DAA

Design and Analysis of Algorithms

Book: Fundamentals of algorithmics

by Brassard and Bratley

Problem Statements

- Sorting
- Searching
- String processing
- Graph problems
- Combinatorial problems
- Geometric problems
- Numerical problems

Design technique for algorithms

- Iterative Algorithm
- Divide and Conquer
- Greedy Technique
- Dynamic Programming
- Branch and Bound
- Randomized Algorithm
- Backtracking Algorithm
- Approximation Algorithm

Analysis of Algorithms

- What is an efficient algorithm?
- Time complexity(T(n)) indicating how fast the algorithm runs.
- Space complexity indicating how much extra memory algorithm uses.

Time Complexity

- Time Complexity is most commonly estimated by counting the number of elementary steps performed by any algorithm to finish execution.

- Time complexity is defined by T(n).

  where n is input size

Time Complexity

- **Problem statement:** Perform sum of 1 to n numbers
- Methods:
  - Using Loop
  - Using Equation
  - Using Recursion

Time Complexity

- **Method-1: Using Loop**

  Algorithm Sum1(n)

```
{
  sum=0;
  for i=1 to n do
     sum=sum+i;
  return sum;
}
```

Time Complexity

- **Method-1: Using Loop**

```
Algorithm Sum1(n)
{
  sum=0;
  for i=1 to n do
     sum=sum+i;
  return sum;
}
```

$\mathbf{T_{sum1}(n) = O(n)}$

Time Complexity

- **Method-2: Using Equation**

```
Algorithm Sum2(n)
{
    return n*(n+1)/2;
}
```

Time Complexity

- **Method-2: Using Equation**

  Algorithm Sum2(n)

  {

  return n*(n+1)/2;

  }

$T_{sum2(n)} = O(1)$

Time Complexity

- **Method-3: Using Recursion**

```
Algorithm Sum3(n)
{
  if(n>0)
     return Sum3(n-1) + n;
  else
     return 0;
}
```

Time Complexity

- **Method-3: Using Recursion**

```
Algorithm Sum3(n)
{
  if(n>0)
     return Sum3(n-1) + n;
  else
     return 0;
}
```

If $n=0 \Rightarrow T(0) = 2$

If $n>0 \Rightarrow T(n) = 2 + T(n-1)$

$\mathbf{T_{sum3(n)} = ?}$

Solving Linear Recurrence Relations

Problem statement:
Find nth Fibonacci number.

Iterative Method:

Algorithm:

```
f1(n):
   first=0;
   second=1;
   for i = 3 to n
   {
      next = first + second;
      first = second;
      second = next;
   }
  return next;
```

Time complexity: O(1) + O(n) + O(1) = O(n)

Recursive Method:

Algorithm:

```
f2(n):
  if n <= 1
     return n
  return f2(n - 1) + f2(n - 2)
```

Time complexity: ?

If n=0 => F(0) = 0

n=1 => F(1) = 1

n>1 => F(n) = F(n-1) + F(n-2)

Recurrence Relation

- A recurrence is an equation or inequality that describes a function in terms of its values on smaller inputs.
- Running time of recursive algorithms can be obtained by solving recurrence.

Solving Recurrence Relation

- Methods for solving recurrences relation:
  - **Linear homogeneous recurrence relation**
  - **Linear non-homogeneous recurrence relation**
  - **Substitution Method**
  - **Change of variable Method**
  - **Iteration Method**
  - **Master Method**
  - **Recurrence Tree Method**

Solving Recurrence Relation

- Types of Recurrence Relation:
  - Linear Recurrence
  - Non-Linear Recurrence

Linear recurrences

- Linear recurrence: Each term of a sequence is a linear function of earlier terms in the sequence.

$$t_n = t_{n-1} + t_{n-2}$$

Linear recurrences

- Linear recurrences

1. Linear homogeneous recurrences
2. Linear non-homogeneous recurrences

Linear homogeneous recurrences

- General form of Homogenous recurrence relation

$$a_0 t_n + a_1 t_{n-1} + \ldots\ldots\ldots + a_k t_{n-k} = 0$$

- $Here,$
- t_i : values we are looking for(1<=i<=k)
- $a_i : co-efficients\ are\ constants$
- This recurrence is

✓ Linear: because it does not contain any square terms and so on.

