第八场赛后讲解

Robert_JYH

洛谷
www.luogu.com.cn

T1 得分

得分

小凯在参加一场竞赛，目前他的基础得分为 x 分，低于其它许多选手。

好在他还有 5 张功能牌，分为加法功能牌和乘法功能牌两种，每张功能牌我们用一个符号$c \in \{+,*\}$ 和一个能力值 val 表示。其中加法功能牌可以让小凯的得分在当前的基础上增加 val 分，乘法功能牌可以让小凯的得分变为原来的 val 倍。

功能牌会按顺序触发。例如，若小凯初始得分为 3 分，5 张功能牌按顺序依次为 $(+\ 3), (+\ 2), (*\ 2), (+\ 5), (*\ 2)$，则小凯最终的得分为 $\Big(\big((3+3+2)\times 2\big)+5\Big)\times 2 = 42$ 分。

功能牌的使用顺序会影响最终的得分。现在，小凯会告诉你他的初始得分和拥有的功能牌，请你任意安排功能牌的使用顺序，告诉他可能的最大得分。

样例

【样例 1 输入】

```
3
+ 3
+ 2
* 2
+ 5
* 2
```

【样例 1 输出】

```
52
```

子任务

对于前 20% 的数据，输入的 5 张功能牌完全相同。

对于前 40% 的数据，有 $c_i = +$，即所有功能牌均为加法功能牌。

对于前 70% 的数据，若 $c_i = \times$，额外保证 $val_i \leq 3$。

对于 100% 的数据，有 $1 \leq x, val_i \leq 10^6$，$c_i \in \{+, *\}$。若 $c_i = \times$，额外保证 $val_i \leq 100$。

子任务1/2

对于前 20% 的数据，输入的 5 张功能牌完全相同。

对于前 40% 的数据，有 $c_i = +$，即所有功能牌均为加法功能牌。

不需要考虑顺序，直接按题意模拟输出即可。

【样例 2 输入】

```
10
+ 10
+ 40
+ 30
+ 20
+ 10
```

【样例 2 输出】

```
120
```

子任务3/正解

对于前 70% 的数据，若 $c_i = \times$，额外保证 $val_i \leq 3$。

对于 100% 的数据，有 $1 \leq x, val_i \leq 10^6$，$c_i \in \{+, *\}$。若 $c_i = \times$，额外保证 $val_i \leq 100$。

考虑到，加法功能牌一定放在乘法功能牌前面，而同类型功能牌内部不需要考虑顺序，因为只需要用两个变量分别保存两种功能牌的累计能力值（加法用加，乘法用乘），最后相乘输出即可。

如果没有开long long，则只能通过子任务3，获得70分。

本题由于牌的数量很少，因此也可以搜索出所有方案，然后统计答案。

参考代码

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll sum,plu = 1;
int main(){
    cin>>sum;
    for(int i=1;i<=5;i++){
        char ch=getchar();
        while(ch!='+'&&ch!='*')ch=getchar();
        ll x;
        cin>>x;
        if(ch=='+')sum+=x;
        else plu*=x;
    }
    cout<<sum*plu<<"\n";
    return 0;
}
```

T2 划分

划分

小凯有一个长度为 n 的排列。具体地，排列是指一个长度为 n 的序列，其中 1 到 n 的每个正整数均会出现且只会出现一次。

小凯想把这个排列变成金明给他的另一个长度为 n 的排列。

小凯可以把初始排列任意划分为连续的若干段，但每一次划分将耗费小凯 1 个单位的体力。

接着小凯可以把划分好的连续段按照任意顺序拼接起来，拼接操作不消耗体力。

显然，小凯一定可以把这个排列变成金明给他的排列，因为他可以把初始排列划分成 n 段然后再按金明给他的排列的顺序拼接起来。

但小凯不希望自己太累，他想请你告诉他，他需要消耗的最少体力是多少，也即他最少需要进行多少次划分操作。

样例

【样例 1 输入】

