Sequence of Sequence

> 给定整数m和n，找到长度为n的序列且序列中后一个元素是前一个元素的2倍或者2倍以上，并且序列最后一个元素需要小于等于m

递归

原问题相当于是找到n长度的序列，最后元素小于等于m，可以分解为该序列包含m和不包含m的两个问题：

1. 包含m，问题转换为n - 1个元素，最后元素小于等于 m / 2
2. 不包含m，问题转换为n个元素最后元素小于等于m - 1

```
#include<bits/stdc++.h>

using namespace std;

int seqOfSeq(int n, int m) {
    if (m < n) return 0;
    if (n == 0) return 1;
    return seqOfSeq(n - 1, m / 2) + seqOfSeq(n, m - 1);
}
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int n, m;
        scanf("%d %d", &m, &n);
        printf("%d\n", seqOfSeq(n, m));
    }
    return 0;
}
```

DP

避免递归的重复计算

```
#include<bits/stdc++.h>

using namespace std;

int seqOfSeq(int n, int m) {
    vector<vector<int>> dp(m + 1, vector<int>(n + 1));
```

```
    for (int i = 0; i < m + 1; ++i) {
        for (int j = 0; j < n + 1; ++j) {
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (j > i)
                dp[i][j] = 0;
            else if (j == 1)
                dp[i][j] = i;
            else
                dp[i][j] = dp[i - 1][j] + dp[i / 2][j - 1];
        }
    }
    return dp[m][n];
}
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int n, m;
        scanf("%d %d", &m, &n);
        printf("%d\n", seqOfSeq(n, m));
    }
    return 0;
}
```