

提高组300分试题 第二组题解

繁繁的数字

题解

首先因为只有 $\log(n)$ 个不同的数，因此可以采用一个 $O(n\log(n))$ 的dp来求解。考虑优化这个dp。首先存在以下状态转移：

$$f(n) = f(n-1) \quad (n \text{ 为奇数})$$

如果 n 为偶数，我们按照划分中是否有1，来把答案分为2类：

- 1、划分方案中包括1。 ($g(n, 0)$)
- 2、划分方案中不包括1。 ($g(n, 1)$)

对于第1类， $g(n, 0) = f(n-1)$ 。即所有 $f(n-1)$ 的方案加1个1。

对于第2类， $g(n, 1) = f(n/2)$ ，即将所有 $f(n/2)$ 的方案，每个数乘2。

因此最终偶数的状态转移方程为：

$F(n) = F(n-1) + F(n/2)$ ，这样就得到了一个 $O(n)$ 的DP。本题还有一个 $O(\log(n)^3)$ 的DP解法，但在这里不展开了。

标准代码

C++

```
#include <iostream>
#include <vector>
using namespace std;

const int mod = 1000000007;

int main() {
    int n;
    cin >> n;
    vector<int> help(n + 1);
    help[1] = 1;

    for (int i = 2; i <= n; i++) {
        if (i & 1) {
            help[i] = help[i-1];
        }
    }
}
```

```

    }
    else {
        help[i] = (help[i>>1] + help[i-1]) % mod;
    }
}

cout << help[n] << endl;
return 0;
}

```

繁繁的游戏

题解

图中最短路长度 $\times d$ 就是答案。这个最短路是指所有点之间最短路的最大值。那么安排这条路径上每个人的分数都相差 d 即可。

标准代码

C++

```

#include<bits/stdc++.h>

using namespace std;
const int oo=1000000;
int m[55][55];
bool vis[55];
int n;

int maxDifference(vector <string> a, int d)
{
    n=a.size();
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            if(a[i][j]=='Y')
                m[i][j]=1;
            else m[i][j]=oo;
    for(int k=0;k<n;k++)
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                m[i][j]=min(m[i][j],m[i][k]+m[k][j]);
    int ans=0;
    for(int i=0;i<n;i++)
        for(int j=i+1;j<n;j++)
            ans=max(ans,m[i][j]);
    if(ans==oo)return -1;
    return ans*d;
}

```

```

int main()
{
    int casT;
    cin>>casT;
    while(casT--){
        vector<string> ST;
        string tmp;
        int n,d;
        cin>>n>>d;
        while(n--){
            cin>>tmp;
            ST.push_back(tmp);
        }
        cout<<maxDifference(ST,d)<<endl;
    }
    return 0;
}

```

繁繁的队列

题解

换一个角度来描述题意简化思考，本题实际相当于从原来的数组中挑一些放到头部或尾部，留下一些保持不变，让留下的元素尽量多。保持不变的部分一定要同最终排序好的数组的一部分，完全匹配。由于最终排序好的数组是严格的 $1 - n$ 。所以保持不变的部分一定是 $i - j$ ，共 $j - i + 1$ 个数。同时数字的顺序也要保留，因此 $i + 1$ 一定在 i 的后面， j 一定在 $j - 1$ 的后面...，我们称之为数字连续的序列。用 $F(i)$ 表示以数字 i 结尾最长的数字连续序列的长度， P_i 表示数字 i 所在的位置，所以存在以下递推：

$F(i) = F(i - 1) + 1$ ($P_i > P_{i - 1}$)，预先处理好每个数的位置，直接dp即可。

标准代码

C++

```

#include<stdio.h>
#define MAX 50001
int a[MAX];
int main()
{
    int n,max=0,m;
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        scanf("%d",&m);
    }
}

```

```
        a[m]=a[m-1]+1;
        if(a[m]>max)
            max = a[m];
    }
    printf("%d",n-max);
}
```