T(\lfloor n/7 \rfloor)$ comparisons are required for Step 2.

The chains now fall into three categories:

- the *primary chain*, defined to be the chain containing $x$;
- *superior chains*, defined to be the ones whose median is greater than $x$; and
- *inferior chains*, defined to be the ones whose median is smaller than $x$.

Divide the $n$ values into six subsets, named $A$ through $F$. 
A: Members of the superior chains that are greater than or equal to the median of their respective chains.
B: Members of the superior chains that are smaller than the median of their respective chains.
C: Members of the primary chain that are greater than or equal to $x$.
D: Members of the primary chain that are less than $x$.
E: Members of the inferior chains that are greater than the median of their respective chains.
F: Members of the inferior chains that are smaller than or equal to the median of their respective chains.
What Do We Know?

- $C$ contains exactly 4 elements
- $D$ contains exactly 3 elements, and
- $A$ contains at least
  \[ 4(\lfloor n/7 \rfloor/2 - 1) = 2\lfloor n/7 \rfloor - 4 \]
  elements, (by the Lemma)
- $F$ contains at least
  \[ 4(\lfloor n/7 \rfloor - 1)/2 = 2\lfloor n/7 \rfloor - 2 \]
  elements (by the Lemma).

Therefore, we know at this stage that $x$ is less than or equal to at least $2\lfloor n/7 \rfloor$ elements in $A \cup C$, and at least $2\lfloor n/7 \rfloor$ elements in $D \cup F$. 
Step 3

Find the exact place of $x$ in the ordering by comparing it with the elements of $B \cup E$ and the overflow group.

This can be done with

- $2(\lfloor n/7 \rfloor - 1)$ comparisons for $B \cup E$ (since $B \cup E$ consists of $\lfloor n/7 \rfloor - 1$ sorted sequences of 3 values),
- at most 6 comparisons for the overflow group (since the overflow group contains at most 6 elements).

Therefore, the number of comparisons required for Step 3 is at most $2\lfloor n/7 \rfloor + 4$. 
Step 4

Suppose $x$ is the $\ell$th smallest value. If $k = \ell$, the algorithm should terminate and return $x$.

If $k < \ell$, then discard $A \cup C$ and recursively find the $k$th smallest value in $B \cup D \cup E \cup F$.

If $k > \ell$, then discard $D \cup F$. Suppose $D \cup F$ contains $m$ values. Recursively find the $(k - m)$th smallest value in $A \cup B \cup C \cup E$. 
Note that in either case, the recursive calls are applied to sets with at most \( n - 2\lfloor n/7 \rfloor \) elements in them. Since
\[
n - 2\lfloor n/7 \rfloor = (7\lfloor n/7 \rfloor + n \mod 7) - 2\lfloor n/7 \rfloor \leq 5\lfloor n/7 \rfloor + 6,
\]
this implies that at most \( T(5\lfloor n/7 \rfloor + 6) \) comparisons are required for this Step 4.
Recurrence Relation

For all $n \geq 56$,

$$T(n) = 14\lfloor n/7 \rfloor + T(\lfloor n/7 \rfloor)$$

Step 1 \hspace{1cm} Step 2

$$+ 2\lfloor n/7 \rfloor + 4 + T(5\lfloor n/7 \rfloor + 6)$$

Step 3 \hspace{1cm} Step 4

$$= T(\lfloor n/7 \rfloor) + T(5\lfloor n/7 \rfloor + 6)$$

$$+ 16\lfloor n/7 \rfloor + 4.$$
Analysis for \( n \leq 55 \)

We claim that for all \( n \geq 8 \), \( T(n) \leq 16n - 100 \).

Why 8?

\[
\begin{array}{c|ccccc}
 n & 1 & 2 & 3 & 4 & 5 \\
\hline
 T_M(n) & 0 & 1 & 3 & 5 & 8 \\
16n - 100 & -84 & -68 & -52 & -36 & -20 \\
\end{array}
\]

It can be easily verified that the claim is true for \( 8 \leq n < 56 \).

For example, \( T_M(55) = 55 \lceil \log 55 \rceil - 2 \lceil \log 55 \rceil + 1 = 55 \cdot 6 - 64 + 1 = 267 \), while \( 16 \cdot 55 - 100 = 880 - 100 = 780 \).
**Analysis for** $n \geq 56$

Now suppose that $n \geq 56$, and that for all $8 \leq m < n$, $T(m) \leq 16m - 100$.

Before we begin, we should verify that we can apply the induction hypothesis to the functions $T(\lfloor n/7 \rfloor)$ and $T(5\lfloor n/7 \rfloor + 6)$ from Steps 2 and 4, respectively.

Step 2 is obvious since $\lfloor n/7 \rfloor < n$ for all $n > 0$, and $\lfloor n/7 \rfloor \geq 8$ since $n \geq 56$.

For Step 4, $5\lfloor n/7 \rfloor + 6 < 5n/7 + 6$, which is less than $n$ when $n > 21$; and $5\lfloor n/7 \rfloor + 6 \geq 8$ when $5\lfloor n/7 \rfloor \geq 2$, which is true when $n \geq 7$.