✓ homogeneous because the linear combination of the t_{n-i} is equal to zero.

✓ t_n is expressed in terms of the previous k terms of the sequence, so its degree is k.

Example

Determine if the following recurrence relations are linear homogeneous recurrence relations with constant coefficients.

- $P_n = (1.11)P_{n-1}$
  a linear homogeneous recurrence relation of degree one
- $a_n = a_{n-1} + a^2_{n-2}$
  not linear
- $f_n = f_{n-1} + f_{n-2}$
  a linear homogeneous recurrence relation of degree two
- $H_n = 2H_{n-1}+1$
  not homogeneous
- $a_n = a_{n-6}$
  a linear homogeneous recurrence relation of degree six
- $B_n = nB_{n-1}$
  does not have constant coefficient

Steps

1. Compare your recurrence relation with homogenous recurrence relation. $a_0 t_n + a_1 t_{n-1} + \dots\dots\dots + a_k t_{n-k} = 0$
2. Find the value of k and coefficients $(a_0, a_1, .. a_k)$
3. Represent the recurrence in following form: Characteristic equation

$$a_0 x^k + a_1 x^{k-1} + \dots\dots\dots + a_k = 0$$

Where x is unknown and constant

4. Solve characteristic equation and find roots. We will have k roots.
5. For roots, we have two possibilities.

For roots, we have two possibilities.

Roots

If all k roots are distinct

$$t_n = \sum_{i=1}^{k} c_i r_i^n$$

Where c_i - constants and r_i - roots

Ex: If roots are 1,2,3 then

$$t_n = c_1 1^n + c_2 2^n + c_3 3^n$$

If roots are not distinct

- If equation has k roots and if root r_i Has multiplicity m_i
- Ex:1,2,2,3,3,3,3
  Root 2 has multiplicity 2
  Root 3 has multiplicity 4

$$t_n = \sum_{i=1}^{l} \sum_{j=0}^{mi-1} c_{ij} n^j r_i^n$$

$$t_n = c_1 1^n + c_2 2^n + c_3 n 2^n + c_4 3^n + c_5 n 3^n$$
$$+ c_6 n^2 3^n + c_6 n^3 3^n$$

Steps

6. Find the value of constants c1,c2…ck using initial conditions.
7. Put the value of c1,c2…ck in t(n).

Problem statement:
Find n^{th} Fibonacci number.

Iterative Method:

Algorithm:

```
f1(n):
    first=0;
    second=1;
    for i = 3 to n
    {
        next = first + second;
    }
   return next;
```

Time complexity: O(1) +O(n)+O(1) = O(n)

Recursive Method:

Algorithm:

```
f2(n):
   if n <= 1
      return n
   return f2(n - 1) + f2(n - 2)
```

Time complexity: ?

If n=0 => F(0) = 0

n=1 => F(1) = 1

n>1 => F(n) = F(n-1) + F(n-2)

What is the solution of the recurrence relation fn = fn-1 + fn-2 with f0=0 and f1=1?

Solution:

Rewrite recurrence, $f_n - f_{n-1} - f_{n-2} = 0$

Compare recurrence with $a_0 t_n + a_1 t_{n-1} + \dots + a_k t_{n-k} = 0$

$K=2$, $a_0=1$, $a_1=-1$, $a_2=-1$

Characteristic eq. is $x^2 - x - 1 = 0$

Possible roots are $r1 = (1+\sqrt{5})/2$, $r2 = (1-\sqrt{5})/2$

General Solution is $fn = c1r1^n + c2r2^n$

$= c1\,((1+\sqrt{5})/2)^n + c2((1-\sqrt{5})/2)^n$

What is the solution of the recurrence relation $f_n = f_{n-1} + f_{n-2}$ with $f_0=0$ and $f_1=1$?

