目录

8.12 练习题

1. New Year Book Reading CF500C

【问题描述】

小明非常喜欢读书，他有一共有 n 本书，编号为 1~n，第 i 本书重 wi。小明计划在暑假的 m 天里每天读一本书，第 i 天读第 di 本书，可能会重复读到同一本书。因为所有的书都是堆成一摞的，所以每次读某本书之前小明都需要先将这本书上面所有的书搬开，拿出这本书，再将搬开的书按原顺序放回去，消耗体力为搬开书的重量之和，读完这本书后将其放在这摞书的最上面。小明想知道这 n 本书以怎样的初始顺序放置，所搬书消耗总体力最小。

【输入格式】

第一行两个正整数 n，m，表示小明一共有 n 本书，要读 m 天。 第二行 n 个正整数，第 i 个数表示表示第 i 本书的重量 wi。第三行 m 个正整数，第 i 个数表示第 i 天要读第 di 本书。

【输出格式】

仅一行一个数，表示读完 m 次书所搬书消耗的最小体力值。

【样例输入】

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

【样例输出】

```
12
```

题目链接：https://www.luogu.com.cn/problem/CF500C

【题目分析】

考虑对于排好序的相邻两本书 i 和 j，先看 i 再看 j，那么同时看完两本书的代价只多出来了 w[i]，如果交换 i 和 j 的顺序，先看 i 再看 j，同时看完两本书的代价为 w[i]+w[j]，以此可以类推出任意相邻两本书的排序关系，即按照第一次出现的位置进行模拟即可。Σ

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e3 + 5;
int n, m, w[N], a[N], ans;//n,m,w[]意义请看题目，a[]为要看的书，ans 为看所有书需要的最少体力
bool vis[N];//vis 数组用来标记
int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> w[i];//输入每本书的重量，需要的体力
    for (int i = 1; i <= m; i++)
    {
        cin >> a[i];//输入要看的书
        memset(vis, 0, sizeof(vis));//对于每次阅读书，都要重新统计上面的所有书的总重量
        int sum = 0;
        for (int j = i - 1; j >= 1; j--)//倒序枚举之前看过的书
        {
            if (a[j] == a[i])//如果之前看过，那么 a[j]前面的书就不需要搬动
                break;
            if (!vis[a[j]])//重复的书只统计一次重量
            {
                sum += w[a[j]];
                vis[a[j]] = true;//标记这本书已经统计过了，后面不计算重量
```

```
            }
        }
        ans += sum;//需要的体力加到总体力里面去
    }
    cout << ans;//输出总体力
    return 0;
}
```

2. Vertical Paths CF1675D

【问题描述】

给定一棵由 n 个顶点组成的有根树。顶点由 1 到 n 编号。任何顶点都可以是树的根。

请在树上找出这样一组路径：

每个顶点恰好属于一条路径，每条路径可以包含一个或多个顶点；

在每条路径中，每个节点的下一个节点是当前节点的子节点（即路径总是向下 —— 从父节点到子节点）；

路径的数量最少。

【样例输入】

```
6
5
3 1 3 3 1
4
1 1 4 1
7
1 1 2 3 4 5 6
1
1
6
4 4 4 4 1 2
4
2 2 2 2
```

【样例输出】

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

2
2
1 2
2
4 3

1
7
1 2 3 4 5 6 7

1
1
```

```
1

3
3
4 1 5
2
2 6
1
3

3
2
2 1
1
3
1
4
```

题目链接：https://www.luogu.com.cn/problem/CF1675D

【题目分析】

题目要求把一棵有根树划分为多条链，要求每个结点只属于一条树链，链必须要从祖先到后代，且树链的数量最少。

对于从非叶子结点出发的每一条树链，其终点必为叶子结点，所以树链数量等于叶子结点的数量，在实现过程中，可以从叶子结点往上推一直推到根结点或是已经加入某个树链的结点，并倒序输出路径即可。或者直接对树做重链剖分即可，因为一棵树的重链数量恰好等于其叶子结点数量。

