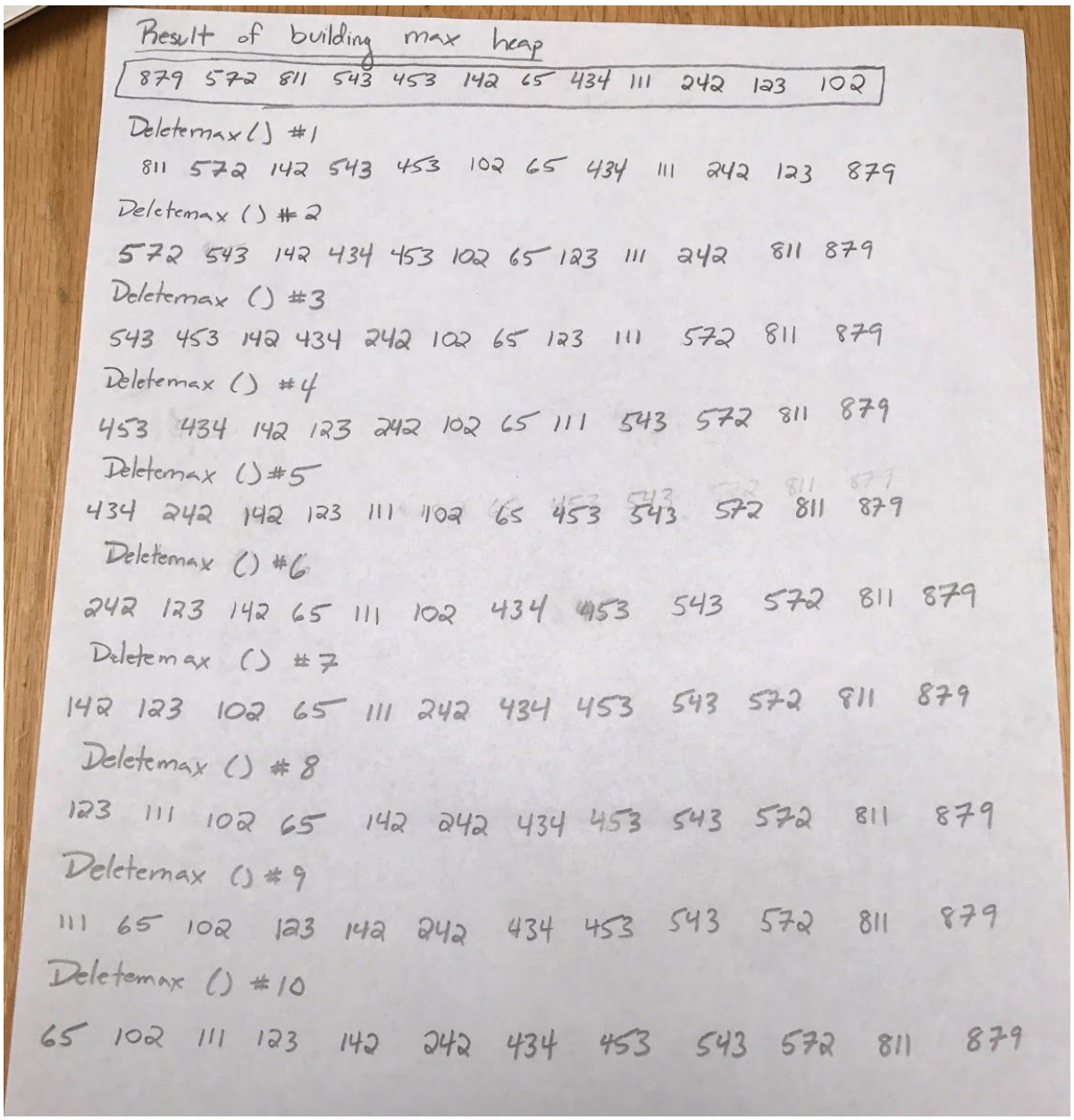
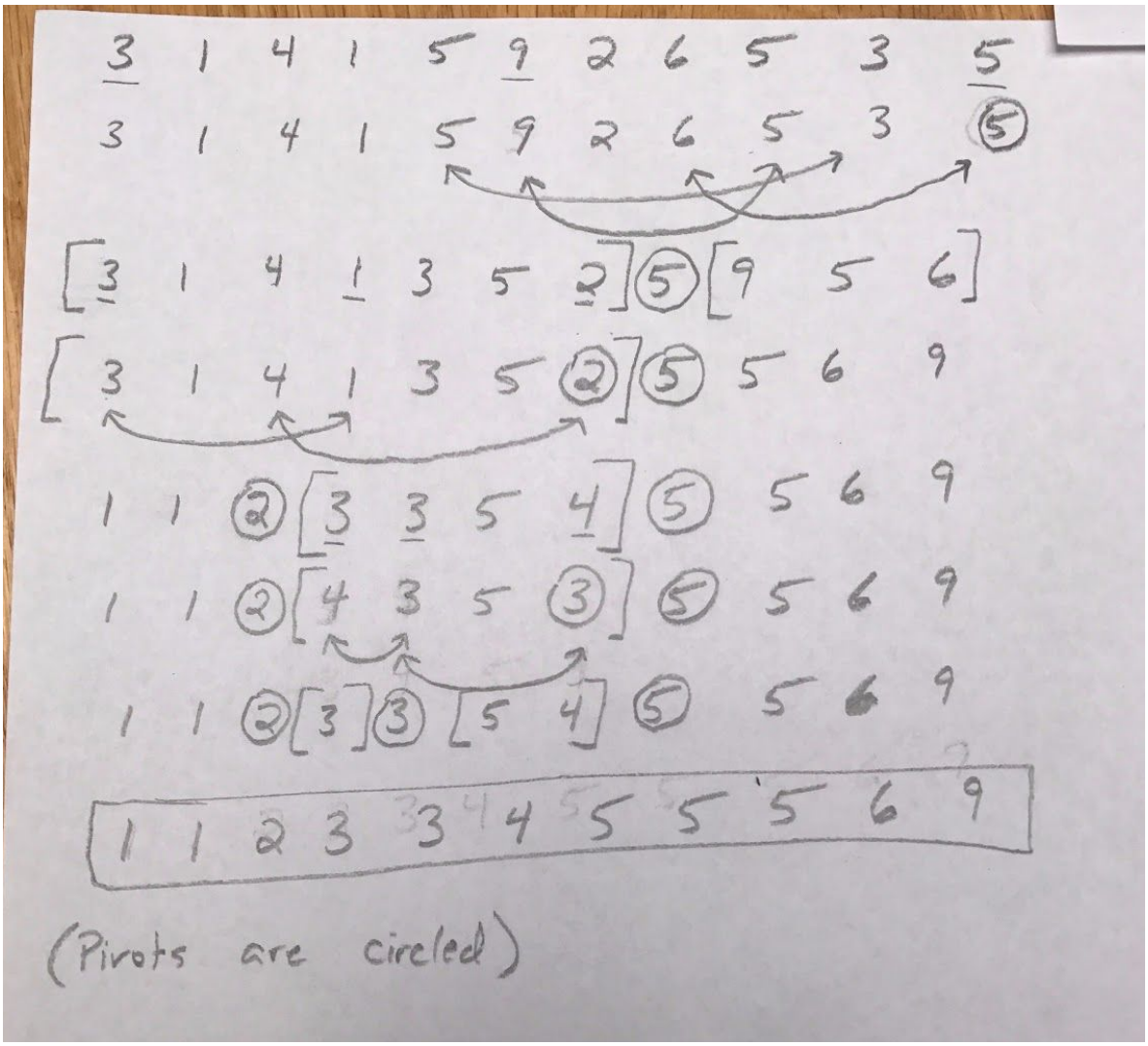
Joe Skimmons
jws2191
Written Homework 5

