Design & Analysis of Algorithms

Assignment-1

Submit By:- 09 Aug, 2019

NOTE: Please donot try to copy the answers. Your assignment will be checked for Plagiarism.

GROUP NUMBERS	QUESTIONS TO ATTEMPT
17AITCC1 (GROUP-A)	Q1,Q3,Q7,Q13
17AITCC1 (GROUP-B)	Q2,Q5,Q11,Q13
17AITCC2 (GROUP-A)	Q4,Q6,Q12,Q13
17AITCC2 (GROUP-B)	Q8,Q9,Q10,Q13

Q1. Suppose you are given the task to sort 100,000 social security numbers (each is a 9 digit number). You have decided to use radix sort for this problem and want to decide how many base-10 digits to use for each radix sort digit. Which is best

(a) 1 base-10 digit per radix sort digit, or
(b) 3 base-10 digits per radix sort digit, or
(c) one radix sort digit of 4 base-10 digits and the other radix sort digit of 5 base-10 digits

You are provided with a counting sort procedure with exact time complexity of 5n+4k where n is the number of elements being sorted and k is the number of possible values for the elements. Show your work.

Q2. Consider the following algorithm:

To compute approximately the square root of a positive real number a:
1. Set r to the mean of 1 and a.
2. While r^2 is not a good enough approximation to a, repeat:
2.1. Set r to the mean of r and a/r.
3. Terminate with answer r.

(a) Use this algorithm to calculate the square roots of the following numbers: 4, 6, 8, 9. In each case, calculate your answer to an accuracy of two decimal places, i.e., the absolute difference between a and r2 should be less than 0.01.
(b) Write a program to implement the algorithm and use it to check your answers to part (a) above.
(c) What would happen if step 2 of the algorithm were simply "While r^2 is not equal to a, repeat:"?

Q3. In the array implementation of stacks, an *overflow* occurs when the stack size is about to exceed the array length. Show how to deal with an overflow by substituting a longer array. Copy all the stack elements into a new and longer new array, and replace the existing array by that new array.

Q4. A keyboard driver is a process (generally provided by the operating system) that allows the user to enter characters at any speed, without waiting for the application program to use these characters.

(a) Outline how the keyboard driver can communicate with the application program via a queue.
(b) Write an algorithm for the keyboard driver. It should 'echo' to the screen each graphic character (visible character or space) entered by the user. It should simply ignore any control character.
(c) Suppose now that the DELETE character is to cancel the last graphic character. A sequence of DELETEs is to cancel the same number of graphic characters. What changes are needed to the keyboard driver, and to the communication between the keyboard driver and the application program?

Q5. When repeatedly searching an unsorted Single Linked List, it can be advantageous to move an item to the head of the list when it is found, if it is likely that the same item will be searched for again in the near future.

(a) Write this modified linear search algorithm.
(b) How does this modification affect the time efficiency of the algorithm when the same item is searched for 50 times out of the next 100 searches? (Hint: Consider the total time taken on average to perform all 100 searches with and without the modification.)
(c) Write an algorithm for the same

Q6. There are two algorithms called *Alg1* and *Alg2* for a problem of size n. *Alg1* runs in N^2 microseconds and *Alg2* runs in 100N(logN) microseconds. *Alg1* can be implemented using 4hours of programmer time and needs 2 minutes of CPU time. On the other hand, *Alg2* require 15 hours of programmer time and 6 minutes of CPU time. If programmers are paid 20 dollars per hour and CPU time costs 50 dollars per minute, how

many times must a problem instance of size 500 be solved using *Alg2* in order to justify its development cost.

Q7. Calculate the run times for the following for-loops:

for (int i = 0; i < n*n*n; ++i) { for (int j = 0; j < i; ++j) { sum += i + j; } }	
for (int i = 0; i < n; ++i) { for (int j = 0; j < i*i*i; ++j) { sum += i + j; } }	
for (int i = 0; i < n*n; ++i) { for (int j = 0; j < i*i; ++j) { sum += i + j; } }	
for (int i = 0; i < n*m; ++i) { for (int j = 0; j < i; ++j) { sum += i + j; } }	
for (int i = 0; i < n; ++i) { for (int j = 0; j < i*m; ++j) { sum += i + j; } }	

Q8. Suppose the numbers 0, 1, 2, …, 9 were pushed onto a stack in that order, but that pops occurred at random points between the various pushes. The following is a valid sequence in which the values in the stack could have been popped: 3, 2, 6, 5, 7, 4, 1, 0, 9, 8
Explain why it is not possible that 3, 2, 6, 4, 7, 5, 1, 0, 9, 8 is a valid sequence in which the values could have been popped off the stack.

Q9. The *nth* Fibonacci number is defined as the sum of the two previous Fibonacci numbers where the 0*th* and 1st Fibonacci numbers are defined as 1. What is the maximum size of the stack when this function is called with the argument 4:

```
int Fibonacci( int n ) {
 return (n <= 1) ? 1 : Fibonacci( n - 1 ) + Fibonacci( n - 2 );
}
```

What is the total number of function calls made, including the initial call with the argument 4?

Q10. Evaluate the following expressions that are written using reverse Polish notation:

(a) 1 2 3 + * 4 5 * 6 + +
(b) 1 2 3 + * 4 5 * + 6 +
(c) 1 2 + 3 * 4 5 * 6 + +

Q11. A queue is a first-in—first-out data structure. Suppose you have two queues accepting requests for a particular service, but at some point, the server for one queue goes down. You must now merge the two queues. If no other information is available, what might a reasonable strategy be for forming a single queue out of the two queues? What additional information would be necessary to form a single queue in such a way to be most fair to all requests?

Q12. An implementation could potentially use any of the following data structures to implement the Queue. Indicate the best-case run times if we use the listed data structures in as optimal a sense as possible. Choose the implementation that minimizes the run time of as many operations as possible. In the case of arrays, you may have to move entries if the array is not full—an O(n) operation.

	Singly linked list a head pointer	Singly linked list with head and tail pointers	Doubly linked list with head and tail pointers	One-ended array	Two-ended array	Circular array
`void push(…)`						
`Type pop()`						
`Type top()`						

Q13. Rank the following functions by increasing order of growth. That is, find any arrangement g1, g2, g3, g4, g5, g6, g7, g8 of the functions satisfying g1 = O(g2), g2 = O(g3), g3 = O(g4), g4 = O(g5), g5 = O(g6), g6 = O(g7), g7 = O(g8).

$$f_1(n) = n^{\pi} \qquad f_2(n) = \pi^n \qquad f_3(n) = \binom{n}{5} \qquad f_4(n) = \sqrt{2^{\sqrt{n}}}$$

$$f_5(n) = \binom{n}{n-4} \qquad f_6(n) = 2^{\log^4 n} \qquad f_7(n) = n^{5(\log n)^2} \qquad f_8(n) = n^4 \binom{n}{4}$$