【参考代码】

```
#include <cstdio>
#include <cstring>
using namespace std;
int fa[200005], path[200005];
bool vis[200005],leaf[200005];
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        memset(vis, 0, sizeof(vis));
        memset(leaf, 1, sizeof(leaf));
        int n, ans = 0;
        scanf("%d", &n);
        for (int i = 1; i <= n; i++) {
            scanf("%d", &fa[i]);
            if (fa[i] != i)
                leaf[fa[i]] = 0;
        }
        for (int i = 1; i <= n; i++)
            if (leaf[i])
                ans++;
        printf("%d\n", ans);
        for (int i = 1; i <= n; i++)
            if (leaf[i]) {
                int now = i, len = 0;
```

```
                while (!vis[now]) {
                    path[++len] = now;
                    vis[now] = 1;
                    if (fa[now] == now)
                        break;
                    now = fa[now];
                }
                printf("%d\n", len);
                for (int i = len; i >= 1; i--)
                    printf("%d ", path[i]);
                puts("");
            }
        puts("");
    }
    return 0;
}
```

3. Phone Numbers CF898C

【问题描述】

Vasya 有几本电话簿，他记录了他的朋友的电话号码。他的每个朋友可以有一个或几个电话号码。 Vasya 决定组织有关朋友电话号码的信息。您将获得 n 个字符串 - 来自 Vasya 电话簿的所有条目。每个条目都以朋友的名字开头。然后跟随当前条目中的电话号码数量，然后是电话号码本身。有可能几部相同的电话被记录在同一记录中。 Vasya 还认为，如果电话号码 a 是电话号码 b 的后缀（即，电话号码 b 以 a 结尾），并且两个电话号码都由 Vasya 写成同一个人的电话号码，则记录 a 并且没有城市代码，它是不被考虑在内的。 任务是输出有关 Vasya 朋友电话号码的组织信息。两个不同的人可能有相同的号码。如果一个人有两个数字 x 和 y，并且 x 是 y 的后缀（即 y 以 x 结尾），那么您不应该输出数字 x。如果 Vasya 电话簿中的朋友的号码以相同的格式记录多次，则有必要将其记录一次。 阅读样例以更好地理解输出的语句和格式。

【输入格式】

第一行包含整数 n（1<=n<=20）表示 Vasya 电话簿中的条目数。 下面的 n 行后面是描述声明中描述的格式的记录。 Vasya 的朋友的名字是非空字符串，其长度不超过 10，只包含小写英文字母。 一个条目中的电话号码不少于 1 不超过 10。电话号码只包含数字。如果您将电话号码表示为字符串，则其长度将在 1 到 10 的范围内。电话号码可以包含前导零。

【输出格式】

输出出有关 Vasya 朋友电话号码的订购信息。 首先输出 m 表示在 Vasya 电话簿中找到的朋友的数量。 下列 m 行必须包含以下格式的条目“姓名 电话号码的个数 电话号码”。电话号码应该用空格隔开。每个记录必须包含当前朋友的所有电话号码。条目可以以任意顺序显示，一个记录的电话号码也可以以任意顺序打印。

【样例输入】

```
3
karl 2 612 12
petr 1 12
katya 1 612
```

【样例输出】

```
3
katya 1 612
petr 1 12
karl 1 612
```

题目链接：https://www.luogu.com.cn/problem/CF898C

【题目分析】

首先简化题意，给出 n 个朋友的名字，每个朋友有 m 个电话号，对于每个朋友，如果存在一个电话号 s1 是 s2 的后缀，则将 s1 省去。输出有几个朋友，和他们的电话号。