Use initial condition to find c1 and c2

$n=0 \Rightarrow f_0 = c_1 + c_2 \quad \Rightarrow 0 = c_1 + c_2 \quad \Rightarrow c_1=-c_2$

$n=1 \Rightarrow f_1 = c_1((1+\sqrt{5})/2) + c_2((1-\sqrt{5})/2) \quad \Rightarrow 1 = c_1((1+\sqrt{5})/2) + c_2((1-\sqrt{5})/2)$

Solving above equations, $c_1=1/\sqrt{5}$, $c_2=-1/\sqrt{5}$

Thus, $f_n = (1/\sqrt{5})\,((1+\sqrt{5})/2)^n - (1/\sqrt{5})\,((1-\sqrt{5})/2)^n$

$f_n = O(1.6^n) = O(2^n)$

Example-2

Solve the recurrence:

$$t_n = \begin{cases} 0 & , \text{if } n = 0 \\ 5 & , \text{if } n = 1 \\ 3t_{n-1} + 4t_{n-2} & , \text{otherwise} \end{cases}$$

Example-3

Solve the recurrence:

$$t_n = \begin{cases} n & \text{, if } n = 0, 1, 2 \\ 5t_{n-1} - 8t_{n-2} + 4t_{n-3} & \text{, otherwise} \end{cases}$$

Inhomogeneous Recurrence

- Inhomogeneous recurrence means Linear recurrence relation is not equal to zero.
- General Form:

$$a_0 t_n + a_1 t_{n-1} + \ldots\ldots\ldots + a_k t_{n-k} = b^n p(n)$$

Where,

a_i = Constant co–efficient

b = Constant

p(n) = Polynomial term with degree(d)

- Example: $t_n - 3t_{n-1} + 2t_{n-2} = 12^n$

Steps

- Compare recurrence with general form of inhomogeneous recurrence:

$$a_0 t_n + a_1 t_{n-1} + \ldots\ldots.+ a_k t_{n-k} = b^n p(n)$$

- Find value of k, co-efficients$(a_0, a_1, \ldots a_k)$, b and p(n)
- Represent recurrence in characteristic equation:

$$(a_0 x^k + a_1 x^{k-1} + \ldots\ldots.+ a_k)\ (x-b)^{d+1} = 0$$

Here d is degree of polynomial term.

Steps

Finding k, co-efficients, b and p(n) from following recurrence:

1) $t_n - 3t_{n-1} + 2t_{n-2} = 12^n$

- Here k = 2, a0 = 1, a1 = -3, a2 = 2, b =12, p(n) = 1, d=0

2) $\mathbf{t_n - 3t_{n-1} + 2t_{n-2} = n^2 + 12}$

- Here k = 2, a0 = 1, a1 = -3, a2 = 2, b = 1, p(n) = $n^2 + 12$, d=2

Steps

- Solve characteristic equation and find roots.
- For roots we have two possibilities:

Roots

If all k roots are distinct

$$t_n = \sum_{i=1}^{k} c_i r_i^n$$

Where c_i - constants and r_i - roots

Ex: If roots are 1,2,3 then

$$t_n = c_1 1^n + c_2 2^n + c_3 3^n$$

If roots are not distinct

- If equation has k roots and if root r_i Has multiplicity m_i
- Ex:1,2,2,3,3,3,3
  Root 2 has multiplicity 2
  Root 3 has multiplicity 4

$$t_n = \sum_{i=1}^{l} \sum_{j=0}^{m_i - 1} c_{ij} n^j r_i^n$$

$$t_n = c_1 1^n + c_2 2^n + c_3 n 2^n + c_4 3^n + c_5 n 3^n + c_6 n^2 3^n + c_6 n^3 3^n$$

Steps

- Use initial condition and Find the value of constants c1,c2…ck.
- Put the value of c1,c2…ck in general solution t(n).

