CSC 226 - Assignment 1

Andrew Hobden (V00788452)

September 17, 2014

1 Searching, Lower Bounds

The selection of the k^{th} smallest element of a sequence of n elements can be accomplished with a lower bound of $\Omega(n)$. A paritioning strategy is taken by Quick Select and Linear Select. With the choice of a good *pivot* value, fractionally signifigant proportions of the sequence can be excluded from further inspection.

1.1 Arguing the Low Bounds

It may be considered that in order to find the k^{th} smallest element of an unsorted sequence of size n where k is actually the n^{th}largest element will always require n elements to be looked at, even in the simplest case of $max(sequence)$, requires iterating over each element in the list. By necessity, this is $\Omega(n)$.

In the case of $k < n$, any selection algorithm must still be $\Omega(n)$ since the element may be at any position in the sequence. Consider the seqeuence $[6, 4, 1, 2, 3]$, in order to find $k = 3$ the algorithm must view at least n items without prior knowledge of the data in order to be assured what it has selected is actually the k^{th} smallest element, otherwise it may select a inappropriate value.

2 Linear Selection

Linear selection utilizes a chunk size of c, where c is typically equal to $\{5, 7\}$. The running time of a linear select behaves like so:

$$T(n)_{Select} < T(\frac{n}{c})_{Pivot} + n_{Partition} + T(an)_{Recursion}$$

Where $a < 1$, and is resultant of the partitioning of n, therefore a subsequence of n. The algorithm relies on the fact that $T(\frac{n}{c}) + T(an) \leq 1 + T(n)$. If this is the case, the algorithm will progress like so:

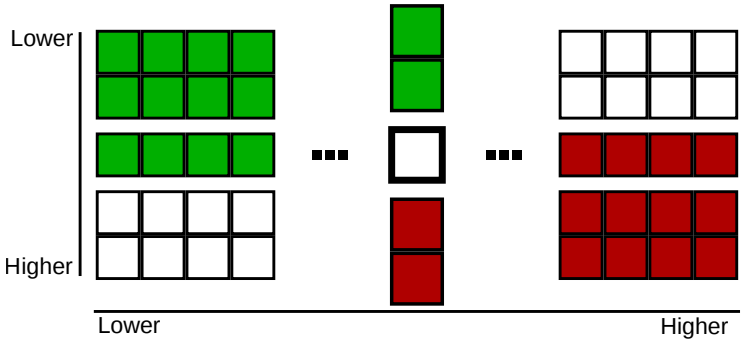
$$T(n)_{Select} = T(\frac{n}{c}) + n + (T(\frac{an}{c}) + an + T(a^2 n))$$

$$T(n)_{Select} = T(\frac{n}{c}) + n + (T(\frac{an}{c}) + an + (T(\frac{a^2n}{c}) + a^2n + T(a^3n))$$

2.1 In the case where $c = 5$

We can show that $a = \frac{7}{10}$. There are $\frac{3}{10}n$ values (white) not either:

- The chosen median (mid-center)
- Less than the chosen median (green)
- Greater than the chosen median (red)



Therefore each recursion operates on $a \approx \frac{7}{10}n$ elements. This results in:

$$T(n) \leq an + T(\frac{n}{5}) + T(\frac{7}{10}n)$$

Following from the slides in class it can be shown that:

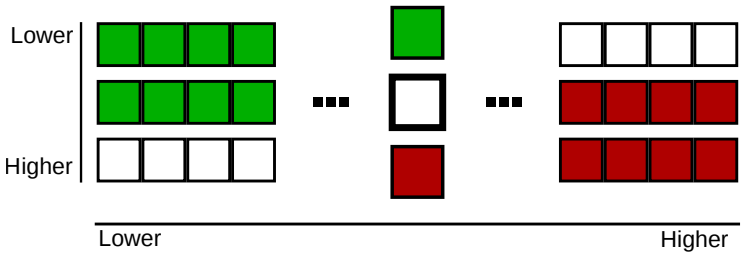
$$T(n) \leq an + \frac{9}{10}gn$$

$$an + \frac{9}{10}cn \leq cn$$

When $c \geq 10a$.