Trie 树可以帮助我们寻找多个字符串的前缀，所以这道题可以利用 trie 进行构造。而这道题的要求是求后缀，实际上只要将字符串进行反转，就可以变换成前缀的问题，而每一个朋友都是独立的，互不干扰，因此可以对每一个朋友建立一棵 Trie 树，将每个朋友的电话号都插入到 Trie 中。然后在 trie 里找后缀即可，注意如果找到了两个字符串完全相同，那么无论是否构成后缀都应该保留这个字符串。而对于多个相同的字符串我们只需保留一次，这时候特判标记是否为 1 即可。同时名字也要特殊处理，我们可以用 map 帮助我们进行朋友名字的处理。

【参考代码】

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

using namespace std;

const int N = 10010;
int n, ncnt;//ncnt 是一共多少个人名
map<string,int>nnum;//人名对应的编号
map<int,string>names;//编号对应的人名
int cnt[21], tag[21][N], pos[21][N], t[21][N][10], num[21];
string a[21][N];//第一维都是表示第 i 个人，a 是当前人所有的电话号

inline void add(int x, string s)//插入字符串
{
    int p = 0, len = s.size();
    for(int i = len - 1; i >= 0; i --)//后缀，倒序插入
    {
        int c = s[i] - '0';
        if(!t[x][p][c])t[x][p][c] = ++ cnt[x];
        p = t[x][p][c];
        tag[x][p] ++;//记录当前点经过的电话号数量
    }
    pos[x][p] ++;//记录当前电话号的末尾
}

inline int ask(int x)//询问当前人的电话号数量
{
    int res = 0;//最终答案数目
    for(int i = 0; i <= cnt[x]; i ++)//遍历每一个点
    {
        if(tag[x][i] == 1 && pos[x][i]) res ++;//如果当前人的当前点只有一个电话号经过并且是末尾那就直接加一
        else if(tag[x][i] && pos[x][i] && tag[x][i] == pos[x][i]) res ++;//这个是处理有两个电话号重复的地方
    }
    return res;
}

inline void print(int x)//打印当前人的所有电话号
{
    for(int j = 1; j <= num[x]; j ++)//枚举所有电话号
    {
        int p = 0, len = a[x][j].size();
```

```
        for(int i = len - 1; i >= 0; i --)//倒序遍历
        {
            int c = a[x][j][i] - '0';
            p = t[x][p][c];
        }
        if(tag[x][p] == 1 && pos[x][p]) cout << a[x][j] << " ";//如果要是只有一个电话号到过这里并且是末尾就输出
        else if(tag[x][p] && pos[x][p] && tag[x][p] == pos[x][p]) cout << a[x][j] << " ";//处理重复的串。
    }
}

signed main()
{
    cin >> n;
    int nnn = n;
    while(nnn --)
    {
        string s;
        int k;
        cin >> s >> k;
        if(!nnum[s])//如果要是当前人名没出现过
        {
            names[++ ncnt] = s;//存一下，标记出现过
            nnum[s] = ncnt;//存对应人名的编号
        }
        for(int i = 1; i <= k; i ++)//枚举当前人的电话号
        {
            string x;
            cin >> x;
            int flag = 0, m = nnum[s];//flag 是标记当前电话号是否合法 ， m 是当前人名的编号
            for(int l = 0; l <= num[m]; l ++)//枚举每一个电话
                if(a[m][l] == x) flag = 1;//如果有一样的就标记不合法
            if(flag == 0)//是合法的电话号
                a[m][++ num[m]] = x;//存入当前的人名下面
            add(m, x);//插入当前号码
        }
    }
    cout << ncnt << endl;//输出人名数量
    for(int i = 1; i <= ncnt; i ++)
    {
        cout << names[i] << " " << ask(i) << " ";//输出当前人的人名和电话号数量
        print(i);//打印电话号码
        cout << endl;
    }
    return 0;
}
```

4. World Tour CF666B

【问题描述】

一张 n 个点 m 条边的有向图,每条边的权值相同.你要找 4 个点 a,b,c,d,使得 a->b->c->d 的最短路最长(a,b,c,d 之间要有路),输出一组解.

【输入格式】

第一行两个整数 n，m;

后面 m 行给出路径关系。

【输出格式】

输出一组解

【样例输入】

