Quizz Solution

Analysis of Recursive Algorithms

(Group 2)

Link: here

Quizz 1:

Give following code:

```
def BinRec(n):
  if n == 1:
    return 1
  else:
    return BinRec(floor(n/2))+1
```

What's the time complexity for the above code

A. $\theta(n)$
B. $\theta(n^2)$
C. $\theta(1)$
D. $\theta(\log n)$

Answer : D

Solution:

Recurrance relation T(n) = T(n/2) + 1. Use backward substitution.

Solve T(n) = $\theta(logn)$

Qizz 2:

Give following code:

```
def Secret(n, s, d, a):
  if (n > 1):
    Secret(n-1, s, a, d)
    Secret(n-1, a, d, s)
```

What's the recurrence realtion for the above code:

A. T(n) = T(n-2) + 1
B. T(n) = 2T(n-1) + 1
C. T(n) = T(n-1) + 2
D. T(n) = 2

Answer : B

Solution:

Setup follow above code

Qizz 3:

Give following code:

```
def Secret(n, s, d, a):
  if (n > 1):
    Secret(n-1, s, a, d)
    Secret(n-1, a, d, s)
```

What's the time complexity for the above code:

A. $\theta(n)$
B. $\theta(n^2)$
C. $\theta(2^n)$
D. $\theta(n \log n)$

Answer : C

Solution:

Use quizz 2 answer. Recurrance relation is T(n) = 2T(n-1) + 1. Use backward substitution.

We get T(n) = $\theta(2^n)$

Qizz 4:

Give following code:

```
def Q(n):
  if (n == 1):
    return 1
  else:
    return Q(n-1) + 2*n - 1
```

what's the result when solve the recurrence relation for function's value (Q) follow the code above?

A. $Q(n) = n^2 - 2n$
B. $Q(n) = n$
C. $Q(n) = n^2$
D. $Q(n) = n^2 + 2n - 1$

Answer : A

Solution

Recurrence relation for Q(n) = Q(n-1) + 2n – 1.

$$Q(n) = Q(n-1) + 2n - 1, Q(1) = 1$$

Use telescope we have:

$$Q(n) = \sum_{i=1}^{n} 2n - 1 = 2\sum_{i=1}^{n} n - \sum_{i=1}^{n} 1 = n(n-1) - n = n^2 - 2n$$

Qizz 5:

Give recurrence relation:

$$F_n = F_{n-1} + F_{n-2}, \qquad n > 1, F_0 = 0, F_1 = 5$$

Follow slide 21, what’s the result what’s the result when solve the recurrence relation above.

A. $F_n = (\phi^n - \hat{\phi}^n)$
B. $F_n = 5\phi^n$
C. $F_n = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$
D. $F_n = \sqrt{5}\phi^n - \sqrt{5}\hat{\phi}^n$

Answer : D

Solution

Follow slide 21 we have:

$$F_n = c_0\phi^n + c_1\hat{\phi}^n$$

$$F_0 = c_0 + c_1 = 0$$

$$F_1 = c_0\phi + c_1\hat{\phi} = 5$$

$$Solve\ for\ c_1,\ c_2\ we\ have\ c_0 = \sqrt{5}\ and\ c_1 = \ -\sqrt{5}$$

$$So:\ F_n = \sqrt{5}\phi^n - \sqrt{5}\hat{\phi}^n$$

Quizz 6:

Not following the slide, how many methods of solving reccurence relation that I mentioned in today's session.

A. 3
B. 4
C. 5
D. 6

Answer : C

Solution

Backward substitution, telescope, use Similar function, master theorem, recursion tree.

Qizz 7:

Give recurrence relation:

$$F_n = F_{n-1} + F_{n-2}, \qquad n > 1, F_0 = 0, F_1 = 1$$

Follow slide 21 (or not), what is 23th Fibonacci number (F_{23}).

A. 17711
B. 10948
C. 28657
D. 39605

Answer : C

Solution

Following slide 21 we have $F_n = \frac{\phi^n}{\sqrt{5}} - \frac{\hat{\phi}^n}{\sqrt{5}}$, with n = 23 so $F_n = 28657$

Qizz 8:

What is the base case of a recursive **algorithm**?

A. The step that breaks the problem down into smaller subproblems.
B. The step that combines the solutions of the subproblems.
C. The step that sets the initial conditions of the problem.
D. The step that returns the solution of the problem for the smallest possible input.

