**2. Given a recurrence relation, solve it using the substitution method:**

**a. T(n) = T(n-1) + c**

To solve the recurrence relation T(n)=T(n−1)+c using the substitution method, follow these steps:

**1. Guess the Solution**

First, make an educated guess about the form of the solution. For a recurrence relation of the form T(n)=T(n−1)+c, where c is a constant, a common guess is that T(n) might be linear. Let's guess that:

T(n)=A⋅n+B

where A and B are constants to be determined.

**2. Substitute the Guess into the Recurrence Relation**

Substitute T(n)=A⋅n+B into the recurrence relation T(n)=T(n−1)+c

T(n)=A⋅n+BT

Now substitute T(n−1) into the recurrence relation:

T(n)=(A⋅(n−1)+B)+c

**3. Simplify the Expression**

Simplify the right-hand side:

T(n)=A⋅(n−1)+B+c

T(n)=A⋅n−A+B+c

**4. Match Terms to Determine Constants**

Compare the expression A⋅n−A+B+c to the guess T(n)=A⋅n+B . To match the two expressions, the coefficients of n and the constant terms must be equal:

A⋅n+B=A⋅n−A+B+c

By comparing terms, we see that the coefficient of n is already matched. Now, set the constant terms equal:

B=−A+B+c

Simplify to find A:

0=−A+c

A=c

So, our guess for A is correct and A=c.

**5. Determine the Final Solution**

Since A=c, the general solution for T(n) can be written as:

T(n)=c⋅n+B

To find B, we need the initial condition T(0)=T0 ,where T0​ is the base case value. Substitute n=0 into the final solution:

T(0)=c⋅0+B

So, B=T(0)

**6. Final Solution**

Thus, the solution to the recurrence relation T(n)=T(n−1)+c

T(n)=c⋅n+T(0)

where T(0) is the base case value of the recurrence relation.

**b. T(n) = 2T(n/2) + n**

To solve the recurrence relation T(n)=2T(n/2)+n using the substitution method, follow these steps:

**1. Guess the Solution**

We’ll guess that T(n) has the form:

T(n)=A⋅nlogn+B⋅n

where A and B are constants to be determined.

**2. Substitute the Guess into the Recurrence Relation**

Substitute T(n)=A⋅nlogn+B⋅n into the recurrence relation T(n)= 2T(n/2) +n:

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

First, compute T(n/2) using the guess:

T(n/2)=A⋅(n/2)log(n/2)+B⋅(n/2)

Simplify log(n/2):

log(n/2)=logn−log2

So:

T(n/2)=A⋅(n/2)⋅(logn−log2)+B⋅(n/2)

T(n/2)=A⋅(n/2)⋅logn−A⋅(n/2)⋅log2+B⋅(n/2)

Now substitute T(n/2) back into the recurrence relation:

T(n)=2[A⋅(n/2)⋅logn−A⋅(n/2)⋅log2+B⋅(n/2)]+n

T(n)=A⋅n⋅logn−A⋅n⋅log2+B⋅n+n

**3. Simplify the Expression**

Combine and simplify the terms:

T(n)=A⋅n⋅logn−A⋅n⋅log2+B⋅n+n

T(n)=A⋅n⋅logn+(B−A⋅log2+1)⋅n

**4. Match Terms to Determine Constants**

Compare this expression to our guess T(n)=A⋅nlogn+B⋅n

We need to match the term:

T(n)=A⋅n⋅logn+(B−A⋅log2+1)⋅n

From this, we see:

B=B−A⋅log2+1

To match the coefficients, we get:

B=B−A⋅log2+1

This simplifies to:

0=−A⋅log2+1

A=1/log2

​

**5. Final Solution**

Substitute A into our guess:

T(n)=nlog2⋅logn+B⋅n

To find B, use the base case T(1) if provided, but often Bremains arbitrary and absorbed into the general form. For simplicity, assume B=0 if not specified:

T(n)=n⋅logn/log2+constant

In general, the solution to the recurrence relation T(n)=2T(n/2)+n is

T(n)=Θ(nlogn)

This result indicates that the time complexity grows proportional to nlogn , reflecting the combined effect of the recursive calls and the linear term in the recurrence relation.

**c. T(n) = 2T(n/2) + c**

To solve the recurrence relation T(n)=2T(n/2)+cT(n) = 2T(n/2) + cT(n)=2T(n/2)+c using the substitution method, follow these steps:

**1. Guess the Solution**

We’ll guess that T(n) has the form:

T(n)=A⋅n+B

where A and B are constants to be determined.

**2. Substitute the Guess into the Recurrence Relation**

Substitute T(n)=A⋅n+B into the recurrence relation T(n)=2T(n/2)+c:

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

Compute T(n/2)T using the guess:

T(n/2)=A⋅(n/2)+B

Substitute T(n/2) into the recurrence relation:

T(n)=2[A⋅(n/2)+B]+c

T(n)=A⋅n+2B+c

**3. Match Terms to Determine Constants**

Compare this expression to our guess T(n)=A⋅n+B. To match the two expressions, the coefficients of n and the constant terms must be equal:

T(n)=A⋅n+B

T(n)=A⋅n+2B+c

By comparing terms, we see:

B=2B+c

Solve for B:

B=2B+c

-B=C

B=-

**4. Determine the Final Solution**

Substitute B=−c back into our guess:

T(n)=A⋅n+(−c)

So:

T(n)=A⋅n−c

**5. Find the Value of A**

To find A, we need to ensure that our guess fits the form of the recurrence relation. We see that the coefficient A does not change and can be any constant as it cancels out in the recurrence relation. Therefore

A=A

**6. Final Solution**

Combining all terms, the solution to the recurrence relation T(n)=2T(n/2)+c

T(n)=Θ(n)

This indicates that the time complexity grows linearly with n. The constant term c does not affect the linear growth rate, so the solution is T(n)=Θ(n).