```
8 9
1 2
2 3
3 4
4 1
4 5
5 6
6 7
7 8
8 5
```

【样例输出】

```
2 1 8 7
```

题目链接：https://www.luogu.com.cn/problem/CF666B

【题目分析】

首先暴力思路，枚举四个点，计算最短路，时间复杂度 O(n^5)

然后可以继续优化，首先求出任意两点间的最短路，将每个点作为起点 BFS 即可，预处理时间复杂度 O（nm），这样枚举四个点，O(1)计算 d[a][b]+d[b][c]+d[c][d]，时间复杂度 O(n^4)。

但数据范围要求我们最多枚举两个点，观察式子，bc 两点均出现两次，而 ad 只出现一次，所以可以先枚举 bc，再确定 ad。

对于每个点 x 与处理出 d[y][x]最大的 y 和 d[x][z]最大的 z。枚举 b,c 时就可以 O(1)找到 d[y][b]最大的 y 作为 a,d[c][z]最大的 z 作为 d，使得此时答案最大。但我们枚举 bc 时，通过 b，c 找到的 a，d 可能相等或者与 bc 重合，因此直接找无法实现。此时只要点 a，d 不与另外三个点重复即可，因此只需要预处理出 d 值前三大的点，就可以使得 a，b，c，d 互不相同。

【参考代码】

```
#include<bits/stdc++.h>
#define res(a,b,c,d) h[a][b] + h[b][c] + h[c][d]
using namespace std;
const int N = 3005,M = 5005,INF = 0xcfcfcfcf;//INF 是一个极小值
int tot,head[N],ver[M],Next[M];
int n,m,h[N][N],s[N][3],t[N][3],A,B,C,D,ans;
// s[x][0]存储 y 表示 d(y,x)最大，s[x][1]存储 y 表示 d(y,x)次大,s[x][2]存储表示 d(y,x)第三大
// t[x][0]则是反过来，例如 t[x][0]存储 z 表示 d(x,z)最大
void add(int x,int y){
    tot++;
    ver[tot] = y;
    Next[tot] = head[x];
    head[x] = tot;
}
void BFS(int s,int (&d)[N]){
    queue<int> q;
```

```
    d[s] = 0;
    q.push(s);
    while(!q.empty()){
        int x = q.front();
        q.pop();
        for(int i = head[x];i;i = Next[i]){
            int y = ver[i];
            if(d[y] == INF){
                d[y] = d[x] + 1;
                q.push(y);
            }
        }
    }
}
void prework(){//预处理前三大
    for(int i = 1;i <= n;i++){
        for(int j = 1;j <= n;j++){
            if(i == j)continue;
            for(int k = 0;k < 3;k++){
                if(h[i][t[i][k]] <= h[i][j]){
                    for(int l = 2;l > k;l--)
                        t[i][l] = t[i][l - 1];
                    t[i][k] = j;
                    break;
                }
            }
            for(int k = 0;k < 3;k++){
                if(h[s[j][k]][j] <= h[i][j]){
                    for(int l = 2;l > k;l--)
                        s[j][l] = s[j][l - 1];
                    s[j][k] = i;
                    break;
                }
            }
        }
    }
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i = 1;i <= m;i++){
        int x,y;
        scanf("%d%d",&x,&y);
        add(x,y);
    }
    memset(h,0xcf,sizeof(h));
    for(int i = 1;i <= n;i++)
        BFS(i,h[i]);//以 i 作为起点，BFS 求最短路
    prework();
    for(int b = 1;b <= n;b++){
        for(int c = 1;c <= n;c++){
            if(b == c || h[b][c] == INF)continue;
```