Example-4

Solve the recurrence:

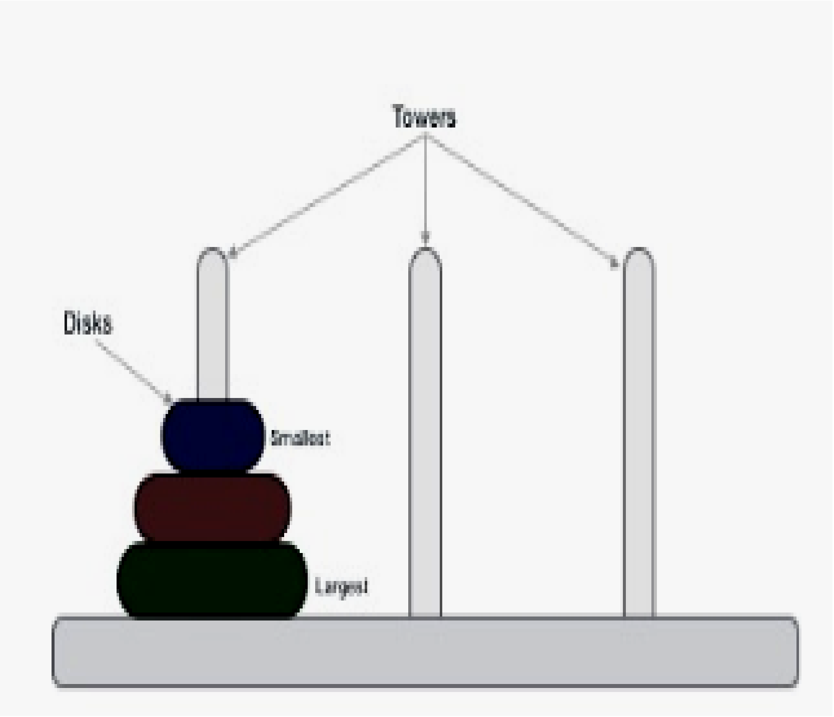
$$t_n - 2t_{n-1} = 3^n$$

Example-5

Solve the recurrence:

$$t_n = 2t_{n-1} + n, \ n>0$$

Tower of Hanoi




```
Toh(n,s,aux,d)
{
if(n==1)
        move disk from s to d
 else
        Toh(n-1, s, d, aux)
        move disk from s to d
        Toh(n-1,aux, s, d)
}
```

Example-6

Solve the recurrence of tower of hanoi:

$$t(m) = \begin{cases} 0 & , \text{if } m = 0 \\ 2t(m-1) + 1 & , \text{otherwise} \end{cases}$$

Example-7

Solve the recurrence:

$$\mathbf{t(n)} = \begin{cases} \mathbf{0} & \mathbf{, if\ n = 0} \\ \mathbf{2t(n-1) + n + 2^n} & \mathbf{, otherwise} \end{cases}$$

Change of Variable

- T(n) – General recurrence
- Replace n with 2^i
- $T(n) = T(2^i) = t_i$

Example:

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

Here, $T(n) = T(2^i) = t_i$

$T(n/2) = T(2^i/2) = T(2^{i-1}) = t_{i-1}$

Example-8

Solve the recurrence:

$$T(n) = \begin{cases} 1 & , \text{if } n = 1 \\ 3T(n/2) + n \ , & \text{if } n \text{ is power of } 2, n > 1 \end{cases}$$

Example-9

Consider recurrence:

$T(n) = 4T(n/2) + n^2$, when n is power of 2, $n \geq 2$

Example-10

Consider following recurrence and solve with change of variable method.

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