Answer : D.

Solution:

the base case of a recursive **algorithm is** The step that returns the solution of the problem for the smallest possible input.

Qizz 9:

Which of the following is an example of a recursive algorithm with exponential time complexity?

A. Binary search
B. Merge sort
C. Fibonacci sequence calculation
D. Quick sort

Answer : C.

Solution

Following slide 22 we have time complexityt of Fibonacci sequence calculation is $O(1.6180^n)$. This is exponential time complexity.

Qizz 10:

Give the following recurrence relations:

x(n) = 3x(n − 1) for n > 1, x(1) = 4

What is x(13)

A. 708588
B. 2125764
C. 19131874
D. 6377292

Answer : B. 2125764

Solution

Solve for x(n), use backward substitution we have, x(n) = 3(3x(n-2)) = 3(3(3x(n-3)))….

= $3^{n-1} * 4$

X(13) = $3^{12} * 4 = 2125764.$

Quizz for Master Method

Quizz 11:

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. $\Theta\ (n^2 /2)$

Answer Option 3 : $\Theta\ (n^2 \log n)$

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:

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

Quizz 12:

\What is the asymptotic value for the recurrence equation T(n) = 2T(n/2) + n?

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

Answer Option 4 : O(n log n)

Detailed Solution:

Master Theorem: T(n)= aT(n/b)+f(n); a>=1,b>1.

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

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

$$f(n) = n^{log_b a} = n$$

∴ T(n) = O(n(logn))

Quizz 13:

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

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

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.

Quizz 14:

What is the complexity of $T(n) = 2T(n/4) + n^2 \times \log_2 n$?

1. $\theta(n^2 \times \log_2(\log_2 n))$
2. $\theta(n^3 \times \log_2 n)$
3. $\theta(n^2 \times \log_2 n)$
4. $\theta(n \times \log_2 n)$

Answer Option 3 : $\theta(n^2 \times \log_2 n)$

Detailed Solution

$T(n) = 2T(n/4) + n^2 \times \log_2 n$

Comparing with:

$T(n) = aT(n/b) + n^k \times \log_2^p n$

a = 2, b = 4, k = 2, p = 1

$b^k = 4^2 = 16$

$a < b^k$ and $p \geq 0$

$T(n) = \theta(n_k \times \log_{2p} n)$

$\mathbf{T(n) = \theta(n^2 \times \log_2 n)}$

Quizz 15:

Solve the following recurrence relation if T(1) = 1?

$T(n) = T(n/2) + 2^n$

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

Answer Option 1 : $\Theta(2^n)$

Detailed Solution:

On comparing with equation $T(n) = aT(n/b) + f(n)$ we get

$a = 1, b = 2, f(n) = 2^n$

Here, $f(n) = \Omega(n)$ … (3^{rd} case of master theorem)

$f(n) = \Omega(n)$

and by regularity condition,

$af(n/b) \leq cf(n)$

$2^{n/2} \leq c2^n$

$T(n) = \Theta(f(n)) = \Theta(2^n)$

Quiz for Recursion Tree Method

Quizz 16:

Solve the following recurrence relation

$T(n) = T(n/2) + n$ when $n > 1$ and $T(n) = 1$ when $n = 1$.

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

Answer Option 1 : $O(n)$

Detailed Solution

<u>Recurrence relation:</u>

$T(n) = T(n/2) + n$ when $n > 1$ and $T(n) = 1$ when $n = 1$.

Using substitution method to solve recurrences,

$T(n) = T(n/2) + n$

$T(n) = T(n/2^2)+ n/2 + n$

$T(n) = T(n/2^3) + n/2^2 + n/2 + n$

Substituting K times.

$T(n) = T(n/2^k) + n/2^{k-1}\dots\dots\dots +n/2^2 + n/2 + n$

Using Termination

$n/2^k = 1$

$n = 2^k$

$k = \log^n$

Put value of K in above equation

$T(n) = T(n/2^{\log n}) + n/2^{\log n - 1}\dots\dots\dots + n/2^2 + n/2 + n$

$T(n) = T (1) + n/2^{\log n - 1} + n/2^{\log n - 2}\dots\dots\dots +n/2^2 + n/2 + n$

$T(n) = 1 + n (1/2^{\log n - 1} + 1/2^{\log n - 2}\dots\dots\dots +1/2^2 + 1/2 + 1/2^0)$

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

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

$T(n) = 1 + 4n$

$T(n) = O(n)$

Tip: Sum of G.P. series with $a = 1$ and $r = 1/2$

$sum = a((1 - r^n))/(1 - r))$

Quizz 17:

What is the solution of given recurrence relation?