Now we are ready to finish the induction.
\[ T(n) \leq T(\left\lfloor n/7 \right\rfloor) + T(5\left\lfloor n/7 \right\rfloor + 6) + 16\left\lfloor n/7 \right\rfloor + 4 \]
\[ \leq (16\left\lfloor n/7 \right\rfloor - 100) + (16(5\left\lfloor n/7 \right\rfloor + 6) - 100) + 16\left\lfloor n/7 \right\rfloor + 4 \text{ (by the ind. hyp.)} \]
\[ = 112\left\lfloor n/7 \right\rfloor - 100 \]
\[ \leq 16n - 100. \]

Therefore, by induction, \( T(n) \leq 16n - 100 \) for all \( n \geq 8 \).
Final Notes

The choice of 7 for the length of the chains was purely for convenience. Any odd number greater than 3 will suffice. The textbook uses 5.

The exact choice of this number will affect the constant multiple in the number of comparisons. Noticing that 7 values can actually be sorted using 13 comparisons (which is optimal) will reduce the number of comparisons to $15n + O(1)$.

Blum et al. (1973) proved an upper bound of $5.43n + O(1)$, which was improved to $3n + O(1)$ by Schönhage, Paterson, and Pippenger (1976), and more recently to $2.95n + O(1)$ by Dor and Zwick (1995).
A Lower Bound

It is obvious that at least $n - 1$ comparisons are needed to find the median of $n$ values in the worst case.

It’s possible to get a lower bound of almost $3n/2$ using a proof technique known as the adversary method.

We’ll prove a lower bound of $n + k - 2$ for finding the $k$th smallest or largest where $k \leq n/2$.

No loss of generality since finding the $k$th smallest out of $n$ is equivalent to negating all of the values and finding the $(n - k + 1)$th smallest.

Let $M(n, k)$ be the number of comparisons required to select the $k$th smallest of $n$ values.
The adversary argument: an adversary or demon attempts to construct a worst-case input by observing the algorithm’s behaviour and modifying the input on-the-fly in such a way that the algorithm’s prior actions are still valid, but lead it to make the maximum number of comparisons in the future.

For comparison based algorithms, this means changing the values in such a way that the result of prior comparisons are not violated.
The adversary observes the algorithm’s initial behaviour.

First, it may proceed by comparing disjoint pairs of values, but at some point it must construct an ordered triple.

It can do this in one of five ways by comparing either

(a) the maxima of two ordered pairs,

(b) the minima of two ordered pairs,

(c) the maximum of one ordered pair to the minimum of the other,

(d) the maximum of an ordered pair to a singleton, or

(e) the minimum of an ordered pair to a singleton.
The adversary changes the input by placing the maximum and minimum values in the positions shown in the picture.

In cases (a), (b), and (c), three comparisons have been wasted and the algorithm can do no better than to ignore the maximum and minimum and proceed as if it had only $n - 2$ values.

In case (d), two comparisons have been wasted and the algorithm can do no better than to ignore the maximum and proceed as if it had only $n - 1$ values.

In case (e), two comparisons have been wasted and the algorithm can do no better than to ignore the minimum and proceed as if it had only $n - 1$ values.
When $k = 1$, $M(n, k) = n - 1$. For $k > 1$ and $n > 2$, we have shown that the following holds. In Cases (a), (b), and (c),

$$M(n, k) \geq M(n - 2, k - 1) + 3. \quad (1)$$

In Case (d),

$$M(n, k) \geq M(n - 1, k) + 2. \quad (2)$$

In Case (e),

$$M(n, k) \geq M(n - 1, k - 1) + 2. \quad (3)$$
We claim that for all \( k \leq n/2 \), \( M(n, k) \geq n + k - 2 \). The proof is by induction on \( n \). The claim is true for \( n = 2 \), in which case \( k = 1 \) and \( M(2, 1) = 1 \). Now suppose that for all \( m < n \) and \( r \leq m/2 \), \( M(m, r) \geq m + r - 2 \). We will treat each of the three cases (1)–(3) separately.

In Case (1), since \( k \leq n/2 \) implies \( k - 1 \leq n/2 - 1 = (n - 2)/2 \),

\[
\begin{align*}
M(n, k) & \geq M(n - 2, k - 1) + 3 \\
& \geq (n - 2) + (k - 1) - 2 + 3 \quad \text{(by the ind. hyp.)} \\
& = n + k - 2.
\end{align*}
\]
In Case (2), if \( k \leq (n - 1)/2 \), then
\[
M(n, k) \\
\geq M(n - 1, k) + 2 \\
\geq (n - 1) + k \quad \text{(by the ind. hyp.)} \\
> n + k - 2.
\]

Otherwise, if \( k > (n - 1)/2 \), then
\[
(n - 1) - k < (n - 1)/2
\]
and hence,
\[
M(n, k) \\
\geq M(n - 1, k) + 2 \\
= M(n - 1, n - 1 - k) + 2 \\
\geq (n - 1) + (n - 1 - k) - 2 + 2 \quad \text{(by the ind. hyp.)} \\
= 2n - k - 2 \\
\geq n + k - 2 \quad \text{(since} \ n \geq 2k)
In Case (3), since $k \leq n/2$ implies $k - 1 \leq n/2 - 1 = (n - 2)/2 < (n - 1)/2$,

\[
M(n, k) \\
\geq M(n - 1, k - 1) + 2 \\
\geq (n - 1) + (k - 1) - 2 + 2 \quad \text{(by the ind. hyp.)} \\
= \ n + k - 2.
\]

Therefore, in all cases, $M(n, k) \geq n + k - 2$.

Therefore, for example, the number of comparisons needed to find the median is at least

\[
M(n, \lceil n/2 \rceil) = M(n, n - \lfloor n/2 \rfloor + 1) \\
\geq 2n - \lfloor n/2 \rfloor - 1 \\
= \ n + \lfloor n/2 \rfloor - 1 \geq 3n/2 - 2.
\]