1.



2.


3 1 4 1 5 9 2 6 5 3 5
3 1 4 1 5 9 2 6 5 3 5
[3 1 4 1 3 5 2] 5 [9 5 6]
[3 1 4 1 3 5 2] 5 5 6 9
1 1 2 [3 3 5 4] 5 5 6 9
1 1 2 [4 3 5 3] 5 5 6 9
1 1 2 [3] 3 [5 4] 5 5 6 9
1 1 2 3 3 4 5 5 5 6 9
(Pivots are circled)

3. The worst case permutation of 10 elements for quicksort would occur if every time the pivot was chosen, one partition had only 1 element. This would lead to N-2 elements being sorted every time, resulting in a runtime of O(N^2) when sorting.
Ex. 20, 15, 25, 17, 10, 16, 22, 23, 24, 15

4. The total running time to sort an array of length N would be O(N). This is because, instead of having to run each branch of the merge sort (i.e sort each array after each split) in series, it will be done in parallel. For N elements, the bottom of the tree will have N processors each sorting and merging one element. But, as the processors merge the elements and move up the tree, each level has twice as long lists, with half as many processors. They move from N to N/2 to N/4 etc until at the top the runtime is close to 2N, which leads to a runtime of O(N).

5.1. A routine to determine if sticks A and B are above, below, or unrelated would be to find the projection of each stick (expressed as a function of x and y) on the XY plane, find the intersection of the projections, and then to compare the z coordinates at that intersection point on the XY plane. For example, if you have stick A's endpoints at (0,2,2) and (3,4,2) and B's endpoints at (1,2,3) and (2,3,4), then they have the XY projection points of (0,2) and (3,4) (for A) and (1,2) and (2,3) (for B). You could find the line connecting these two points, then solve for an intersection. If an intersection does not exist, then they are unrelated. If there is an intersection, then plug that point into the line equation for each stick and find which z coordinate is higher. If A has a higher z coordinate then it is above B, and vice versa.

5.2. First you would need to create a directed graph using each stick as a vertex. For each stick, find all of its intersections on the XY plane and evaluate if it is over or under that stick (as explained in part 1). Draw an edge from the above stick vertex to the below stick vertex and continue for all intersections and all sticks. When you have the graph, perform a topological sort on the graph and the resultant order will correspond to the order the sticks should be picked up. If the graph has a cycle, and is therefore not a DAG, then topological sort will fail and the sticks would not be able to be picked up.

6. If you have a graph with 3 nodes, (A, B, C), and the distance between each as follows:

A >> B = 4
A >> C = 1
B >> C = -4

A run through Dijkstra's algorithm would run as follows:

- Starting at Node A, it would go to each adjacent node and update their cost with the distance from A to that node:
  Cost of B: 4
- Cost of C: 1
- A is then marked as visited
- C is the smallest cost, that will be deleted from the priority queue and marked as visited.
- Because the graph is directed C does not lead to any other nodes, the algorithm will terminate.

The costs given in the first step will now be final:

A = 0
B = 4
C = 1

However, a shorter path to C exists if you were to travel from A to B to C (resulting in a cost of 0 due to the negative weight edge between B and C). Dijkstra's algorithm therefore results in the wrong shortest path.