第五周直播课（下午）
综合模拟/作业讲评

洛谷网校
基础-提高衔接计划
2024-08
disangan233

综合模拟 总结

- 考点全是前四周的范围
- 难度对标 CSP-J，部分分给的很仁慈
- 作为综合模拟的第一试，希望大家准备好迎接后续的模拟
- 熟悉 OI 赛制、学会对拍和测试大样例
- 考完要学会反思总结，认真补题

T1 分身（haikei）

- 有 n 个分身，分身自行组成了团体，其中第 i 号分身将进入第 a_i 号团体
- 在接下来的 m 天里，每天指定两个不同的团体 x 和 y。在这天里，每一位 x 团体里的分身都和每一位 y 团体里的分身进行一次对话
- 对于每一次对话，如果两个分身的编号分别为 a 和 b，会提供 $a \times b$ 的情绪值
- 求每天可以获得多少情绪值
- $1 \leq n, m \leq 2 \times 10^5$，$1 \leq a_i \leq 10^6$

T1 分身（haikei）

- 部分分：$O(n^2)$ 枚举
- 枚举 i, j，判断是否 $a_i = x$ 且 $a_j = y$
- 如果满足，答案累加上 $i \times j$
- 60 pts

T1 分身（haikei）

- 令 x, y 团体的编号为 $p_1, p_2, \ldots, p_s$ 和 $q_1, q_2, \ldots, q_t$，求的是

$$\begin{aligned} ans &= (p_1 q_1 + p_1 q_2 + \ldots + p_1 q_t) + \ldots + (p_s q_1 + p_1 q_2 + \ldots + p_s q_t) \\ &= p_1(q_1 + q_2 + \ldots + q_t) + \ldots + p_s(q_1 + q_2 + \ldots + q_t) \\ &= (p_1 + p_2 + \ldots + p_s) \times (q_1 + q_2 + \ldots + q_t) \end{aligned}$$

- 更形式化的推导：

$$ans = \sum_{a_i=x} \sum_{b_j=y} i \times j = \left(\sum_{a_i=x} i \right) \times \left(\sum_{b_j=y} j \right)$$

- 对于 10^6 个团体，每个团体 i 用一个桶 s_i 记录成员的编号和
- 询问的结果就是 $s_x \times s_y$，时间复杂度 $O(n+m)$

T1 分身（haikei）

```
scanf("%d%d",&n,&m);
for(int i=1;i<=n;i++) {
    int x;
    scanf("%d",&x);
    a[x]+=i;
}
while(m--) {
    int x,y;
    scanf("%d%d",&x,&y);
    printf("%lld\n",a[x]*a[y]);
}
```

T2 工作（shigoto）

- 给出长度为 n 的正整数序列 $\{a_n\}$ 和 $\{b_n\}$，对于每个 $a_i(1 \le i \le n)$，进行恰好一次以下操作：
  - 将 a_i 变成满足 $|a_i - x| \le k \times b_i$ 的任意整数 x
- 求出最小的非负整数 k，使得存在至少一种方法使得操作后序列 $\{a_n\}$ 所有数都相等
- $1 \le T \le 10^5$，$2 \le \sum n \le 5 \times 10^5$，$1 \le a_i, b_i \le 10^9$

T2 工作（shigoto）

- 对于答案 k，第 i 个数对应 x 的取值范围为 $[a_i - k \times b_i, a_i + k \times b_i]$
- k 有解，对应的是这 n 个取值范围区间有交，即至少有一个整数在 n 个区间中
- $O(n)$ 判断区间是否有交：$\max l_i \leq \min r_i$，循环维护左端点最大值和右端点最小值

```
1  ll x=-1e18,y=1e18;
2  for(int i=1;i<=n;i++) {
3      x=max(x,a[i]-mid*b[i]);
4      y=min(y,a[i]+mid*b[i]);
5  } // x <= y 即有解
```

- 直接从小到大枚举 k，判断是否有解，时间复杂度 $O(n^2)$
- 60 pts

T2 工作（shigoto）

- 如果 k 满足，那么 $k+1$ 一定满足，考虑二分答案
- 考虑答案的上界：$a=\{1,10^9\},b=\{1,1\}$，$k=5\times10^8$
- 考虑答案的下界：$a=\{1,1\},b=\{1,1\}$，$k=0$
- 从 $[0,5\times10^8]$ 开始二分，区间有交 $r\leftarrow mid-1$，区间无交 $l\leftarrow mid+1$
- 时间复杂度 $O(n\log a)$
- `l = 1` 会被卡成 60 pts，`r = 1e8` 也会被卡成 60 pts

