Specific Sequences of given length

DAA ASSIGNMENT-4 , GROUP 8

Swaraj Bhosle
IIT2019024

Ritesh Raj
IIT2019025

Utkarsh Garg
IIT2019026

Abstract: In this Paper we have devised an algorithm to find the number of possible sequences of specific length (let say n) such that each of the next element is greater than or equal to twice of the previous element but less than or equal to specific number (let say m).

I. INTRODUCTION

In this report, we are going to discuss about the particular sequence of given length. We are given two integers m and n. We have to find the number of possible sequences of length n such that each of the next element is greater than or equal to twice of the previous element but less than or equal to m. There should be n elements and value of last element should be at-most m. we are dividing the problem into sub problems and again dividing that sub problem in sub problems.basically we are using Divide and Conquer method.

This report further contains -

II Algorithm Design.
III Algorithm And Analysis.
IV Time Calculation
V Time Complexity
VI Conclusion
VII References

II. ALGORITHM DESIGN

We are given two integers as input i.e. M and N. where N is the length of sequence and M is the number where greatest number in the sequence is less than or equal to M. if M is present in the sequence then it must be present at the last element of sequence. As per the given condition the n-th value of the sequence can be at most M.

Now there are two possible cases for the n-th element:

a If the last element i.e. n-th element is m, then the $(N-1)th$ element is at most $M/2$. So We recur for $M/2$ and $(N-1)$.
b If the last element is not m, then the last element is at most m-1. We recur for $(M-1)$ and N.

The total number of sequences is the sum of the number of sequences including m and the number of sequences where m is not included.
Thus the original problem of finding number of sequences of length N with max value M can be subdivided into independent sub problems of finding number of sequences of length N with max value M-1 and number of sequences of length N-1 with max value M/2.

Step 1 : take 2 numbers M and N as input from user .N is length of sequence and no other element in the sequence is greater than M . if M is present in the sequence then it is present only at the last position.

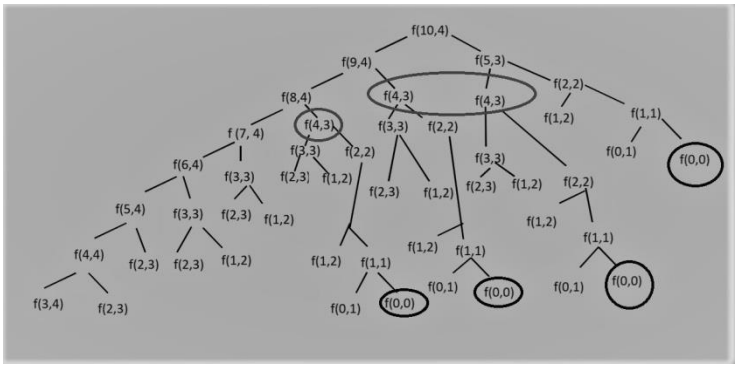
Step 2 : call the function:
TotalNumberOfSequence(M, N)

Step 3 : in the function, check whether M is greater than N or not. If m is smaller than N then return 0, as n is more than the maximum value M.
If N is 0, then the sequence is empty, Return 1

Step 4 : now there are 2 possibilities :

a Reduce last element value i.e. call the function TotalNumberOfSequence($M-1$, n)
b Consider last element as M and reduce number of terms i.e. call the function TotalNumberOfSequence($M/2$, $N-1$)

Step 5 : print the total no of sequences



Recursive Tree for $M = 10$ and $N = 4$

III. ALGORITHM AND ANALYSIS

Algorithm 1: Algorithm 1: Recursive Method

Input: two numbers M and N
Output: Total number of possible sequences

```
1  Function TotalSequences(int m, int n):
2      if M < N then
3          return 0
4
5      if N == 0 then
6          return 1
7
8      return TotalSequences (M − 1, N) +
           TotalSequences (M/2, N − 1)
9  Function Main():
10     Get M;
11     Get N;
12     Print (TotalSequences(M, N))
```

Algorithm 2: Algorithm 2: Dynamic Programming

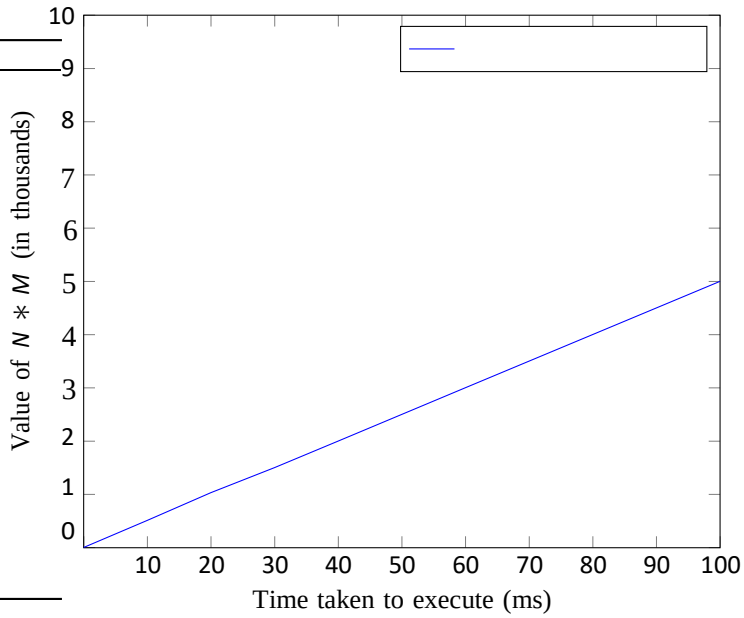
Input: two numbers M and N
Output: Total number of possible sequences

```
1  Function TotalSequences(int M, int N):
2      int DP[M+1][N+1];
3      for i ← 0 to M do
4          for j ← 0 to N do
5              if (i==0 || j==0);
6              then DP[i][j] = 0;
7
8              else if i < j;
9              then DP[i][j] = 0;
10
11             else if j == 1;
12             then DP[i][j] = i;
13
14             else
15             DP[i][j] = DP[i-1][j] + DP[i/2][j-1];
16     return DP[M][N];
17 Function Main():
18     Get M;
19     Get N;
20     Print (TotalSequences(M, N))
```

IV. TIME CALCULATION

The total number of sequences is the sum of the number of sequences including m and the number of sequences where m is not included. Thus the original problem of finding number of sequences of length N with max value M can be subdivided into independent sub-problems of finding number of sequences of length n with max value $M - 1$ and number of sequences of length $N - 1$ with max value $M/2$.

Complexity analysis for Dynamic programming approach



V. TIME COMPLEXITY

BEST	AVERAGE	WORST CASE
$O(M * N)$	$O(M * N)$	$O(M * N)$

VI. CONCLUSION

The above problem can be solved by using Dynamic Programming also. but the most efficient way is the divide and conquer. We can conclude from our model that the running time of our algorithm is O(M*N). We have tested our Algorithm Theoretically and experimentally and both yielded simi 3 results. the above problem can be solved by using Dynamic Programming also. but the most efficient way is the divide and conquer.

VII. REFERENCES

1) https://www.geeksforgeeks.org/sequences-given-length-every-element-equal-twice-previous/
2) https://tutorialspoint.dev/algorithm/dynamic-programming-algorithms/sequences-given-length-every-element-equal-twice-previous