Eli Sobylak
Winter DS-Final

10-Mar-16

1.

Prove: A balanced tree forms if keys 1 to 2^{k+1} are inserted in order to an empty tree

BC: If $k=1$, there must be atleast one node in the tree ✓

RC: K works for 1, 2, ... k
Consider a tree with k+1 nodes. The root has a right subtree with K nodes on the right path, and a left subtree with at least k nodes on the right path.

IH: These subtrees yield a minimum of $2^k - 1$ nodes on each subtree. This plus the root gives us a perfect balanced tree with $2^{k+1} - 1$ nodes.

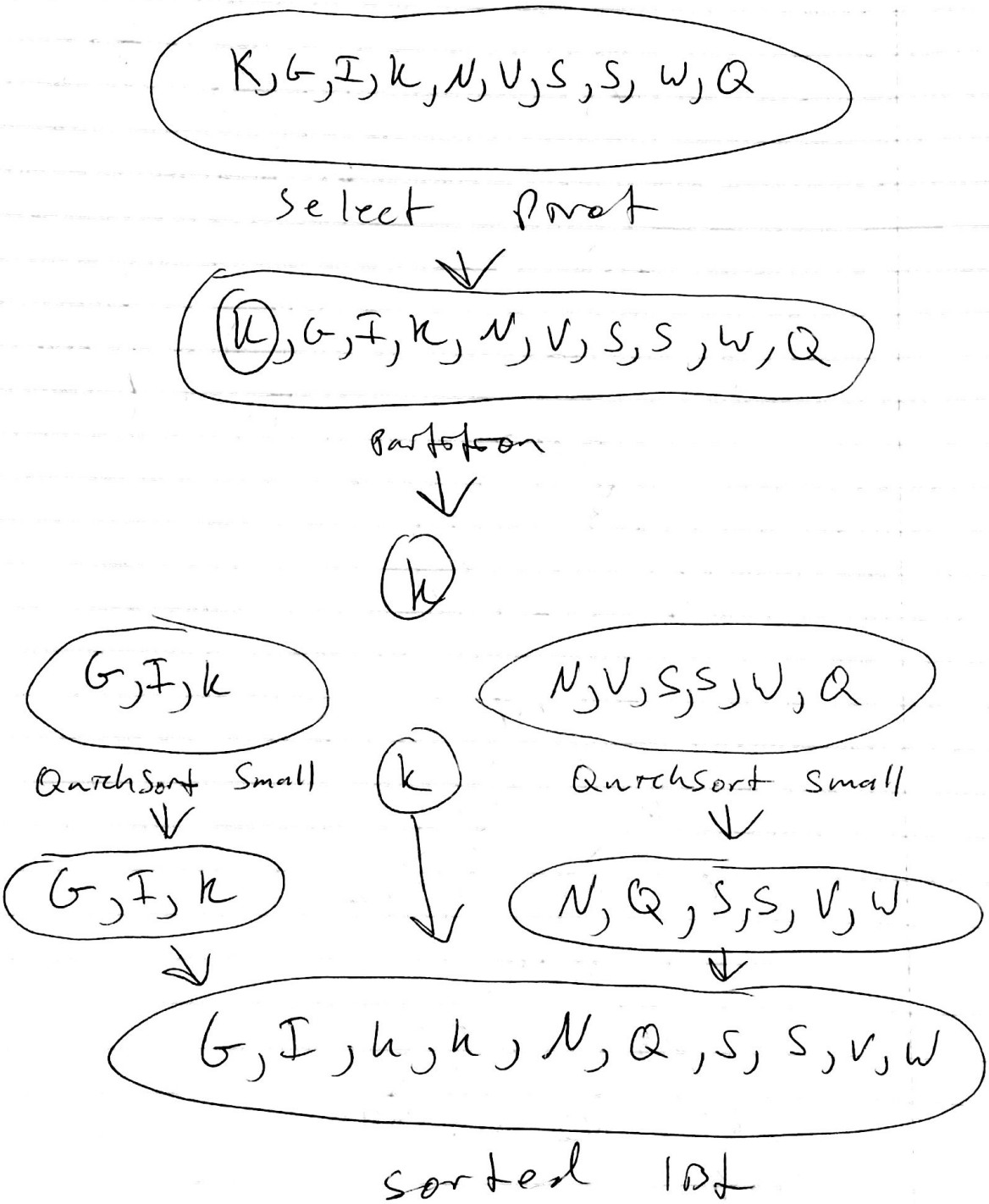
3.

a) - "Add" corresponding trees from the ~~two~~ forests
- For k from 0 to max height
→ If neither queue has a B_k OR (if H_1 has no trees AND the carry tree is NULL), SKIP
→ If only 1, leave it
→ If two, ~~after~~ attach the larger priority root as the child of the other, producing a tree of height k+1
→ If three, pick two to merge, leave 1

b)

c) It would have an impact on run time since we are adding more comparisons to the algorithm.

4.



5.

a)

	879	811	572	434	453	123	543	65	111	142	242	102
0	1	2	3	4	5	6	7	8	9	10	11	12

b) i, delete Max

Sorted

	811	453	572	434	242	123	547	65	111	142	102	879
0	1	2	3	4	5	6	7	8	9	10	11	12

ii, delete Max

Sorted

	572	453	547	434	242	123	102	65	111	142	879	811
0	1	2	3	4	5	6	7	8	9	10	11	12

iii, delete Max

Sorted

	547	453	142	434	242	123	102	65	111	879	811	572
0	1	2	3	4	5	6	7	8	9	10	11	12

iv, delete Max

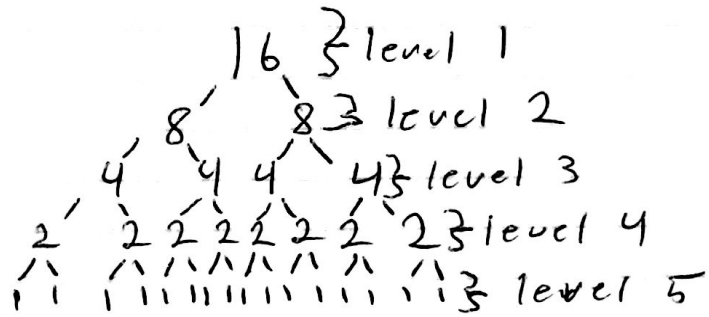
Sorted

	453	434	142	111	242	123	102	65	879	811	572	547
0	1	2	3	4	5	6	7	8	9	10	11	12

6.

a) To prove that any comparison based sorting algorithm to sort 4 elements requires 5 comparisons, we can use a decision tree to show this.

4 elements means 16 comparisons to start out with,



And we know that only one of the paths will be the solution to the sort. But they all take at least 5 comparisons

b) An example of this can be shown through selection sort.

The unsorted list: 4, 3, 2, 1

i - first iteration
1 | 4, 3, 2

ii - second iteration
1, 2 | 4, 3

iii - third iteration
1, 2, 3 | 4

iv - fourth iteration
1, 2, 3, 4 |

v - to check if list is in-order
1, 2, 3, 4 ✓

7.

a) $\sum_{i=2}^{N} i = \frac{n(n-1)}{2}$

I'd think that the running time for this version Quicksort would be $O(N^2)$ ~~especially~~

b) The run time on this instance would be $O(n \log n)$