Recurrences Questions

Recurrences MCQ Question 1

View this Question Online >

Which one of the following correctly determines the solution of the recurrence relation with
T(1) = 1?

$$T\left(n\right) = 2T\left(\frac{n}{2}\right) + \log n$$

1. Θ(n)
2. Θ(n log n)
3. Θ (n^2)
4. Θ(log n)

Answer (Detailed Solution Below)

Option 1 : Θ(n)

Recurrences MCQ Question 1 Detailed Solution

$$\mathrm{T}\left(\mathrm{n}\right) = 2\mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + \log \mathrm{n}$$

Comparing with:

$$\mathrm{T}\left(\mathrm{n}\right) = \mathrm{aT}\left(\frac{\mathrm{n}}{\mathrm{b}}\right) + f\left(n\right)$$

a = 2, b = 2, f(n) = log n

$n^{\log_2 2} = n$ > f(n)

By Master's theorem

$T\,(\mathrm{n}) = \mathrm{O}\,(\mathrm{n})$

Recurrences MCQ Question 2

View this Question Online >

What is the time complexity for the following C module? Assume that n > 0;

```
int module(int n)
{
if(n == 1)
return 1;
else
return (n + module(n-1));
}
```


1. O(n)
2. O(log n)
3. O(n^2)
4. O(n!)

Answer (Detailed Solution Below)

Option 1 : O(n)

Recurrences MCQ Question 2 Detailed Solution

Answer: Option 1

Explanation:

f(n) = n + module(n-1); // since **return (n + module(n-1));**

Here recurrence relation will be

T(n) = T(n-1) + c

≡ T(n-2) + c + c

≡ T(n-3) + 3c

≡ T(n-k) + kc

when n-k = 1 ; k = n-1 and T(1) = 1

≡ T(1) + (n-1)c

≡ (n-1)c

≡ O(n)

Hence Option 1 is correct answer.

Confusion Points

Here some of you might have cam up with recurrence relation T(n) = T(n-1) + n, which is incorrect because in function n is just a variable it will have a constant value which will get added to call return value and In recurrence variable n indicate cost.

Recurrences MCQ Question 3

View this Question Online >

Consider the recurrence relation :
$T(n) = 8T\left(\frac{n}{2}\right) + Cn, if\ n > 1$

= b, if n = 1
Where b and c are constants.
The order of the algorithm corresponding to above recurrence relation is :

1. n
2. n^2
3. n lg n
4. n^3

Answer (Detailed Solution Below)

Option 4 : n^3

Recurrences MCQ Question 3 Detailed Solution

The correct answer is **option 4.**

Master's theorem:

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ where a > = 1 and b > 1

Case: **1.** If f(n) = $\Theta(n^c)$ where c < $Log_b a$ then T(n) = $\Theta(n^{\log_b a})$

EXPLANTION

$T(n) = 8T\left(\frac{n}{2}\right) + Cn, if\ n > 1$

= b, if n = 1

Where b and c are constants.

The above recurrence is in the form of the

Here, c = $Log_2 8$ = 3

f(n) = $\Theta(n^3)$

Hence the correct answer is n^3.

Recurrences MCQ Question 4

View this Question Online >

Consider the algorithm that solves problems of size n by recursively solving two sub problems of size n - 1 and then combining the solutions in constant time. Then the running time of the algorithm would be:

1. $O(2^n)$
2. O(logn)
3. O(nlogn)
4. $O(n^2)$

Answer (Detailed Solution Below)

Option 1 : $O(2^n)$

Recurrences MCQ Question 4 Detailed Solution

The correct answer is **option 1.**

Key Points

The recurrence relation is,

T(n)=2.T(n-1)+c , n>0

T(n)=1, n=0.

as per the back substitution method,

$T(n) = 2^kT(n-k)+ 2^{k-1}c+....+2^0c$ substitute where, n-k=0

$T(n) = 2^nT(0)+ 2^{k-1}c+....+2^0c$

$T(n) = 2^{n+1}-1$

$T(n) = O(2^n)$

Hence the correct answer is: $O(2^n)$

Additional Information

- Recurrence relation of Tower of Hanoi is, $T(n)=2.T(n-1)+c$
- Time complexity of the Tower of Hanoi is $O(2^n)$.

Mistake Points

No option is correct so here I modified option 1 $O(n)$ to $O(2^n)$.

Recurrences MCQ Question 5 View this Question Online >

The master theorem

1. Assumes the subproblems are unequal sizes
2. can be used if the subproblems are of equal size
3. cannot be used for divide and conquer algorithms
4. cannot be used for asymptotic complexity analysis

Answer (Detailed Solution Below)

Option 2 : can be used if the subproblems are of equal size

Recurrences MCQ Question 5 Detailed Solution

Concept:

The master's theorem divides the problem into a finite number of sub-problems each of same size and solves recursively to compute the running time taken by the algorithm.

Explanation

Master Theorem is used to determine running time of algorithms (divide and conquer algorithms) in terms of asymptotic notations.

According to master theorem the runtime of the algorithm can be expressed as:

$T(n) = aT(n/b) + f(n)$,

where,

n = size of input

a = number of sub-problems in the recursion

n/b = size of each sub-problem.

f(n) = cost of the work done outside the recursive call, which includes the cost of dividing the problem and cost of merging the solutions.

Here, $a \geq 1$ and $b > 1$ are constants, and f(n) is an asymptotically positive function.

