Educational Codeforces Round 55 (Div. 2) 解题报告

SGColin

目录

1 A. Vasya and Book

1.1 Description

现在有一本页码为 $1...n$ 的书，你现在展开在第 x 页，想要翻到第 y 页。

你一次只能向前或向后翻 d 页，向前翻到第 1 页之后会停在第 1 页，向后翻到第 n 页同理。

问最少翻页次数，若无解则输出 -1 。

1.2 Solution

签到题。分情况讨论一下，答案有三种情况，直接翻，从 x 翻到第 1 页再翻到第 y 页，从 x 翻到第 n 页再翻到第 y 页。注意第一个情况时答案的符号处理。

2 B. Vova and Trophies

2.1 Description

给出一个由字母 G 和字母 S 构成的序列，允许选择两个位置交换他们的字符，求操作后最长的 G 序列有多长。

2.2 Solution

签到题。讨论一下答案的构成：

(1) 原序列中的一段，答案为 len

(2) 原序列中的两段，中间隔着一个 S ，拿其中一段的端点字符来补，答案为 $len1 + len2$

(3) 原序列中的两段，中间隔着一个 S ，拿两段以外的字符来补，答案为 $len1 + len2 + 1$

其中 len 数组可以预处理，两端外的字符可以正反各扫描一遍，同时维护一个计数器。

3 C. Multi-Subject Competition

3.1 Description

有 n 个包含若干权值的集合，现让你确定一个 x ，使得从这 n 个集合中选出若干集合，每个集合再选出 x 个数得到的和最大。如果一个集合里的元素个数 $< x$ ，这个集合不能被选中。

3.2 Solution

二次前缀和，想清楚细节。

我们可以将每一个集合内部先从大到小排序，显然如果要选这个集合，我们一定是选择一个前缀。

我们设 $sum[i][j]$ 表示第 i 个集合前 j 项的和，特殊的，如果第 i 个集合元素个数不到 j，这个值为 $-\infty$。我们用 $w[i]$ 表示 $x = i$ 时的最佳答案，有等式成立：[1]

$$w[j] = \sum_{i=1}^{n} [sum[i][j] > 0]\ sum[i][j]$$

显然最后的答案就是 $\max w[i]$ 。其实这个答案关于 i 应该是一个单峰函数，但是此数据范围下并不需要使用三分法。

```
for(ll i=1,x,y;i<=n;++i){
  x=rd(); y=rd(); s[x].push_back(y);
}
for(ll i=1;i<=m;++i){
  sort(s[i].begin(),s[i].end(),cmp);
  for(ll j=0,tot=0;j<s[i].size();++j){
    tot+=s[i][j]; if(tot<0) break; sum[j+1]+=tot;
  }
}
for(ll i=1;i<=n;++i) ans=max(ans,sum[i]);
```

[1] $[\ x\ ]$ 是一个布尔表达式，若条件 x 为真则表达式的值为 1，否则为 0

4 D. Maximum Diameter Graph

4.1 Description

给出一个长度为 n 的数组 A ，请构造一张 n 个点的无向连通图，满足：

(1) 第 i 号节点度数 $\leq A[i]$

(2) 可能的前提下使图的直径 [2] 最大。

数据范围：$n \leq 500, A[i] \leq n-1$

4.2 Solution

构造题，要明确直径的含义是一条链。

首先要想清楚 $\leq A[i]$ 意味着什么。如果说一个图是合法的，那么这张图包含直径的一棵生成树也是合法的，因为原图中不在直径上的边删掉之后，点度只会变小，而图的直径也是这个生成树的直径。

因此我们只需要构造一棵生成树，并使得树的直径最大，构造方法就比较显然了。

首先把所有 $A[i] \geq 2$ 的点首尾连起来构成一条链，再在两端各连上一个 $A[i]=1$ 的点，直径就构造好了。此处注意特判 $A[i]=1$ 的点数为 0 或 1 的情况。

然后只需解决剩下的 $A[i]=1$ 的点。如果所有 $A[i] \geq 2$ 的点除掉用于构造链之后，剩下的点度之和 $<$ $A[i]=1$ 的点的个数，显然图是不成立的 [3]。特判掉之后，我们只需把那些剩下的 $A[i]=1$ 的点一个个插到链上有空余点度的点上即可。

[2]这里指的是，在简单路径的前提下，最远点对的距离

[3]这里仔细思考，显然这些点不能再空出更多的点度了，否则图的连通性无法保证

5 E. Increasing Frequency

5.1 Description

给出一个序列，你可以选择一个区间，为区间里的所有数都加上一个你选择的值。问最后整个数列里最多有多少个 x 。

数据范围：$n \leq 5 \times 10^5$

5.2 Solution

答案只会有三种组成：一个都不变，取原数列里的 x ；保留一部分原数列里的 x ，剩下的取一个区间的众数变成 x ；取数列的众数变成 x 。

有一种比较巧妙的利用差分的解法。考虑上述的三种情况中，第二种是最普通的形式，另两种也可以归纳进去。因此我们只考虑这种情况。设 $a[i]$ 表示到当前位置为止，数值为 i 的我们保留后 $a[i]$ 个（即后 $a[i]$ 个之前我们选择原数列中的 x ，后面我们将这 $a[i]$ 个变成 x ）。设 $b[i]$ 表示除掉后 $a[i]$ 个，剩下的前缀里 x 的个数。设 cnt 表示数列到当前位置 x 的个数。

考虑现在来了一个 i ，什么时候后 $a[i]$ 个是不优的。答案是当且仅当 $a[i]+b[i] < cnt$ 时，即后 $a[i]$ 个所在的极小后缀里 i 的个数少于 x 的个数。此时我们只保留最后一个 i ，令 $a[i] = 1$, $b[i] = cnt$ ，否则合法只需让 $a[i] + +$ 。答案就是 $max\{a[i] + b[i]\}$ 。