```
            int x = 0,y = 0;
            if(s[b][x] == c)x++;//如果与 c 相同，跳过
            if(t[c][y] == b)y++;//如果与 b 相同，跳过
            if(s[b][x] == t[c][y]){//找到的 ad 相同，分两种情况
                int i = x + 1,j = y + 1;
                if(s[b][i] == c)i++;//换一个 a,当然还要保证 a 不为 c
                if(t[c][j] == b)j++;//换一个 c，同上不为 b
                int a = s[b][x],d = t[c][j];
                if(res(a,b,c,d) > ans){//换 d 不换 a
                    A = a,B = b,C = c,D = d;
                    ans = res(a,b,c,d);
                }
                a = s[b][i],d = t[c][y];
                if(res(a,b,c,d) > ans){//换 a 不换 d
                    A = a,B = b,C = c,D = d;
                    ans = res(a,b,c,d);
                }
          //两个同时换显然不优，不做讨论
            }
            else{
                int a = s[b][x],d = t[c][y];//ad 不相同，直接与答案比较
                if(res(a,b,c,d) > ans){
                    A = a,B = b,C = c,D = d;
                    ans = res(a,b,c,d);
                }
            }
        }
    }
    printf("%d %d %d %d\n",A,B,C,D);
    return 0;
}
```

5. WIL 洛谷 P3594

【问题描述】

给定一个长度为 n 的序列，你有一次机会选中一段连续的长度不超过 d 的区间，将里面所有数字全部修改 0。请找到最长的一段连续区间，使得该区间内所有数字之和不超过 p。

【输入格式】

输入的第一行包含三个整数，分别代表 n,p,d。第二行包含 n 个整数，第 i 个整数代表序列中第 i 个数 wi

【输出格式】

包含一行一个整数，即修改后能找到的最长的符合条件的区间的长度。

【样例输入】

```
9 7 2
3 4 1 9 4 1 7 1 3
```

【样例输出】

```
5
```

题目链接：https://www.luogu.com.cn/problem/P3594

【题目分析】

首先我们可以暴力枚举 O(n^3)，枚举左右端点，然后减去这段区间内 sum[x] - sum[x-d+1]的最大值，这样可以求出答

案。然后我们可以发现，区间的答案是单调不降的，对于每一个位置 i 为结尾能形成的最长区间的座端点是单调不降的，将 i-1 这个位置作为结尾形成的最长区间的左端点不可能比 i 作为结尾形成的最长区间的左端点更靠右。

所以我们枚举右端点 r，对于每一个 r，可以找到一个最小的 l，使得修改其中一段 d 后，[l,r]是可行的，此区间可行当且仅当 sum[r]-sum[l-1]-max(sum[x]-sum[x-d])<=p。其中 sum[x]-sum[x-d]需要我们使用单调队列进行维护。

此外也可以二分区间长度利用单调队列 check 来得到答案。

【参考代码】

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

using namespace std;

const int maxn = 2000005;
int n, d, l, r, start, ans;
int que[maxn];
long long p, a[maxn], sum[maxn], ad[maxn];

int solve()
{
    l = r = 1;
    start = 1;
    for(int i = d; i <= n; i++){
        ad[i] = sum[i] - sum[i - d];
    }
    int ret = d;
    que[r++] = d;
    for(int i = d + 1; i <= n; i++){
        while(l < r && ad[i] > ad[que[r - 1]]) r--;
        que[r++] = i;
        while(l < r && sum[i] - sum[start - 1] - ad[que[l]] > p){
            start++;
            while(l < r && que[l] - d + 1 < start) l++;
        }
        ret = max(ret, i - start + 1);
    }
    return ret;
}
int main()
{
    scanf("%d%lld%d", &n, &p, &d);
    for(int i = 1; i <= n; i++){
        scanf("%lld", &a[i]);
        sum[i] = sum[i - 1] + a[i];
    }
    ans = solve();
    printf("%d\n", ans);
    return 0;
}
```

6. Arpa's weak amphitheater and Mehrdad's valuable CF741B

【问题描述】

有 n 个人(1≤n≤1000)。每个人有一个重量 wi (1≤wi≤1000) 和一个魅力值 bi (1≤bi≤10^6)。n 个人之间有 m(1≤m≤min(n×(n−1)/2,10^5)) 个关系。第 i 个关系由两个数字 xi 和 yi 组成，表示第 xi 个人和第 yi 个人是朋友，朋友关系是双向的。