Recurrences MCQ Question 6

View this Question Online >

For constants $a \geq 1$ and $b > 1$, consider the following recurrence defined on the non-negative integers:

$$T(n) = a.T\left(\frac{n}{b}\right) + f(n)$$

Which one of the following options is correct about the recurrence T(n)?

1. If f(n) is $\frac{n}{\log_2(n)}$, then T(n) is $\theta(\log_2(n))$.

2. If f(n) is $n \log_2(n)$, then T(n) is $\theta(n \log_2(n))$.

3. If f(n) is $O(n^{\log_b(a) - \epsilon})$ for some $\epsilon > 0$, then T(n) is $\theta(n^{\log_b(a)})$.

4. If f(n) is $\theta(n^{\log_b(a)})$, then T(n) is $\theta(n^{\log_b(a)})$

Answer (Detailed Solution Below)

Option 3 : If f(n) is $O(n^{\log_b(a) - \epsilon})$ for some $\epsilon > 0$, then T(n) is $\theta(n^{\log_b(a)})$.

Recurrences MCQ Question 6 Detailed Solution

Key Points

The recurrence T(n) = aT(n/b) + f(n), where a, b are constants. Then

(A) If $f(n) = O(n^{\log_b(a) - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \theta(n^{\log_b(a)})$..

(B) If $f(n) = \theta(n^{\log_b(a)})$., then $T(n) = \theta(n^{\log_b a} \log n)$

(C) If $f(n) = \Omega(n^{\log_b(a) + \varepsilon})$ for some constant $\varepsilon > 0$, and if f satisfies the smoothness condition $a \times f(n/b) \le c \times f(n)$ for some constant $c < 1$, then $T(n) = \Theta(f(n))$.

The above statements illustrate the master's theorem.

option 3 is the correct answer

Recurrences MCQ Question 7

View this Question Online >

Solve the following recurrence relation:
$T(n) = 4T(n/2) + n^2$

1. $\Theta(n^3)$

2. $\Theta(n^2)$

3. $\Theta(n^2 \log n)$

4. Θ (n^2 /2)

Answer (Detailed Solution Below)

Option 3 : Θ (n^2log n)

Recurrences MCQ Question 7 Detailed Solution

Recurrence relation: T(n) = 4T(n/2) + n^2

Comparing with T(n) = aT(n/b) + f(n)

a = 4 and b = 2

∴ a ≥ 1 and b > 1

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

$$f(n) = \theta\left(n^{\log_b a}\right)$$

By using mater theorem:

$$T(n) = \theta(n^{\log_b a}\ \log(n)) = \theta(n^2 \log n)$$

Recurrences MCQ Question 8 View this Question Online >

Raymonds tree based algorithm ensures

1. no starvation, but deadlock may occur in rare cases
2. no deadlock, but starvation may occur
3. neither deadlock nor starvation can occur

testbook.com

4. deadlock may occur in cases in cases where the process is already starved

Answer (Detailed Solution Below)

Option 2 : no deadlock, but starvation may occur

Recurrences MCQ Question 8 Detailed Solution

Raymond's tree-based algorithm is a lock-based algorithm that ensures mutual exclusion in a distributed system.

Steps of Algorithm:

1. A site is allowed to enter the critical section if it has the token.
2. Site which holds the token is also called root of the tree.
3. For acquiring token, all the other sites are arranged as a directed tree such that the edges of the tree are assigned direction towards the root.

Raymond's algorithm uses greedy approach and a site can enter the critical section on receiving the token even if its request is not on the top of the request queue. This affects the fairness of the algorithm sometimes leading to starvation.

However, deadlock is not possible to occur.

Recurrences MCQ Question 9

View this Question Online >

Which one of the following correctly determines the solution of the recurrence relation with
T(1) = 1?

$$T(n) = 2T\left(\frac{n}{2}\right) + \log n$$

1. Θ(n)
2. Θ(n log n)
3. Θ (n^2)
4. Θ(log n)

testbook.com

Answer (Detailed Solution Below)

Option 1 : Θ(n)

Recurrences MCQ Question 9 Detailed Solution

$$\mathrm{T}(\mathrm{n}) = 2\mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + \log \mathrm{n}$$

Comparing with:

$$\mathrm{T}(\mathrm{n}) = \mathrm{aT}\left(\frac{\mathrm{n}}{\mathrm{b}}\right) + f(n)$$

a = 2, b = 2, f(n) = log n

$n^{\log_2 2} = n$ > f(n)

By Master's theorem

$$T(\mathrm{n}) = \mathrm{O}(\mathrm{n})$$

Recurrences MCQ Question 10

View this Question Online >

Consider the following recursive function:

```
int sum(int n)
{
   if (n == 1)
   {
      return 1;
   }
   return n + sum(n-1);
}
```

What is the time complexity of the above code?

1. $\Theta(n^2)$
2. $\Theta(n)$
3. $\Theta(\log n)$
4. $\Theta(n \log n)$

Answer (Detailed Solution Below)

Option 2 : $\Theta(n)$

Recurrences MCQ Question 10 Detailed Solution

Recurrence relation for above code is

$T(1) = 1$

$T(n) = 1 + T(n-1)$, when $n > 1$

$1 + T(n-1)$

$= 1 + (1 + T(n-2))$

$= 2 + T(n-2)$

$= 2 + (1 + T(n-3))$

$= 3 + T(n-3)$ ...

$= k + T(n-k)$ ...

$= n - 1 + T(1)$

$= n - 1 + 1$

$= \Theta(n)$