Name: Sabyasachi Sahoo

COSC 40403 - Analysis of Algorithms: Fall 2018: Homework 1

Due: 23:59:59 on September 4, 2018

Question	Points	Score
1	10	
2	10	
3	10	
4	15	
Total:	45	

1. (10 points) Consider the ***max-index problem***:
Input: A sequence of n numbers $A = \langle a_1, a_2, a_3, \ldots, a_n \rangle$.
Output: An index i such that $v = max(A[i])$. If v is in the sequence mutliple times, this algorithm returns the first occurance (first index) of v.

Write pseudocode for MAX-INDEX, which scans through the sequence, looking for v. Develop a cost-model equation, $T(n)$, for the running time for MAX-INDEX similar to what was developed in class and in our textbook. Express your solution in terms of Θ-notation. How many elements of the input sequence need to be checked on average, assuming that the element being searched for is equally likely to be any element in the array? How about in the worst-case? What are the average-case and worst-case running times of MAX-INDEX in Θ-notation. Justify your answers.

Solution:
Pseudocode:

-)max-index(A)		
-)max = 0	c1	1
-)for i = 0 to A(len)-1	c2	n+1
-) if A[i] > A[max]	c3	n
-) max = i	c4	n
-)return max	c5	1

Cost model Equation:
T(n) = c1 + c2(n+1) + c3(n) + c4(n) + c5
T(n) = Θ(n)
We would have to check all the elements all the time because it is equally likely to be any element in the array.
Hence, the average case and the worst case both have the same running times i.e., Θ(n).

2. (10 points) Consider the ***Searching problem***:
Input: A sequence of n numbers $A = \langle a_1, a_2, a_3, \ldots, a_n \rangle$ and a value v.
Output: An index i such that $v = A[i]$. If v is not in $A[i]$, then the algorithm will return -1..

Observe that if a sequence A is sorted, we can check the midpoint of the sequence against v and eliminate half of the sequence from further consideration. The BINARY-SEARCH algorithm repeats this procedure, halving the size of the remaining portion of the sequence each time. Write pseudocode, either iterative or recursive, for BINARY-SEARCH. Develop the cost-model equation for $T(n)$. Argue that the worst-case running time is $\Theta(\lg n)$.

Solution:
Pseudocode:

```
-)binary-Search(A, l , u , v)
-)if l ≤ u
-)      mid = l + (U -1)/2
-)      if arr[mid] == v
-)            return mid
-)      if arr[mid] > v
-)            return binary-Search(A, l, mid-1, v)
-)      return binary-Search(A, mid+1, u, v)
-)return -1
```

Cost Model Equation:
Divide : Array division in two space ranges gives us D(n) = $\Theta(1)$
Conquer : We recursively contract the problem size to half of the problem which gives us C(n) = T(n/2)
Now when we add the fuctions from these equations, we obtain the recurrence relation
T(n) = $\Theta(1)$ —— if n = 1
T(n) = T(n/2) + $\Theta(1)$ —— if n ≥ 1

Solving the recurrence relation:
replace n with 2^k , assuming n is power of 2
So now, T(2^k) = T(2^k / 2) + 1 = T(2^{k-1}) + 1
Replace T(b^k) with t_k and the equation becomes
$t_k = t_{k-1} + 1$, result in homogenous linear recurrence
Gives (x-1)(x-1) — root 1 of multiplicity 2
General solution is , $t_k = c_1 + c_2$k Now, replacing T(b^k) for t_k and n for b^k
We get, T(2^k) = $c_1 + c_2$k = $c_1 + c_2 \log_2 n$
Solving for c_1 , c_2, using T(1) = 1
We get, T(n) = $\log_2 n + 1$
Thus, T(n) = O(log$n$)

3. (10 points) Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in $A[1]$. Then find the second smallest element of A, and exchange it with element $A[2]$. Continue in this manner for the first $n-1$ elements of A. Write pseudocode for this algorithm, which is known as SELECTION-SORT. Develop a cost-model equation for the running time for SELECTION-SORT similar to what was developed in class and in our textbook. How many comparisons are made in the worst-case using SELECTION-SORT? How many data exchanges are made in the worst-case using SELECTION-SORT? Express your answers in Θ-notation.

Solution:

Selection-Sort(A)
n = length[A] ——- c1 ——- 1
for j =1 to n-1 ——- c2 ——- n
——- smallest = j ——- c3——- n-1
——- for i = j + 1 to n ——- c4 ——- [j=1]SUM[n-1] [n-j +1]
——- ——- if A[i] < A[smallest] ——- c4 ——- [j=1]SUM[n-1] [n-j]
——- ——- smallest = i ——- c6 ——- [j=1]SUM[n-1] [n-j]
——- Exchange A[j] - A[smallest] ——- c7 ——- n-1

Cost Model Equation:
T(n) = c1(n)+c2(n)+c3(n-1)+c4(((n$\hat{2}$ - n)/2) + n)+c5(((n$\hat{2}$ - n)/2))+c6(((n$\hat{2}$ - n)/2))+c7(n-1)
Worst case time complexity : Θ(n$\hat{2}$)
Worst case comparison: Θ(n$\hat{2}$)
Worst case data exchanges: Θ(n)

4. (15 points) Develop a Jupyter notebook and using the Python programming language, implement the insertion sort and merge sort algorithms as described in our textbook and notes. Using a random array of integers of at most 250, empirically determine what value of n (approximate) where merge sort is faster than Selection sort.

Solution: Ans (n = 40) attached in the zip folder.