Question 1: 11 points (3 + 4 + 4)

a. Show the running time of the following code using big O notation (show your work):

```
for (int i = 0; i < n + 100; ++i)
{
        for (int j = 0; j < i * n ; ++j)
        {       sum = sum + j;
        }
        for (int k = 0; k < n + n + n; ++k)
        {       c[k] = c[k] + sum;
        }
}
```

Ans. Outer for loop is n + 100 times, first inner for loop is n^2 times and 2nd inner for loop is n + n = 2n times.

Hence, total running time $T(n) = O(n^3) + O(n^2) = O(n^3)$

b. Determine whether f is O(g), Theta (g) or Omega (g). Mention all that applies. Show your work.

i) $f = 2^{2n}$, $g = 2^n$

2n times.

Hence, total running time T(n) = O(n²) + O(n²) = O(n²)

b. Determine whether f is O(g), Theta (g) or Omega (g). Mention all that applies. Show your work.

i) $f = 2^{2n}$, $g = 2^n$

Ans.

i) $f = 2^{2n}$, $g = 2^n$

Lt f/g (with n -> infinity) = $2^{2n} / 2^n = 2^n$ = Infinity. So, f is Omega (g) i.e. g is o(f) as shown below:

Lt g/f = $2^n / 2^{2n} = 1/2^n = 1/\text{Infinity} = 0$.

ii) $f = n^{\log c}$, $g = c^{\log n}$

Ans. If we take f/g, it will be undefined at n = infinity. Hence, we can take the log of f/g i.e.

Lt log(f/g) = Lt [log(f) - log(g)] = Lt [log $n^{\log c}$ - log $c^{\log n}$] = Lt [log c log n - log n log c] = 0.

This means log (f/g) = 1. Hence, f is O(g), g is O(f) and f is Theta (g).

Step 4:

[5]$_4$ k =1, pivot is 5. So, return 5.

Question 4: 12 points (3+5+4)

a. Is Mathematics Consistent? Explain your answer with an example.

Ans. No, mathematics is not consistent. Consider the following example,

X < 2^x in general for all x >= 0. However at x = Infinity, x both left side and right side is infinity, and hence inconsistent.

b.

① Why do we need to use "Expected Value" in computing running time in Bucket sort?

Ans.

Total number in the input array needs to be put in the buckets. However, if a cell gets more than one number then we need to use something similar to Linked list to create more cells. The data in such cells can be sorted using e.g. Insertion Sort.

Since, depending on the input data, different cells will get different # of cells, the data distribution in each bucket is random. Hence, we need to use an Expected value.

AutoSave Off Midterm_Dec_2021_SOLN - Compatibility Mode - Saved to this PC

Home Insert Draw Design Layout References Mailings Review View Help

Search (Alt+Q)

c. Give a Big O estimate for $f(x) = (x^3 + 4)\log(x^2 + 1) + 4x^3$

Ans. The first term has a multiplication. The first part of the first term is $O(x^3)$. The 2nd term can be analyzed as follows:

$X^2 + 1 < 2x^2$ for $x > 1$,... Hence $\lg(x^2 + 1) <= \lg(2x^2) = \lg 2 + 2\lg x <= 3\lg x$ for $x > 2$. So, for $x > 2$, 2nd part is $O(\lg x)$. Hence, the order of the first term is $O(x^3 . \lg x)$.

The 2nd term is $O(x^3)$... Now, taking the max of the orders of the 2 terms, we have, $O(x^3 . \lg x)$ which is the required answer.

Comments

Text Predictions: On Accessibility: Unavailable

Focus

$T(n) = [5^{n-1+1} - 1] / (5-1) = [5^n - 1] / 4$ which is $O(5^n)$

Question 2 (contd): 12 points (4 + 4 + 4)

c. Use Induction to prove that $2^n \leq n!$, Use base case as n = 4.

Ans.

Base case n = 14. LS = 16, RS = 4! = 24. So, base case is valid as $16 < 24$.

Inductive case: Assume for k i.e. assume $2^k \leq k!$, and show for K+1.

$2^k \leq k!$ as per assumption

Or $2 . 2^k < 2.k!$ (multiply both sides by 2). – (1)

Now consider right side : $2k! < (k+1).k!$ as k+1 is > 2 [A mathematical fact needed for this problem)

Thus, from (1) 2. $2^k < (k+1).k!$

Or $2^{k+1} < (k+1).k!$ i.e. $(k+1)!$

substitution or Master Theorem.

a. Consider the following recurrence algorithm [Use Master Theorem – See LAST Page]

```
long power(long x, long n)
    if (n==0) return 1;
    if (n==1) return x;
    if ((n % 2) == 0)
        return power(x*x, n/2);
    else
        return power(x*x, n/2) * x;
```

Assume n is power of 2.

i. (2 points) Write a recurrence equation for T(n)

Soln.

T(0) = 2
T(1) = 2
T(n) = T(n/2) + 5

ii. (2 points) Solve recurrence equation using Master's method i.e. give an expression for the runtime T(n).

Ans.

Soln.

A = 1, b = 2, c = 5, k = 0.
So, $a = b^k$
Thus, we have, T(n) = Theta (lgn)

Note that some cells may not even get any data.

(ii) Why Quicksort, in general performs better than Mergesort. Explain with an example using solutions for running time T(n) for both methods.

Ans.

The Big O equation for both QS and MS are same i.e. O(nlogn). However, when we solve this equation, we get the following:

$T_{qs}(n) = C_{qs}(n\log n)$ (1)

and

$T_{ms}(n) = C_{ms}(n\log nn)$ (2)

However, Cqs is about 3 times smaller than Cms as Merge operation needs lots of processing using temporary storage in MS, especially when array is large. Also, the possibility of getting Worst case of $O(n^2)$ in QS is very low as per Chernoff bound. So, even though theoretically QS can take longer time, statistically, it is not that high.

c. Use basic concepts of "Decision Tree" and prove that all comparison bound sorting algorithms cannot do better than O(nlog n).

Ans.

Ans. Consider 4d as the pivot. Initially i is at 4a and j is at 4c. They are both stuck as their value is 4 (duplicate). So, we swap them and then advance i to the right and j to the left. Thus, we have {4c, 4b, 4a, 4d}

(i,j)

Now, i and j are stuck at 4b and so we advance both by one position and thus we have

{4c, 4b, 4a, 4d}

(j) (i)

Since i crossed j, we are done after swapping value indicated by i with the pivot. The final position is

{4c, 4b, 4d, 4a}. So, the next iteration will be on {4c, 4b, 4d} and {4a}. Clearly, no matter what we do on the first sub array, 4a is already out of the original order as it will stay on the right of 4b and 4c, and hence Quick sort is unstable.

b. Use RadixSort, using LSB to MSB (IBM method) for the following array:

A = [455, 61, 63, 45, 67, 135, 74, 49, 15, 5]

Soln:

[455, 61, 63, 45, 67, 135, 74, 49, 15, 5] – Original unsorted array

[455, 061, 063, 045, 067, 135, 074, 049, 015, 005] – length of each number adjusted

LSB sorted [061, 063, 074, 455, 045, 135, 015, 005, 067, 049]

* BST delete node, show steps.
* Build heap
* build Red Black Tree
* Bellmanford apply
* Red Black Tree proof $2\log(n+1) \geq h$
* Dijkstra algorithm and running time given, show how?
* NP T or F x3
* Why Dijkstras algorithm fails for negative values?
* Using kruskal → draw MST show steps.
* Logical Gate, Satisfiable?
* Knapsack as in the lab
* Knapsack runningtime
* PowerSet is it NP-Complete?