```
5
3 4 1 2 5
1 2 3 4 5
```

【样例 1 输出】

```
2
```

子任务

对于 20% 的数据，有 $n \leq 3$。

对于 50% 的数据，有 $n \leq 8$。

对于 70% 的数据，有 $n \leq 3000$。

对于 100% 的数据，有 $1 \leq n \leq 10^6, 1 \leq s_i, t_j \leq n$，输入数据为合法的排列。

子任务1/2

对于 20% 的数据，有 $n \leq 3$。

对于 50% 的数据，有 $n \leq 8$。

当$n \leq 3$时，分类讨论/打表即可。

如何打表？我们可以假设输入均为1、2、3，枚举所有可能的输出打表。然后在实际求解时通过映射关系转换。

当$n \leq 8$时，搜索出所有可能的解即可。

子任务3

对于 70% 的数据，有 $n \leq 3000$。

我们可以枚举所有数字在目标串中出现位置，如果有某个连续子串在目标串中也以连续的顺序不变的形式出现，那么这个子串我们可以整体处理，只产生一次贡献。

【样例 1 输入】

```
5
3 4 1 2 5
1 2 3 4 5
```

【样例 1 输出】

```
2
```

子任务3

对于 70% 的数据，有 $n \leq 3000$。

我们可以枚举所有数字在目标串中出现位置，如果有某个连续子串在目标串中也以连续的顺序不变的形式出现，那么这个子串我们可以整体处理，只产生一次贡献。

因此我们可以对于原串中每个数字$a[i]$，$O(n)$地寻找它在目标串中出现的位置$b[j]$，并判断$a[i-1]$与$b[j-1]$是否相等，若相等，则数字$a[i]$不产生贡献。

总时间复杂度$O(n^2)$。

正解

对于 100% 的数据，有 $1 \leq n \leq 10^6, 1 \leq s_i, t_j \leq n$，输入数据为合法的排列。

整体思路与子任务3相同，但我们不必对于一个数字，$O(n)$地寻找它在另一个串中出现的位置。

我们可以借助桶的思想，在输入原串时保存每个数字在原串中前一位置的数字的数值，在输入目标串时即可直接判断，无需查找。

总时间复杂度$O(n)$。

参考代码

```
int n, ans, a[N];
int main(){
    scanf("%d", &n);
    int pre = 0;
    for (int i = 0; i < n; i++){
        int x;
        scanf("%d", &x);
        a[x] = pre;
        pre = x;
    }
    pre = 0;
    for (int i = 0; i < n; i++){
        int x;
        scanf("%d", &x);
        if (i > 0 && a[x] != pre)
            ans++;
        pre = x;
    }
    printf("%d\n", ans);
    return 0;
}
```

T3 购物

购物

小凯参加了一个购物团，这个团里有 n 个人一起去购物，他们的总购物资金为 m 元，现在要给每个人分配个人购物资金。

他们有一个奇怪的要求：要求前 k 个人每个人的购物资金为 a 的倍数，后 $n-k$ 个人每个人的购物资金为 b 的倍数，且每人至少分得一块钱。所有的资金都要完全分配，求资金分配的方案数。两个方案不同，当且仅当至少存在一个人分得的钱数目不同。

由于资金分配方案数可能较大，你只需要输出方案数对 1,000,000,007 取模后的结果。

样例

【样例 1 输入】

```
10 20
8 1 4
```

【样例 1 输出】

```
332
```

子任务

对于 30% 的数据，有 $n \leq 10, m \leq 20$。

另有 30% 的数据，有 $a = b = 1$。

另有 10% 的数据，有 $k = 0$。

对于 100% 的数据，有 $1 \leq n \leq 2000, 1 \leq a, b \leq 10, 1 \leq m \leq 4000, 0 \leq k \leq n$。

子任务1

对于 30% 的数据，有 $n \leq 10, m \leq 20$。

直接暴力搜索所有分配方案再判断是否合法即可。

需要通过控制枚举上限等方式适当剪枝。

子任务2

另有 30% 的数据，有 $a = b = 1$。

这时候无需考虑分组与倍数的限制，问题转化为把 m 块钱分配给 n个人，且每个人分得钱非空的方案数。

应用隔板法，我们得到$ans = C_{m-1}^{n-1}$。时间复杂度与求组合数的时间相关，若使用杨辉三角，复杂度为$O(n^2)$。

杨辉三角预处理组合数参考代码