其实这样子是没有办法统计到中间一段做修改的情况的，因为我们只会在一个后缀做区间修改。然后就有了一个绝妙的思路，考虑每次我们都把选取的后缀里前 $cnt - b[i]$ 个 i 看作取了 x ，也就是每次合法时令 $a[i] += b[i] - cnt$, $b[i] = cnt$，此时我们记录 $a[i]$ 每一位置的最大值，再加上全局的 cnt 即为答案。这种方法统一了两种过程的更新方式，当不合法时，$a[i] = 1$ ，剩下的都取原数列中的 x ; 当合法时，我们把一部分答案放到 $b[i]$ 里，不影响后续的判断，同时记录了对于全局 cnt 的一个可能的最优增量。

```
n=rd(); m=rd();
for(R int i=1,x;i<=n;++i)
  if((x=rd())==m) ++cnt;
  else{
    a[x]+b[x]>=cnt?a[x]+=b[x]-cnt:a[x]=0;
    b[x]=cnt; ans=max(ans,(++a[x]));
  }
printf("%d\n",ans+cnt);
```

6 F. Speed Dial

6.1 Description

有一个电话册，里面记录了若干个由数字 $0\sim 9$ 组成的数字串 s_i，以及一个当天需要拨打的次数 w_i。现在你的手机上有 $0\sim 9$ 这 9 个普通键，还有 k 个自定义键，你可以为这 k 个自定义键每个都定义一个独特的数字串。每次拨打一个号码的时候，按下自定义键就会直接输进去对应的数字串，但特殊的是每次拨打**只能按一次**自定义键，并且必须是本次拨打按下的**第一个键**，即该自定义键对应的字符串是你想要拨打的号码的一个前缀。

请设置这 k 个自定义键的数字串，使得拨打完电话册里的所有号码的所有次数的过程中，按下普通键的次数最少。

数据范围：$\sum |s_i| \le 500, k \le 10$

6.2 Solution

考虑把电话册的串建成一棵 Trie ，并把一个串出现的次数记录到这个串在 Trie 里对应的最深的节点（权）上。那么我们自定义键对应的字符串，在这个 Trie 上一定是从根出发的一个路径，那么我们可以看作是标记了一个特殊点，然后把它到根的路径视为一个自定义键对应的串。访问一个节点的代价，即为这个点的点权，乘上它到最近的被标记祖先的距离。总结一下，问题就变为，在一棵树上，点有点权，可以标记 k 个特殊点，每个节点的代价为点权乘以到最近的被标记祖先节点（含自己）的距离，默认根节点被标记，最小化代价和。

这个问题的模型就是 [IOI 2005]River 了。模型和那道题一模一样。树形 DP 的思路非常奇特，设 $f[u][v][k]$ 表示当前是节点 u ，**最近的被标记的祖先是** v ，子树内有 k 个点被标记，子树代价的最小值。为了方便转移，我们设 $g[u][v][k]$ 表示 u 节点强制标记的答案，最后再把两个数组合并到一起。为了便于寻找祖先，在 DFS 的时候我们维护一个访问节点的栈 stk 。

DP 的过程是 $\mathrm{O}(n^2k^2)$ 的，注意 g 数组更新的时候子节点对应的第二维祖先是当前节点。

```
for(R int j=1;j<=top;++j)
 for(R int k=m;~k;--k){
   f[u][stk[j]][k]+=f[v][stk[j]][0];
   g[u][stk[j]][k]+=f[v][u][0];
   for(R int x=k;~x;--x){
     f[u][stk[j]][k]=min(f[u][stk[j]][k],f[u][stk[j]][k-x]+f[v][stk[j]][x]);
     g[u][stk[j]][k]=min(g[u][stk[j]][k],g[u][stk[j]][k-x]+f[v][u][x]);
   }
 }
```

接下来就是合并数组的过程了，为了便于计算到被标记祖先的距离，我们使用树上前缀和的思路，记录每个节点到根节点的距离 d 。此时要注意当前节点的代价在 DP 过程中还未计算！考虑每一个 k 对应的 f 数组，如果它来自 g ，那么对应的 $k' = k - 1$ ；否则来自 f ，那么它需要加上 $w[u] * (d[u] - d[v])$ 的代价。两种情况取 min 即可。

```
for(R int j=1;j<=top;++j)
  for(R int k=m,dis;~k;--k){
    dis=d[u]-d[stk[j]];
    if(k) f[u][stk[j]][k]=min(g[u][stk[j]][k-1],f[u][stk[j]][k]+c[u].w*dis;
    else f[u][stk[j]][k]+=c[u].w*dis;
  }
ans=f[root][root][m];
```

其实本题两个数组可以合到一起，还有能用多叉树转二叉树的 DP 顺序做。

7 G. Petya and Graph

7.1 Description

给出一张无向图，点有点权，边有边权。给出子图的定义是，如果你选择了一条边，那么必选选上它的两个端点。定义子图的价值是，所有选中的边权之和 – 所有选中的点权之和。现在需要你选择一个子图，子图可以不连通，问选出的子图价值最大可以多大。

数据范围：$n, m \leq 10^3$，权值均为非负数。

7.2 Solution

CodeForces 竟然在压轴题上出了一道知识壁垒题......

学过最大权闭合子图的人应该很容易发现模型。把每条边看成一个新点，选这个新点就必须选原来那条边连接的两个端点，此时新点点权为边权，原来的端点点权为原来点权的相反数，问题就变成了最大权闭合子图。

直接重建图之后，使用最大流最小割定理求解即可，答案为边权和 – 最大流。