$T(n) = T (n – 2) + \log^n$ when $n > 0$ and $T(n) = 0$ when $n = 0$.

1. $O(n)$
2. $O(n \log^n)$
3. $O(\log^n)$

4. $O(n^2)$

Answer Option 2 : $O(n \log^n)$

Detailed Solution

Recurrence relation:

$T(n) = T(n-2) + \log^n$ when $n > 0$ and $T(n) = 0$ when $n = 0$.

Using substitution method to solve recurrences,

$T(n) = T(n-2) + \log^n$

$T(n) = T(n-4) + \log^{n-2} + \log^n$

$T(n) = T(n-6) + \log^{n-4} + \log^{n-2} + \log^n$

Substituting K times.

$T(n) = T(n-2K) + \log^{(n-2K-2)} \ldots\ldots\ \log^{n-4} + \log^{n-2} + \log^n$

Using Termination

$n - 2K = 0$

$K = n/2$

Put value of K in above equation

$T(n) = T(n - 2.\ n/2) + \log^{(n-2.n/2-2)} + \ldots\ldots\ \log^{n-4} + \log^{n-2} + \log^n$

$T(n) = T(0) + \log^2 + \log^4 + \ldots\ldots\ \log^{n-4} + \log^{n-2} + \log^n$

$T(n) = 1 + \log^{(2.4.6.8\ldots\ldots \log n-4\ .\ \log n-2\ .\ \log n)}$

$T(n) = 1 + \log^{2n/2(1.2.3.4\ldots\ldots n/2)}$

$T(n) = O(\log^{n/2!)}$

$T(n) = O(\log n^n)$

$T(n) = O(n\ .\ \log^n)$

Quizz 18:

What is the solution of given recurrence relation?

$T(n) = T(n/2) + c$ when $n > 1$ and $T(n) = 1$ when $n = 1$ where ‘c’ is a constant.

1. $O(n)$

2. $O(n \log^n)$
3. $O(\log n)$
4. $O(n^2)$

Answer Option 3 : O(logn)

Detailed Solution

Recurrence relation:

$T(n) = T(n/2) + c$ when $n > 1$ and $T(n) = 1$ when $n = 1$.

Using substitution method to solve recurrences,

$T(n) = T(n/2) + c$

$T(n) = T(n/2^2) + c + c$

$T(n) = T(n/2^3) + c + c + c$

Substituting K times.

$T(n) = T(n/2^k) + k.c$

Using Termination

$n/2^k = 1$

$n = 2^k$

$k = \log n$

Put value of K in above equation

$T(n) = T(n/2^{\log n}) + \log nc$

$T(n) = T(1) + \log n.c$

$T(n) = 1 + \log n.c$

$T(n) = O(\log n)$

Quizz 19:

What is the solution of given recurrence relation?

$T(n) = T(n/2) + c$ when $n > 1$ and $T(n) = 1$ when $n = 1$ where ‘c’ is a constant.

1. $O(n)$

2. $O(n \log^n)$
3. $O(\log n)$
4. $O(n^2)$

Answer Option 3 : $O(\log n)$

Detailed Solution

Recurrence relation:

$T(n) = T(n/2) + c$ when $n > 1$ and $T(n) = 1$ when $n = 1$.

Using substitution method to solve recurrences,

$T(n) = T(n/2) + c$

$T(n) = T(n/2^2) + c + c$

$T(n) = T(n/2^3) + c + c + c$

Substituting K times.

$T(n) = T(n/2^k) + k.c$

Using Termination

$n/2^k = 1$

$n = 2^k$

$k = \log n$

Put value of K in above equation

$T(n) = T(n/2^{\log n}) + \log nc$

$T(n) = T(1) + \log n.c$

$T(n) = 1 + \log n.c$

$T(n) = O(\log n)$

Quizz 20:

Consider the following recursive function:

int sum(int n)

{

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

What is the time complexity of the above code?

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

Answer Option 2 : Θ (n)

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

= Θ(n)