T2 工作（shigoto）

```cpp
scanf("%d",&n);
for(int i=1;i<=n;i++)
    scanf("%d",a+i); // a+i 即 &a[i]
for(int i=1;i<=n;i++)
    scanf("%d",b+i);
ll l=0,r=1e9;
while(l<=r) {
    ll mid=(l+r)/2,x=-1e18,y=1e18;
    for(int i=1;i<=n;i++) {
        x=max(x,a[i]-mid*b[i]);
        y=min(y,a[i]+mid*b[i]);
    }
    if(x<=y)
        r=mid-1;
    else
        l=mid+1;
}
printf("%lld\n",r+1);
```

T3 手链（bracelet）

- 手链的样式可以抽象为一个由 0 和 1 组成的字符串 s，且首尾相接
- 有 $00, 01, 11$ 三种长度为 2 的连续珠子，分别有 n, m, k 个。
- 珠子不能从中间分开，但可以翻转，01 珠子可以翻转为 10
- 求最长能串出多长的 s 的连续区间
- $0 \leq n, m, k \leq 10^6$，$1 \leq |s| \leq 10^6$，$s_i \in \{0, 1\}$

T3 手链（bracelet）

- 固定起点后，每个位置能放的珠子是唯一的
- 有一个显然的 $O(n^2)$ 做法，从每个位置开始，放到不能放为止
- 拆环成链，即在字符串结尾再拼接上一个原串
- 双指针枚举 i 为起点，j 为当前放的珠子，初始化 $j = i$
- 每次新放的珠子是 $s_j s_{j+1}$，如果有则放入珠子，$j \leftarrow j + 2$
- 到第一个不能放的地方结束，此时放入的长度是 $j - i$
- 50 pts

T3 手链（bracelet）

- 放下一个珠子，不会影响在模式串上对应位置的奇偶性
- 所以实质上只有两种放置方式：奇数/偶数起点
- 分奇偶性，做两次 2-pointer 即可
- 维护双指针 $[j, i)$，每次加入 $s_i s_{i+1}$ 时，若当前珠子有剩余，则直接加入，$i \leftarrow i + 2$
- 否则，弹出最左珠子，$j \leftarrow j + 2$，直到可加入为止
- 此处的做法是枚举终点，枚举起点同理
- 注意答案不能超过 $n - n \bmod 2$
- 时间复杂度 $O(n)$

T3 手链（bracelet）

```
int ask(char *s) {
    if (s[0] == '0' && s[1] == '0') return 0;
    else if (s[0] == '1' && s[1] == '1') return 2;
    else return 1;
}
cin >> n >> m >> k >> s;
int len = s.length();
s += s;
int cnt[3] = {n, m, k};
for (int i = 0, j = 0; i < 2 * len - 1; i += 2) {
    ans = max(ans, i - j);
    int type = ask(&s[i]);
    while (j < i && cnt[type] == 0) {
        cnt[ask(&s[j])]++;
        j += 2;
    }
    if (cnt[type] == 0) j += 2;
    else cnt[type]--;
}
```

T3 手链（bracelet）

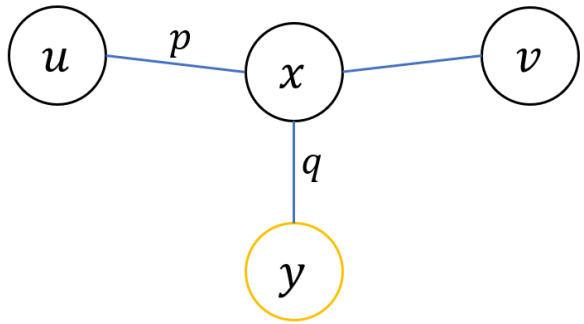
```
cnt[0] = n;
cnt[1] = m;
cnt[2] = k;
for (int i = 1, j = 1; i < 2 * len - 1; i += 2) {
    ans = max(ans, i - j);
    int type = ask(&s[i]);
    while (j < i && cnt[type] == 0) {
        cnt[ask(&s[j])]++;
        j += 2;
    }
    if (cnt[type] == 0) j += 2;
    else cnt[type]--;
}
ans = min(ans, len / 2 * 2);
printf("%d\n", ans);
```

T4 单车（iris）

- 有一棵 n 个节点的树，其中 k 个节点上有共享单车
- 步行出发，从一个节点到相邻节点需要 2 秒
- 到达了有共享单车的节点，就可以马上改为骑行，此后从一个节点到相邻节点只需要 1 秒
- 求从 x 节点出发，到 y 节点，至少需要多少时间
- q 次询问，询问间互相独立，即每次询问后共享单车的位置不会改变，每个节点可以经过多次
- $1 \leq n, k, q \leq 5 \times 10^5$，$1 \leq u, v, x, y \leq n$