**d. T(n) = T(n/2) + c**

To solve the recurrence relation T(n)=T(n/2)+cT(n) = T(n/2) + c

**1. Guess the Solution**

We’ll guess that T(n) has the form:

T(n)=A⋅logn+B

where A and B are constants to be determined.

**2. Substitute the Guess into the Recurrence Relation**

Substitute T(n)=A⋅logn+B into the recurrence relation T(n)=T(n/2)+c

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

Compute T(n/2) using the guess:

T(n/2)=A⋅log(n/2)+B

Simplify log(n/2):

log(n/2)=logn−log2

So:

T(n/2)=A⋅(logn-log2)+B

T(n/2)=A⋅logn−A⋅log2+B

Substitute T(n/2) into the recurrence relation:

T(n)=(A⋅lognA⋅log2+B)+C

T(n)=A⋅logn−A⋅log2+B+c

**3. Match Terms to Determine Constants**

Compare this expression to our guess T(n)=A⋅logn+B

For the terms to match:

T(n)=A⋅logn+B

T(n)=A⋅logn−A⋅log2+B+c

By comparing terms, we get:

B=B−A⋅log2+c

Solve for A:

0=−A⋅log2+c

A=c/log2 ​

**4. Determine the Final Solution**

Substitute A=clog2 ​ back into our guess:

T(n)=c/log2⋅logn+B

To determine B, use the base case T(1) if provided. If not specified, B can be a constant. The solution in general form is:

T(n)=clog2⋅logn+constant

**5. Final Solution**

Combining all terms, the solution to the recurrence relation T(n)=T(n/2)+c

T(n)=Θ(logn)

This indicates that the time complexity grows logarithmically with n. The term c/log2 ​ scales the logarithmic function, but it doesn’t change the overall logarithmic complexity.

**3. Given a recurrence relation, solve it using the recursive tree approach:**

**a. T(n) = 2T(n-1) +1**

To solve the recurrence relation T(n)=2T(n−1)+1

### ****1. Expand the Recurrence Relation****

Start by expanding the recurrence relation to understand the pattern of how T(n) is built from previous terms.

#### First Few Expansions

T(n)=2T(n−1)+1

Expand T(n−1):

T(n−1)=2T(n−2)+1

So:

T(n)=2[2T(n−2)+1]+1

T(n)=2^2T(n−2)+2^1+1

Expand T(n−2):

T(n−2)=2T(n−3)+1

So:

T(n)=2^2[2T(n−3)+1]+2^1+1

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

### ****2. Identify the General Pattern****

From the expansions, you can see a pattern forming:

T (n) = 2^k T(n-k) + 2^{k-1} + 2^{k-2} + …. + 2^1 + 2^0

The series 2^{k-1} + 2^{k-2} + ….. + 2^1 + 2^0 is a geometric series.

### ****3. Calculate the Sum of the Geometric Series****

The sum of the geometric series 2^{k-1} + 2^{k-2} + …. + 2^1 + 2^0 is:

Sum= 2^k - 1

### ****4. Determine the Base Case****

To solve the recurrence, determine the base case T(0) or T(1). Let’s assume T(0)=T0

​ for generality.

### ****5. Substitute the Base Case****

In the general formula, when k=n:

T(n)=2^nT(n−n)+2^n−1+2^n−2+⋯+2^1+2^0

Since T(n−n)=T(0)​:

T(n)=2^nT0+(2^n−1)

T(n)=2^n(T0​+1)−1

### ****6. Final Solution****

So the solution to the recurrence relation T(n)=2T(n−1)+ is:

T(n)=2n(T0​+1)−1

In Big-O notation, ignoring the constants and lower-order terms:

T(n)=Θ(2n)

This indicates that the time complexity of the recurrence relation grows exponentially with n.

**b. T(n) = 2T(n/2) + n**

To solve the recurrence relation T(n)=2T(n/2)+n using the recursive tree method,

### ****1. Draw the Recursive Tree****

For T(n)=2T(n/2)+n, each level of the recursion breaks the problem into two subproblems, each of size n/2. At each level, the cost is proportional to the size of the current problem.

#### First Level

At the first level, the cost is T(n)=2T(n/2)+n.

#### Second Level

At the second level, each T(n/2) splits into 2T(n/4), so we have:

T(n)=2[2T(n/4)+n/2]+n=4T(n/4)+2n

#### Third Level

At the third level, each T(n/4) splits into 2T(n/8), giving:

T(n)=4[2T(n/8)+n/4]+2n=8T(n/8)+3n

### ****2. Generalize the Pattern****

At each level, the number of subproblems doubles, and the work at each level is a multiple of n.

* **Level 0**: n
* **Level 1**: 2×n/2=n2
* **Level 2**: 4×n/4=n4
* **Level 3**: 8×n/8=n8
* **Level kkk**: 2k×n/2^k=n

So, the total work at each level remains n.

### ****3. Determine the Height of the Tree****

The height of the recursive tree corresponds to how many times we can divide n by 2. This is given by log2​n.

Thus, the tree has log2​n levels.

### ****4. Total Work****

At each level, the work is n, and there are log2​n levels. Therefore, the total work is:

Total Work=n⋅log2​n

### ****5. Final Solution****

The time complexity of the recurrence T(n)=2T(n/2)+n is:

T(n)=Θ(nlogn)

This indicates that the time complexity grows logarithmically with respect to n, multiplied by the size n at each level.