COP 3530
Section U02
Fall 2012

Recurrence Relations

Problem 1.

Solve the recurrence relation

$F(0) = 0$

$F(1) = 1$

$F(n) = F(n-1) + F(n-2) \qquad$ for $n > 1$

Problem 2.

Compute $T(n)$ for $n = 2^k$.

$T(n) = a \qquad$ for $n \leq 2$

$T(n) = 8T(n/2) + bn^2 \qquad$ for $n > 2$

Problem 3.

Compute $T(n)$ for $n = 2^k$.

$T(n) = a \qquad$ for $n \leq 2$

$T(n) = 7T(n/2) + bn^2 \qquad$ for $n > 2$

Problem 4.

Compute $T(n)$ for $n = 2^k$.

$T(n) = a \qquad$ for $n \leq 2$

$T(n) = 3T(n/2) + bn \qquad$ for $n > 2$

Problem 5.

Compute $T(n)$ for $n = 2^k$.

$T(n) = a \qquad$ for $n \leq 2$

$T(n) = bT(n/2) + cn^2 \qquad$ for $n > 2$

Problem 6.

Compute $T(n)$ for the Towers of Hanoi Problem.

$T(1) = 1$

$T(n) = 2T(n-1) + 1 \qquad$ for $n > 1$

Problem 7.

Compute $T(n)$ for any n.

$T(0) = 1$

$T(1) = -2$

$T(2) = 8$

$T(n) + 6T(n-1) + 12T(n-2) + 8T(n) = 0 \qquad$ for $n > 2$

Problem 8.

Compute $T(n)$ for any n.

$T(0) = 3$

$T(n) + 2T(n-1) = n + 3 \qquad$ for $n > 0$

Problem 9.

Let a, b, c be three positive integers. Compute $T(n)$ for $n = c^k$.

$T(1) = c$

$T(n) = aT(n/c) + bn \qquad$ for $n > 1$

Problem 10.

Compute $T(n)$ for any n.

$T(1) = 0$

$T(n) = T(n-1) + n - 1 \qquad$ for $n > 1$

Problem 11.

Compute $T(n)$ for $n = 2^k$.

$T(1) = 1$

$T(n) = T(n/2) + bnlog(n) \qquad$ for $n > 1$

Here b is a positive constant and $log(n)$ is the logarithm in base 2 of n.

Problem 12.

What is the O for the following programs?

a.

```
sum = 0;
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        sum = sum +a[i][j] ;
```

b.

```
for (int i = 0; i < n; i++)
```

```
    for (int j = 0; j < n; j++)
    {
        a[i][j] = 0.0;
        for (int k = 0; k < n; k++)
            a[i][j] = a[i][j] + b[i][k] * c[k][j];
    }
```

c.

```
int low = 0; high = n - 1;
while ( low <= high)
{
    int mid = ( low + high) / 2;
    if ( a[mid] == x)
        return true;
    if (a[mid] < x)
        low = m + 1;
    else
        high = mid -1;
}
return false;
```

d.

```
public static long F(int n)
{
    if ( n == 0)
        return 0;
    if ( n == 1)
        return 1;
    return F(n - 1) + F(n - 2);
}
```