```
int c[N][N];
void getC(){
    for(int i=0;i<=m;i++)c[i][0]=1;
    for(int i=1;i<=m;i++)
        for(int j=1;j<=i;j++)
            c[i][j]=(c[i-1][j]+c[i-1][j-1])%mod;
}
int C(int a,int b){
    if(a<0||b<0) return 1;
    return c[a][b];
}
```

子任务3

另有 10% 的数据，有 $k = 0$。

这时候无需考虑分组的限制。

当m不是b的倍数时，无解。

当m是b的倍数时，我们可以考虑枚举每个人的钱是b的几倍，我们发现，如果将m除以b，此时的情形和子任务2完全相同。

应用隔板法，我们得到$ans = C_{m/b-1}^{n-1}$。

时间复杂度为$O(n^2)$。

参考代码

```
if(k==0){
    if(m%b!=0)puts("0");
    else cout<<C(m/b-1,n-1)<<endl;
    return 0;
}
else if(k==n){
    if(m%a!=0)puts("0");
    else cout<<C(m/a-1,n-1)<<endl;
    return 0;
}
```

正解

对于 100% 的数据，有 $1 \leq n \leq 2000, 1 \leq a, b \leq 10, 1 \leq m \leq 4000, 0 \leq k \leq n$。

这时候需要枚举给前k个人a的几倍，并判断剩余的钱是否是b的倍数。

若合法，则分别对两组应用隔板法，并将答案相乘。具体参见答案。

时间复杂度为$O(n^2)$。

参考代码

前面需加上之前边界情况的特判。

