Recurrences

Instructor: Dr. Tarunpreet Bhatia
Assistant Professor, CSED
Thapar Institute of Engineering and Technology

Recurrence Relation

- Many algorithms (such as divide and conquer) are recursive in nature.
- When an algorithm contains a recursive call to itself then its running time can be described by mathematical relation called recurrence relation or recurrence equation.
- Recurrence relations are used to determine the running time of a recursive program.

Recurrence Relation

- We get running time as a function of n (input size) and we get the running time on inputs of smaller sizes.
- A recurrence is a recursive description of a function, or a description of a function in terms of itself.
- A recurrence relation recursively defines a sequence where the next term is a function of the previous terms.

Recurrence Examples

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

Recurrence Example 1

```
void display (int n)
{
        if (n > 0)
        {
        printf ("%d", n);
        display (n -1);
        }
}
```

Recurrence Modified Example 1

```
void display (int n)
{
        if (n > 1)
        {
        printf ("%d", n);
        display (n -1);
        }
}
```

Recurrence Example 2

```
void loop _ test(int n)
{
        if (n > 1)
        {
                for(i = 0;i < n; i++)
                    printf ("%d", i);
        }
        loop _ test(n -1);
}
```

Recurrence Example 3

```
void loop _product(int n)
{
        if (n > 0)
        {
                for(i = 1;i <= n;i = i * 2)

                        printf ("%d", i);
        }
        loop _product(n -1);
}
```

Recurrence Example 4

```
void test (int n)
{
    if (n > 0)
    {
        printf("%d",n);
        test (n-1);
        test (n-1);
    }
}
```

Recurrence Example 5

```
void test (int n)
{
    if (n > 0)
    {
        printf(“%d”, n);
        test (n/2);
    }
}
```

Example: Binary search using recursion

```
int binarySearch(int arr[], int l, int r, int x)
{
        if (l<=r) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);
        return binarySearch(arr, mid + 1, r, x);
        }
        return -1;
}
```

Recurrence Example 6

```
void test (int n)
{
    if (n > 0)
    {
        printf("%d", n);
        test (n/2);
        test (n/2);
    }
}
```

Recurrence Example 7

```
void test (int n)
{
    if (n > 1)
    {
        test (2n/3);
        test (n/3);
    }
}
```

Recursion Example 8

```
int doSomething(n)
{
   if n < 2
      return n;
   return doSomething(n-1) + doSomething(n-2);
}
```

Solving Recurrences

- Iteration method or Back substitution
- Recursion Tree method
- Master method

Iteration Method

- In iteration method,
  - Expand the recurrence
  - Work some algebra to express as a summation
  - Evaluate the summation
- We will show several examples

Recurrence Example 1

$$T(n) = \begin{cases} 1 & n = 0 \\ 1 + T(n-1) & n > 0 \end{cases}$$

Recurrence Example 2

$$T(n)=\begin{cases} 1 & n=0 \\ n+T(n-1) & n>0 \end{cases}$$

Recurrence Example 3

$$T(n)=\begin{cases}1 & n=0\\ T(n-1)+\log n & n>0\end{cases}$$

Recurrence Example 4

$$T(n) = \begin{cases} 1 & n = 0 \\ 2T(n-1)+1 & n > 0 \end{cases}$$

Recurrence Example 5

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n/2) + 1 & n > 1 \end{cases}$$

Recurrence Example 6

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n/2) + 1 & n > 1 \end{cases}$$

Examples

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

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

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

$$T(n) = T(n-1) + \log n ---------- O(n \log n)$$

$$T(n) = T(n-2) + 1 \rightarrow \frac{n}{2} \rightarrow O(n)$$

$$T(n) = T(n-50) + n ---------- O(n^2)$$

Try this!

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n/2) + n & n > 1 \end{cases}$$

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T((n/2) + 16) + n & n > 1 \end{cases}$$

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n/3) + n^{\wedge}4/3 & n > 1 \end{cases}$$

Recurrence Relation for Root Function

void Test(int n)	————————	T(n)
{		
if(n > 2)	————————	1
{		
stmt;	————————	1
Test($\sqrt{n}$);	————————	T($\sqrt{n}$)
}		
}		

Recurrence Relation: $T(n) = T(\sqrt{n}) + 1$

$$T(n) = \begin{cases} 1 & n = 2 \\ T(\sqrt{n}) + 1 & n > 2 \end{cases}$$

$$T(n) = T(\sqrt{n}) + 1$$

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

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

$$T(n) = T(n^{\frac{1}{2^3}}) + 3$$

$$\vdots$$

$$T(n) = T(n^{\frac{1}{2^k}}) + k$$

Assume n = 2^m

$T(2^m) = T(2^{\frac{m}{2^k}}) + k$

Assume $T(2^{\frac{m}{2^k}}) = T(2^1)$

Therefore, $\frac{m}{2^k} = 1$

$m = 2^k$ and k = $\log_2 m$

Since $n = 2^m$, m = $\log_2 n$

k = $\log \log_2 n$

$\theta(\log \log_2 n)$

$$T(n) = 2T(\sqrt{n}) + \log n$$

Try this!!

$$T(n) = 2T(\sqrt{n}) + 1$$

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

$$\mathrm{T}(n) = \sqrt{n}\,\mathrm{T}(\sqrt{n}) + n$$

$$T(n) = 5T\left(\frac{n}{5}\right) + \frac{n}{\log n} \quad \textit{where } T(1) = 1$$