已知若 a 和 b 是朋友，b 和 c 是朋友，则 a 和 c 是朋友。 现在 Mehrdad 要邀请一些人来到派对，使这些人的重量总和不超过 wi(1≤wi≤1000)，且魅力值总和尽量大。同一个朋友圈里的人，只能邀请其中的一个人，或者全部人，或者一个人也不邀请。

【输入格式】

第一行，三个整数 n，m，w
第二行，n 个整数 w1，w2，…，wn。
第三行，n 个整数 b1，b2，…，bn。
接下来 m 行，每行表示一个关系。

【输出格式】

一行，表示最大的魅力值总和。

【样例输入】

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

【样例输出】

```
6
```

题目链接：https://www.luogu.com.cn/problem/CF741B

【题目分析】

并查集+分组背包的综合应用。首先题目中要求我们判断朋友之间的关系，那么并查集可以帮助我们处理，但是在邀请朋友时，只能找一个朋友或者全部都不找，这与分组背包的模型很像，这里涉及到一个处理的技巧，可以把同一个组内的所有元素的总和当作一个元素进行处理，即要么找一个，要么不找，要么找全部。同时需要注意，如果只有一个元素，就不应该把所有的元素总和加入分组背包，时间复杂度 O（nw）。

【参考代码】

```
#include<bits/stdc++.h>
#include <bits/stdc++.h>
using namespace std;
int n,m,wt,w[1007],b[1007],a[1007],num[1007],totw[1007],totb[1007],cnt;//解释一下
//n：初始人数 m：关系数 wt:weight tot 最高体重和
//w，b 数组：初始体重及魅力 a 数组：记录每一个并查集标志数（即该组内所有人的 father）所属于的组数，由此来将一个并查集中的元素放入一个组 num 数组：记录每一个组内元素个数（不含所有元素和，这个后面加） totw，totb 数组：记录每一个组的 w 及 b 之和 cnt：记录组数
int fa[1007],d[1007][1007],x,y,f[1007];
//fa 数组：并查集 d 数组：记录每一个组别内的每一个元素的编号（是编号哦！！） x，y ： 读入工具 f 数组：记录最大魅力和，即背包
int find(int x){
    if(x==fa[x]) return x;
    return fa[x]=find(fa[x]);//并查集+路径压缩
}
int main(){
    scanf("%d%d%d",&n,&m,&wt);
    for(int i=1;i<=n;i++) scanf("%d",&w[i]);
    for(int i=1;i<=n;i++) scanf("%d",&b[i]);
    for(int i=1;i<=n;i++) fa[i]=i;//读入初始化
    for(int i=1;i<=m;i++){
```

```
        scanf("%d%d",&x,&y);
        int p=find(x);
        int q=find(y);
        if(p!=q) fa[p]=q;//构造并查集
    }
    for(int i=1;i<=n;i++){
        int p=find(i);
        if(a[p]==0) a[p]=++cnt;//新建组别，通过标志数帮每一个元素找到自己的组别
        d[a[p]][++num[a[p]]]=i;//为组别内加元素
        totw[a[p]]+=w[i];totb[a[p]]+=b[i];//更新该组别总重，总魅力
    }
    for(int i=1;i<=cnt;i++){
        if(!(totw[i]==w[d[i][1]]&&totb[i]==b[d[i][1]])){//上面的注意点
            d[i][++num[i]]=n+1;//将组别的总和作为最后一个元素加入组别
            w[n+1]=totw[i];b[n+1]=totb[i];
        }
        for(int v=wt;v>=0;v--)
            for(int j=1;j<=num[i];j++)
                if(v>=w[d[i][j]])
                    f[v]=max(f[v],f[v-w[d[i][j]]]+b[d[i][j]]);//背包
    }
    printf("%d",f[wt]);
    return 0;
}
```