T4 单车（iris）

- 称有共享单车的节点为特殊点。假设最优路线如图所示（其中 p 为 u 到点 x 的距离，q 为 x 到最近特殊点的距离）：



- 那么需要的时间为

$$2p + 2q + q + (\mathrm{dis}(u, v) - p) = 3d + p + \mathrm{dis}(u, v)$$

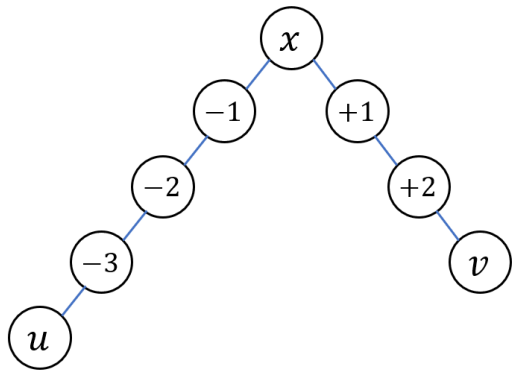
- 令点 i 最近特殊点的距离为 d_i，那么要求的是 $3d_i + p$ 的最小值
- d_i 可以用一遍 bfs 来 floodfill 预处理

T4 单车（iris）

- 部分分做法
- 对于每次查询，枚举经过的单车中转点，计算当前答案
- 使用倍增 $O(\log n)$ 求 lca 和 dis，单次查询 $O(k \log n)$
- 总时间复杂度 $O(nk \log n)$
- 50 pts

T4 单车（iris）

- 固定一个节点作为树根，每次询问形如：



- 求出 u, v 的 lca 后，p 可以用 $dep_u - dep_i$ 来计算，只需求：
  - $u \sim$ lca 的 $3d_i - dep_i$ 最小值
  - lca $\sim v$ 的 $3d_i + dep_i$ 最小值
- 用树上倍增维护 2^k 级父亲和 $3d_i \pm dep_i$ 的最小值
- 时间复杂度 $O((n+q)\log n)$

T4 单车（iris）

```
void add(int x,int y) {
    e[x].push_back(y);
    e[y].push_back(x);
}
void dfs(int u,int fa) {
    dep[u]=dep[fa]+1;
    p[u][0]=fa;
    f[0][u][0]=min(3*d[u]+dep[u],3*d[fa]+dep[fa]);
    f[1][u][0]=min(3*d[u]-dep[u],3*d[fa]-dep[fa]);
    for(int i=1;i<M;i++) {
       p[u][i]=p[p[u][i-1]][i-1];
       f[0][u][i]=min(f[0][u][i-1],f[0][p[u][i-1]][i-1]);
       f[1][u][i]=min(f[1][u][i-1],f[1][p[u][i-1]][i-1]);
    }
    for(int v:e[u])
        if(v^fa)
            dfs(v,u);
}
```

T4 单车（iris）

```
int lca(int x,int y) {
    if(dep[x]>dep[y]) swap(x,y);
    int l=dep[y]-dep[x];
    for(int i=0;i<M;i++)
        if((l>>i)&1)
            y=p[y][i];
    if(x==y)
        return x;
    for(int i=M-1;i>=0;i--)
        if(p[x][i]^p[y][i])
            x=p[x][i],y=p[y][i];
    return p[x][0];
}
```

T4 单车（iris）

```
int solve(int x,int y) {
    int z=lca(x,y),len=dep[x]+dep[y]-(dep[z]<<1);
    int ans=len*2;
    int tx=dep[x]-dep[z],ty=dep[y]-dep[z];
    int dx=dep[x],dy=dep[y];
    for(int i=0;i<M;i++)
        if((tx>>i)&1) {
            ans=min(ans,len+dx+f[1][x][i]);
            x=p[x][i];
        }
    for(int i=0;i<M;i++)
        if((ty>>i)&1) {
            ans=min(ans,len*2-dy+f[0][y][i]);
            y=p[y][i];
        }
    return ans;
}
```

T4 单车（iris）

```
n=read(),k=read();
for(int i=1;i<n;i++)
    add(read(),read());
memset(d+1,-1,4*n);
for(int i=1,x;i<=k;i++)
    x=read(),q.push(x),d[x]=0;
while(!q.empty()) {
    int u=q.front();q.pop();
    for(int v:e[u])
        if(d[v]==-1)
            d[v]=d[u]+1,q.push(v);
}
d[0]=1e8;
dfs(1,0);
m=read();
while(m--)
    printf("%d\n", solve(read(),read()));
```