2.2 In the case where $c = 3$

In this case, $a \approx \frac{2}{3}$.



This results in:

$$T(n) \leq an + T(\frac{1}{3}n) + T(\frac{2}{3}n)$$

However the running time of $c = 3$ is not linear. Recall that $T(n) \in O(n)$ if $T(n) \leq gn$ for constant$g$.

$$T(n) \leq an + g\frac{1}{3}n + g\frac{2}{3}n$$

$$T(n) \leq an + gn$$

However observe that this implies:

$$an + gn \leq gn$$

This is problematic, as a would need to be a negative number, or zero, in order for this to hold true. Since $a \nleq 0$, Linear Select with $c = 3$ cannot be shown to behave linearly.

3 MST, Graph Properties

3.1 Cycle Property Correctness (Non-unique weights)

> For any cycle C in the graph, if the weight of an edge e of C is larger than the weights of all other edges of C, then this edge cannot belong to an MST.

Assume there exists a cycle C containing two edges, $\{a, b\}$, which have equal weight to each other, but are heavier than the other edges in the cycle.

$$a = b > c \geq d \geq e \geq ...$$

If both a and b were part of the MST, there would either be:

- A lighter weight edge which was rejected from the MST, which would mean that the resultant MST would not be of minimum wieght possible.
- A cycle, which cannot exist in an MST.

Therefore at either a or b will not be in the MST. The cycle property still holds, and a choice of which edge to remove must be made.

3.2 Cut Property Correctness (Non-unique weights)

> For any cut C in the graph, if the weight of an edge e of C is strictly smaller than the weights of all other edges of C, then this edge belongs to all MSTs of the graph.

Assume there exists a cut C with connecting edges, $\{a, b\}$, of equal weight, but are lighter than all other edges in the cut.

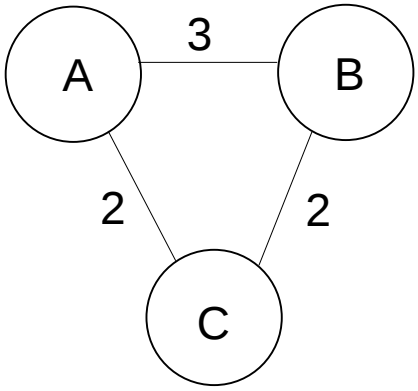
$$a = b < c \leq d \leq e \leq ...$$

Since only one edge is needed between the cut edges $c, d, e, ...$ may be discarded. Since the cut between the two components needs only one of these cuts, all possible MSTs must contain either a or b. If the function used to choose which edge to discard uses lexicographical order then the set of possible MSTs will all contain the chosen element. If the function is unpredictable in it's choice, then the set of possible MSTs contains all MSTs which contain one of the equally weighted edges.

4 MST Algorithm

4.1 Running into problems (non-unique weights)

Borůvka's algorithm encounters trouble when it attempts to select which edge to connect with node C. It may either choose node B using an edge weight of 2, or node A, also with an edge weight of 2. Since Borůvka's algorithm attempts to choose the lightest edge, it is unable to choose either, since they are of the same weight.



4.2 Overcoming the Difficulty

This hiccup can be resolved by simply choosing the next node in the trees lexicographic ordering. Meaning, in the above case, that the algorithm would choose to connect C with A over B. If there is no distinct ordering to the node or edge values, one can be chosen randomly, but this is not ideal.

```
1  Input: A connected graph G.
2  Initialize a forest T to be a set of one-vertex trees
3  While T has more than one component:
4    For each component C of T:
5      Begin with an empty set of edges S
6      For each vertex v in C:
7        Find the cheapest edge(s) from v to a vertex
             outside of C
8        If there is more than one cheapest edge:
9          Choose the first in lexicographic order
10       Add the edge to S
11     Add the cheapest edge in S to T
12 Output: T, the minimum spanning tree of G.
```