CPSC 331 Assignment 1

Samer Swedan

30098990

Q1

September 30, 2023 6:32 PM

2-2

n=5

1	2	3	4	5
5	9	10	4	6

```
BUBBLESORT(A, n)
1  for i = 1 to n − 1
2      for j = n downto i + 1
3          if A[j] < A[j − 1]
4              exchange A[j] with A[j − 1]
```

a) We must also show that the array A' is a permutation of A meaning that A' contains all the same elements as A but in sorted order.

Because no modifications are made to A other than swapping elements we can prove that A' contains all the same elements of A.

b) Loop invariant: Before each iteration of the for-j loop on lines 2-4 A[j] is the smallest element of the subarray A[j...n].

Initialization: Initially j=n and A[n...n] = A[n], therefore A[n] is a single element and thus it is also the smallest element so the invariant is vacuously true.

Maintenance: Let j=K. Therefore A[K] is the smallest element of A[K...n]. Now if A[K] < A[K-1] then A[K] and A[K-1] are swapped on line 4, thus making A[K-1] the smaller element in the subarray. If no swap occurs then A[K-1] was already smaller than A[K] and j is then decrementing to preserve the loop invariant.

Termination: Termination occurs when j=i and A[i] is the smallest element of A[i...n].

c)

Loop Invariant: At the start of every iteration of the for-i loop of lines 1-4 the subarray $A[1 \ldots i-1]$ consists of the smallest elements of $A[1 \ldots n]$ and in sorted order.

Initialization: Before the initial iteration $i=1$ and thus $A[1 \ldots i-1]$ is empty therefore the loop invariant is vacuously true.

Maintenance: From the loop invariant from part b) we know that after the execution of the inner loop $A[i]$ is the smallest element of subarray $A[i \ldots n]$. Thus, at the start of the after for loop $A[1 \ldots i-1]$ contains elements in sorted order that are less than the elements of $A[i \ldots n]$. Therefore, once execution of the after loop is complete the subarray $A[1 \ldots i]$ contains elements that are smaller than the elements contained in $A[i+1 \ldots n]$ and in sorted order.

Termination: The after for-i loop terminates when $i=n$. Thus, $A[1 \ldots n]$ will contain all elements of A and in sorted order.

d)

The outer for-i loop will run n times and will cause the inner for-j loop to run $n-i$ times, thus resulting in a worst-case running time of $\Theta(n^2)$.

Insertion sort also has a worst case running time of $\Theta(n^2)$ which is the same as bubble sort however the best case running time of insertion sort is $\Theta(n)$ which is better than bubble sort's best-case running time of $\Theta(n^2)$.

Q2

October 3, 2023 7:05 PM

a)

Greatest order of growth

$2^{2^{n+1}}$

2^{2^n}

$(n+1)!$

$n!$

e^n

$n \cdot 2^n$

$(\lg n)^{\lg n}$ and $n^{\lg \lg n}$

$(\lg n)!$

n^3

n^2 and $4^{\lg n}$

$n \lg n$ and $\lg(n!)$

$2^{\lg n}$ and n

$(\sqrt{2})^{\lg n}$

$\lg^2(n)$

$\ln(n)$

$\sqrt{\lg(n)}$

$\ln(\ln(n))$

$2^{\lg^* n}$

$\lg^* n$ and $\lg^*(\lg n)$

$\lg(\lg^* n)$

least order of growth

$n^{1/\lg n}$ and 1

Q3

October 3, 2023 7:45 PM

If $T(n)$ is defined by a standard recurrence, with parameters $a \geq 1$, $b > 1$, and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d \log n) & \text{if } a = b^d \quad \text{[Case 1]} \\ O(n^d) & \text{if } a < b^d \quad \text{[Case 2]} \\ O(n^{\log_b a}) & \text{if } a > b^d \quad \text{[Case 3]} \end{cases}$$

a) $T(n) = 2T(n/4) + \sqrt{n}$

$a = 2 \quad b = 4 \quad d = \frac{1}{2}$

$b^d = 4^{\frac{1}{2}} = 2$

$2 = 2$

Therefore case 1 because $a = b^d$

$T(n) = O(\sqrt{n} \log n)$

b)

Conjecture:

We will guess that $T(n) = O(\sqrt{n} \lg n)$

Meaning that there exists $n_0 > 0$ and a constant $c > 0$ such that for all $n \geq n_0$:

$T(n) \leq c \cdot \sqrt{n} \lg n$

Basis:

We know that $T(n)$ is constant when $n \leq 2$, therefore the base case trivially holds and we can choose $n_0 = 2$.