```
q=n-k;
for(int l=k;l*a<=m;l++){
    int r=m-l*a;
    if(r%b!=0)continue;
    r/=b;
    if(r<q)break;
    ans=(ans+1ll*C(l-1,k-1)*C(r-1,q-1)%mod)%mod;
}
```

扩展：裴蜀定理

裴蜀定理

对任何整数 a,b，关于未知数 x 和 y 的线性不定方程（称为裴蜀等式）：

$$ax+by=c$$

方程有整数解当且仅当 c 是 $\gcd(a,b)$ 的倍数，即 $\gcd(a,b)|c$ 。

裴蜀等式有解时必然有无穷多个解。

一定存在 x,y 满足 $ax+by=\gcd(a,b)$ 。

推论：a,b 互素等价于 $ax+by=1$ 有解。

T4 地铁换乘

地铁换乘

小凯所在的城市的地铁网是一张 n 个点 m 条边的无向连通图，他要乘坐地铁从学校附近的 1 号站点到达家附近的 n 号站点。

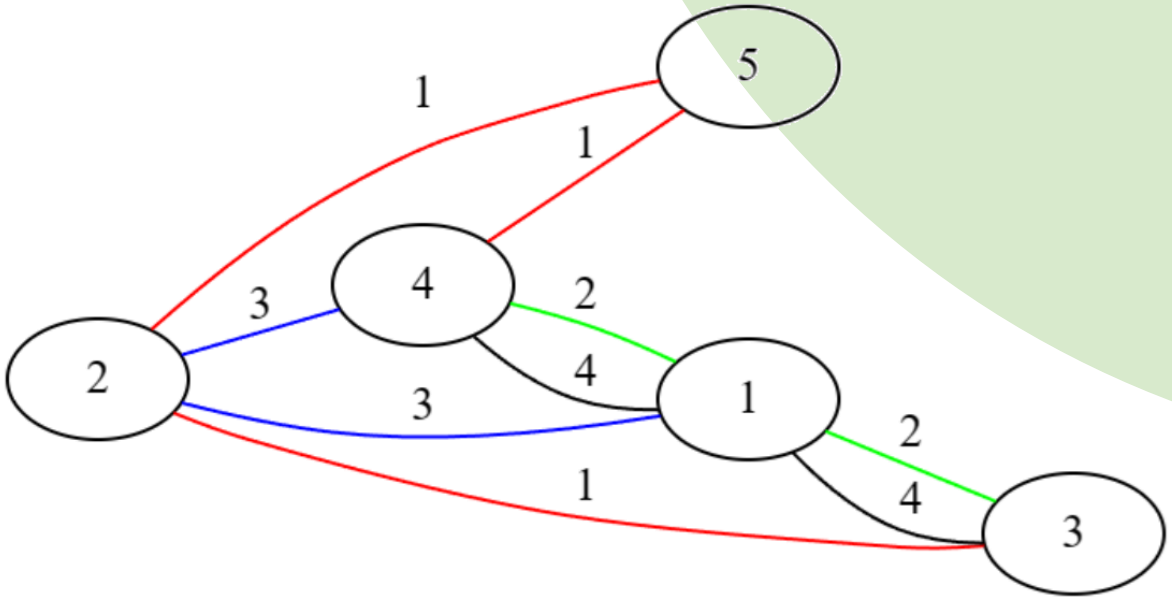
地铁共有 k 条线路，每条边只属于一条地铁线路，但**相邻两个站点之间可能存在多条边**，因而相邻两个站点可能可以乘坐多条线到达。每条线路上的点**互相之间可以仅通过该线路到达**。

每条地铁线采用通票制度，票价为 c_i 元。即当小凯花 c_i 元买了 i 号线的通票后，小凯乘坐 i 号线不再需要付费。即便他离开 i 号线乘坐其它线路后再次乘坐 i 号线，他也不需要再次付费。

请你告诉小凯，他回家（到达 n 号站点）所需的最小花费是多少。

样例

【样例 1 解释】



上图中每条边上标记的数字为其所属的地铁线。该组样例中小凯可以购买 1 、4 两条线的通票，总价为 9 元，然后经过 1 – 4 – 5 车站回家。

子任务

对于 20% 的数据，有 $m = k$，即每条边分别属于一条不同的地铁线路。

另有 20% 的数据，有 $m \leq 1000, k \leq 12$。

另有 30% 的数据，有 $m \leq 1000$。

对于 100% 的数据，有 $1 \leq n, k \leq m \leq 10^6, 1 \leq u_j, v_j \leq n, u_j \neq v_j, 1 \leq line_j \leq k, 1 \leq c_i \leq 1000$。

子任务1

对于 20% 的数据，有 $m = k$，即每条边分别属于一条不同的地铁线路。

此时与正常的最短路模型完全相同，写一个dijkstra板子即可。

时间复杂度 $O((n+m)\log m)$。

子任务2

另有 20% 的数据，有 $m \leq 1000, k \leq 12$。

此时$k \leq 12$，因此可以2^k没进每张地铁通票买不买，然后bfs看一下只通过买了的票能否到达终点，取所有合法方案中票价总和最小的输出。

时间复杂度 $O(2^k m)$。

子任务3

另有 30% 的数据，有 $m \leq 1000$。

从这个部分分开始，我们需要更改建图方式。
可能存在别的方案，这里介绍其中一种。

考虑到每条线路上的点互相之间可以仅通过该线路到达，我们可以点边互换，对于每个点枚举所有与它相连的边，并且把地铁线路作为点，在边所属的地铁线路之间建边。然后跑最短路。

为了统计答案，还需要建出超级源点和超级汇点。

时间复杂度 $O(m^2 \log m^2)$。

正解

对于 100% 的数据，有 $1 \le n, k \le m \le 10^6, 1 \le u_j, v_j \le n, u_j \ne v_j, 1 \le line_j \le k, 1 \le c_i \le 1000$。

依然需要更改建图方式。

考虑到实际上我们不会在下一条线后重复登上该线路。我们把地铁线路也作为点，枚举每条边，由原端点向所属地铁线路连长度为通票价格的有向边，由所属地铁线路向原端点连长度为0的有向边，这样就可以模拟只付一次票价的过程。然后跑最短路即可。

时间复杂度 $O((n+m)\log m)$。

参考代码

```
int main(){
    scanf("%d%d%d",&n,&m,&k);
    for(int i=1;i<=k;i++)
        scanf("%d",&c[i]);
    for(int i=1;i<=m;i++){
        int u,v,be;
        scanf("%d%d%d",&u,&v,&be);
        g[n+be].push_back({u,0});
        g[n+be].push_back({v,0});
        g[u].push_back({n+be,c[be]});
        g[v].push_back({n+be,c[be]});
    }
    dij();
    printf("%d\n",dis[n]);
    return 0;
}
```

洛谷

THANKS

Q&A