CS 3851 Homework 1

3/23/20
Draven Schilling

1) a) [5,4,3,2,1]

[5] [4,3,2,1]

$[\underline{4}\ \underset{\rightarrow}{5}]\ [3,2,1]$

$[\underline{3}\ \underset{\rightarrow}{4}\ \underset{\rightarrow}{5}]\ [2,1]$

$[\underline{2}\ \underset{\rightarrow}{3}\ \underset{\rightarrow}{4}\ \underset{\rightarrow}{5}]\ [1]$

$[\underline{1}\ \underset{\rightarrow}{2}\ \underset{\rightarrow}{3}\ \underset{\rightarrow}{4}\ \underset{\rightarrow}{5}]$ ✓Sorted

b) [5,2,3,4,8,1,2]

[5] [2,3,4,8,1,2]

$[\underline{2}\ \underset{\rightarrow}{5}]\ [3,4,8,1,2]$

$[2\ \underline{3}\ \underset{\rightarrow}{5}]\ [4,8,1,2]$

$[2\ 3\ \underline{4}\ \underset{\rightarrow}{5}]\ [8,1,2]$

$[2\ 3\ 4\ 5\ \underline{8}]$

2) where $n > 0.396$, $8n^2+3n+8$ runs faster than $64n\log(n)+37n+6$. (used desmos to find intersection point)

3) $8n^2+3n+8$ is $O(n^2)$
$64n\log(n)+37n+6$ is $O(n\log(n))$

4 a) [1,2,3,4,5]

[1] [2,3,4,5]
[1,2] [3,4,5]
[1,2,3] [4,5]
[1,2,3,4] [5]
✓ [1,2,3,4,5]

[4,2,3,8,1]

[1] [2,3,8,4]
[1,2] [3,8,4]
[1,2,3] [8,4]
[1,2,3,4] [8]
✓ [1,2,3,4,8]

[5,4,3,2,1]

[1] [4,3,2,5]
[1,2] [3,4,5]
[1,2,3] [4,5]
[1,2,3,4] [5]
✓ [1,2,3,4,5]

b) Selection Sort Scans the entire list for the smallest number and swapps it with the First number in the list. It then scans the list again looking for the next smallest to swap with the second item in the list. This process is repeated for the entire length of the list. This makes selection sort always run $O(n^2)$ no matter how the Values of the unsorted list appear. Compared to insertion sort which can run either $O(n)$ or $O(n^2)$.

4 c) The loop invariant maintained by the for loop on line 2 maintains the sorted portion of smallest numbers in the list. It finds the smallest number in the remaining unsorted section of the list and swapps it with the

d) As the inner loop runs, it finds the smallest number and swapps it to the end of the sorted section. Using this approach, the numbers in the sorted section will always be in sorted order of the smallest numbers within the list. There is no initialization, and starting with the first itteration it finds the smallest number overall, so it begins sorting correctly.

e)

```
n = A.length                          c1
for j = 1 to n-1                      c2
    smallest = j                      c3
    for i in i+1 to n                 c4
        if A[i] < A[smallest]         c5
            smallest = i              c6
    tmp = A[j]                        c7
    A[j] = A[smallest]                c8
    A[smallest] = tmp                 c9
```

$$c_1 + c_2 + \sum_{j=1}^{n-1}\left[c_3 + c_4 + \sum_{i=i+1}^{n}(c_5 + c_6) + c_7 + c_8 + c_9\right]$$

f) $O(n^2)$

g) the selection sort will Always execute in $O(n^2)$ reguardless of how the initial list is setup. This is in contrast to insertion sort which runs $O(n)$ when the initial list is pre-sorted and $O(n^2)$ when its reverse sorted.

5 a) [1, 2, 3, 4, 5]
* compares each number pair, but Since they are already Sorted, no swappings are nessicary for each 5 passes through the list

[4, 2, 3, 8, 1]
[4, 2, 3, 1, 8]
[4, 2, 1, 3, 8]
[4, 1, 2, 3, 8]
[1, 4, 2, 3, 8]
[1, 2, 4, 3, 8]
[1, 2, 3, 4, 8]

[5, 4, 3, 2, 1]
[5, 4, 3, 1, 2]
[5, 4, 1, 3, 2]
[5, 1, 4, 3, 2]
[1, 5, 4, 3, 2]
[1, 5, 4, 2, 3]
[1, 5, 2, 4, 3]
[1, 2, 5, 4, 3]

[1, 2, 5, 3, 4]
[1, 2, 3, 5, 4]
[1, 2, 3, 4, 5]

b) Bubble sort passes through an N-length list N times where each pass through it compares each pair of adjsent numbers (starting right to left) and swaps them if the right number is less than the left number.

c) the loop invariant in line 2 will propogate the smallest number in the remaining portion of the list down to the (greatest) position in the sorted 'Front' part of the list. This will always happen because once the smallest number has been found it will always be shifted left when its compared with its neighbor until it reaches the sorted section.

d) to prove the loop invariant, We start with the initialize condition which holds true because starting with no items sorted, bubble sort will always propogate the smallest item (right to left) all the way down to the First sorted position because its less than every other item. for maintnance, the next smallest items will then be propogated down as they are then less than the remaining items to the position right of the previous smallest item. Finally, bubble sort terminates after N-1 iterations

5e)

```
for i = 1 to A.length-1
    for j = A.length downto i+1
        if A[j] < A[j-1]
            tmp = A[j]
            A[j] = A[j-1]
            A[j-1] = tmp
```

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{bmatrix} \quad \sum_{i=1}^{N-1}\left[C_1 + \sum_{j=N}^{i+1}(C_2+C_3+C_4+C_5+C_6)\right] =$$

$$\sum_{i=1}^{N-1}\left[C_1+(C_2+C_3+C_4+C_5+C_6)((i+1)-N)\right]$$

f) $O(n^2)$

g) Like Selection sort, Bubble sort will ALWAYS run in $O(n^2)$. reguardless if the list is pre-sorted and/or reverse sorted.