Sequences of given length where every element is more than or equal to twice of previous.

DAA ASSIGNMENT-4 , GROUP 8

Hitika Rajesh kumar
IIB2019021

Gaurav yadav
IIB2019022

Aditya raj
IIB2019023

***Abstract*—Given two integers m n, 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.**

I. Introduction

For finding 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 can be solved using recursion. Wherever we see a recursive solution that has repeated calls for the same inputs, we can optimize it using Dynamic Programming. Simple optimization reduces time complexities from exponential to polynomial.

II. Algorithmic Design

A. *Approach 1(By Recursion)*

1) Take input of integers n and m in variables n and m respectively.
2) We will call function with parameters as n and m which will compute the answer.
3) Used function is using a recursive approach to find the answer.
4) Base cases for this function is when m is less than n then we will return zero and also when n will become zero then we will return
5) We will call the function : return fun(m-1, n) + fun(m/2, n-1);

Algorithm 1

Input: N(length of sequence) and M(upper bound on sequence elements)

Output: 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.

Method:

1) Call the solve function with parameters as n and m: solve(n,m)
2) if (m $<$ n) We will return 0
3) if (n == 0) We will return 1
4) We will recur the function fun(m-1, n) + fun(m/2, n-1)
5) And return the value as ans.

B. *Approach 2(By Dynamic Programming)*

1) Take input of integers n and m in variables n and m respectively.
2) Initializing the array dp[m][n] as zero.
3) Then from i=1 to i¡=m initialising dp[i][1]=1 as there will be only one way to create a sequence of size n(where n=1 ,this is the base case),where the sequence is ending at i.
4) Prefix sum algorithm is being used in this question here and precomputation technique
5) After that the prefix sum technique is being applied then for the given n at last the sum from i=1 to i¡=m is being done compute the final answer
6) printing the ans i.e the no. of total ways to form a sequence of length n from 1 to m.

Algorithm 2

Input: N(length of sequence) and M(upper bound on sequence elements)

Output: 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.

Method:

Initialize a 2-D array dp[m+1][n+1]

```
for i=0 to i=m
      for j=0 to j=n
            dp[i][j] = 0;

for i=1 to i=m
      Initialize dp[i][1] = 1;

for i = 2 to i=n
      for j = 1 to j=m
            dp[j * 2][i] += dp[j][i - 1];

        for j = 1 to j=m
            dp[j][i] += dp[j - 1][i];
```

We will declare a variable ans and initialize it to 0

```
for i = 1 to i = m
      ans += dp[i][n];
```

Finally we will return the ans as output.

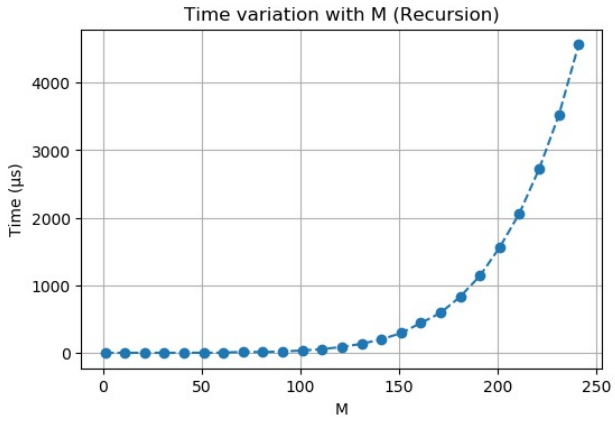
III. Algorithm Analysis

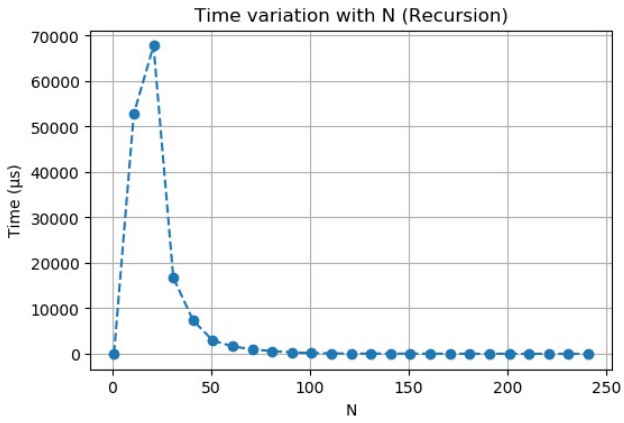
Algorithm 1(By Recursion)
Time Complexity Analysis:
Worst case : O(m^n)
Best case : $\Omega(1)$ when $n = 1$
Space Complexity Analysis:
Space complexity is O(1)

Algorithm 2(By Dynamic Programming)
Time Complexity Analysis:
O(N*M), because the states for the problem are the length of the sequence and the maximum number which can be considered. Thus the time complexity is polynomial.
Space Complexity Analysis:
O(N*M), because we have created a 2D. DP table to store the intermediate results. The space complexity is also polynomial.

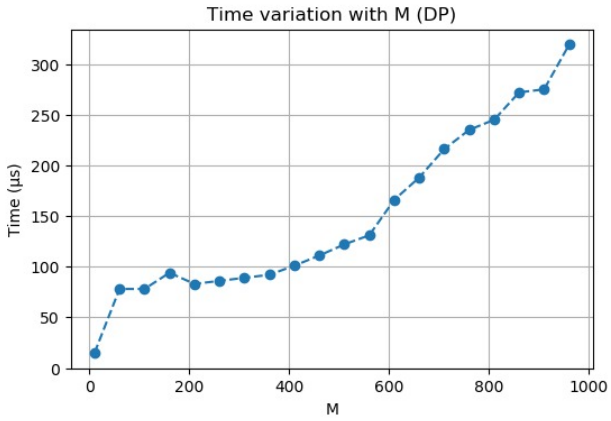
IV. Experimental study

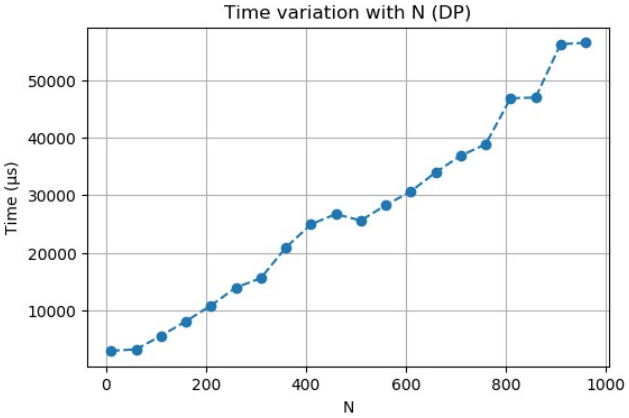
A. Approach 1(By recursion)





B. Approach 2(By Dynamic Programming)





V. Conclusion

Above two methods have different time complexities and meet to fulfill the problem statement. The order in which they are good can be listed as:
I. Approach 2 which is the DP in tabular form
II. Approach 1 which is the recursive approach
Based on the time complexities.

VI. References

1).https://www.geeksforgeeks.org/sequences-given-length-every-element-equal-twice-previous/
2).https://www.tutorialcup.com/interview/dynamic-programming/sequences-of-given-length-where-every-element-is-more-than-or-equal-to-twice-of-previous.htm