Inductive hypothesis : Suppose for every $m < n$:

$$T(m) \leq c\sqrt{m}\lg m$$

Inductive step:

$T(n) = 2T(n/4) + \sqrt{n}$ by def. of $T(n)$

$T(n) \leq 2(c\sqrt{n/4}\lg(n/4)) + \sqrt{n}$ by inductive hypothesis

$T(n) \leq (c\frac{\sqrt{n}}{2}) \cdot (2\lg n - 2) + \sqrt{n}$

$T(n) \leq (c\sqrt{n}) \cdot (\lg n - 1) + \sqrt{n}$

$T(n) \leq (c\sqrt{n}) \cdot \lg n - c\sqrt{n} + \sqrt{n}$

$T(n) \leq c\sqrt{n}\lg n$ if we choose $c \geq 1$

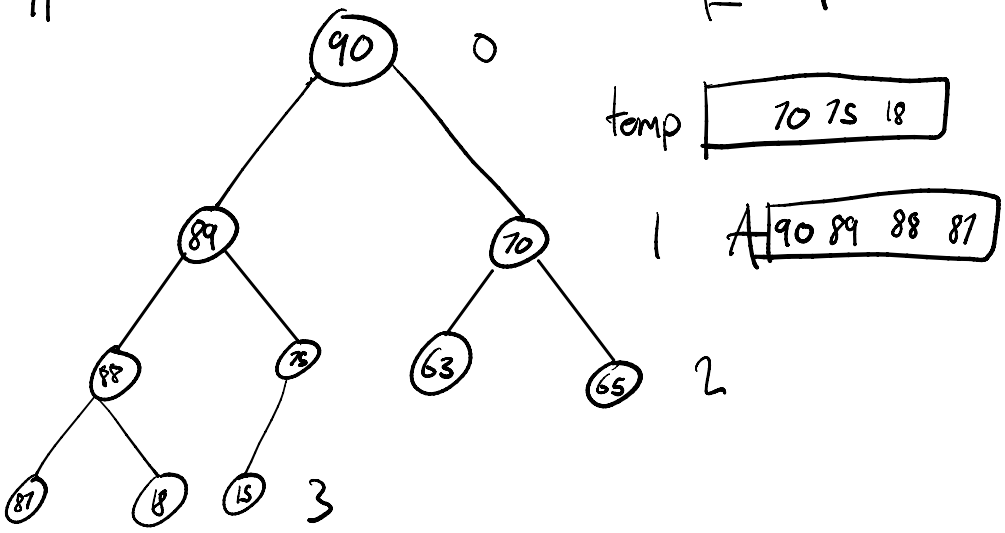
Conclusion:

- we chose n_0 in the basis
- The induction step works if $c \geq 1$
- So we pick $n_0 = 2$ and $c = 1$

Q5

October 6, 2023 1:13 AM

#



Thought process:

1. Create a list called tmp
2. Start from i=0
3. Append i to tmp
4. iterate through temp and find the largest key
5. Pop off the largest key and append it to A
6. Append the children of the popped key in the previous step to tmp
7. Repeat from step 4. until there are K elements in A

```
kthLargestElements(H, K):
1.  tmp = new list
2.  largestIndex = 0
3.  A = array of size K

4.  tmp.add(largestIndex)

5.  for(i in range(K))
6.      largestIndex = max(tmp, index with greatest value)
7.      largestValue = H[tmp.pop(largestIndex)]
8.      A[i] = largestValue
9.      left = leftChild(largestIndex)
10.     right = rightChild(largestIndex)
11.     if(left != null):
12.         tmp.add(left)
13.     if(right != null):
14.         tmp.add(right)
15.  return A
```

Proof of correctness:

Loop invariant: At the beginning of each iteration of the loop A contains the i largest node values and tmp contains the indices of the i greatest nodes in H

Initialization: Before the very first iteration of the loop i=0 and A is empty. tmp contains the index of the root and the root of a max-heap is always the largest value. Therefore the loop invariant holds.

Maintenance: During every iteration of the loop the node containing the greatest value is extracted from its index in tmp and is the value is added to A. Then the indices of the children of the extracted node are added to tmp if they are not null. Therefore, ensuring that A contains the i greatest values in H and tmp contains the indices of the i greatest nodes.

Termination: the loop terminates when $i = K$, at this point A contains the K greatest values of H and tmp contains the indices of the K values.

Proof of termination:

The for loop iterates from 0 to K and no modifications are made to i other than incrementation by the loop. No modifications are made to K or H, therefore termination of the loop occurs when $i = K$.