CSP2024 练习赛（1）解题报告

T1 等边三角 解题报告

【解题思路】

首先如果我们的答案需要用到i,j,k三根木棍，那么一定只会修改i,j,k三根木棍。

观察到$n \leq 300$，所以枚举所有(i,j,k)的时间复杂度是可以接受的。

不妨设$a_i \leq a_j \leq a_k$，那么容易知道$(a_j - a_i) + (a_k - a_j) = a_k - a_i$是最小的花费。

对所有(i,j,k)取最小值即可。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;

int n, a[555];

int main()
{
    freopen("equal.in","r",stdin);
    freopen("equal.out","w",stdout);
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)  scanf("%d", &a[i]);
    sort(a + 1, a + n + 1);
    int ans = 1E9;
    for(int i = 1; i <= n; i++)
        for(int j = i + 1; j <= n; j++)
            for(int k = j + 1; k <= n; k++)
            {
                ans = min(ans, a[k] - a[i]);
            }
    cout << ans << endl;
    return 0;
}
```

T2 直线 解题报告

【解题思路】

由于一个人视线范围内的其他人无论向左或向右都会被统计进入答案，所以一个人的方向改变只会影响到他自己所能够看到的人，对于其他人来说他们所能看到的人是不会变的。

而人i向左能够看到的人数为$i-1$，向右能够看到的人数为$n-i$，故记录s_i代表一个人转向前后能够看到的人数之差，若初始朝左，则$s_i=2i-n-1$，反之则为$n+1-2i$。

然后将s_i从大到小排序，按照贪心的方式每次选择剩余的最大的s_i，将答案加上s_i。

若剩余最大的$s_i<0$，则不如不选，答案不变。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;

int n, f[100005], s[100005];

int main()
{
    freopen("line.in","r",stdin);
    freopen("line.out","w",stdout);
    scanf("%d", &n);
    long long sum = 0;
    for(int i = 1; i <= n; i++)
    {
        char ch = getchar();
        while(ch != 'L' && ch != 'R')  ch = getchar();
        if(ch == 'L')  sum += i - 1, s[i] = (n - i) - (i - 1);
        if(ch == 'R')  sum += n - i, s[i] = (i - 1) - (n - i);
    }
    sort(s + 1, s + n + 1);
    for(int i = n; i >= 1; i--)
    {
        if(s[i] > 0)  sum = sum + s[i];
        printf("%lld ", sum);
    }
    return 0;
}
```

T3 塔 解题报告

【解题思路】

要求最大值最小，首先考虑二分答案。

假设二分出一个值x，考虑验证其是否合法。

首先距离大于x的塔需要到不同的组中去，距离小于等于x的塔可以随意安排。

看到题目要求将塔分为两组，那么首先考虑到的就是一张二分图，不同组之间的塔可以连边，而同组的塔不能连边。

于是我们将距离大于x的塔用一条边连接，然后直接二分图染色即可判断是否合法，如果是一张二分图，则合法，反之不合法。

确定了最小值后再求出二分图中连通块数量，设为m，一个连通块可以以任意染色方式加入另一个连通块中，故最终方案数为2^m。

【参考代码】

```
#include<bits/stdc++.h>
#define Mod 1000000007
using namespace std;

int n, x[5005], y[5005], dis[5005][5005];

int qpow(int x, int k)
{
    int res = 1;
    while(k) {
        if(k & 1)  res = 1ll * res * x % Mod;
        x = 1ll * x * x % Mod;
        k >>= 1;
    }
    return res;
}

int vval, col[5005];




int chk(int x)
{
    int num = 0;
    memset(col, 0, sizeof(col));
```

```
    for(int i = 1; i <= n; i++)
    {
        if(!col[i])
        {
            queue<int> q; q.push(i); col[i] = 1;
            while(!q.empty())
            {
                int u = q.front(); q.pop();
                for(int j = 1; j <= n; j++)
                {
                    if(dis[u][j] > x)
                    {
                        if(col[j] == col[u])  return 0;
                        if(col[j])  continue ;
                        q.push(j), col[j] = 3 - col[u];
                    }
                }
            }
            ++num;
        }
    }
    vval = num;
    return 1;
}

int main()
{
    freopen("tower.in","r",stdin);  freopen("tower.out","w",stdout);
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)  scanf("%d %d", &x[i], &y[i]);
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            dis[i][j] = abs(x[i] - x[j]) + abs(y[i] - y[j]);
    int l = 0, r = 10000, ans = -1; chk(0);
    while(l <= r)
    {
        int mid = (l + r) >> 1;
        if(chk(mid))  ans = mid, r = mid - 1;
        else  l = mid + 1;
    }
    chk(ans);
    cout << ans << endl << qpow(2, vval) << endl;
    return 0;
}
```

T4 最短路 解题报告

【解题思路】

当$m = n - 1$时，1到i的路径唯一，直接 dfs 即可。

考虑图的情况，我们可以转化一下题意，转化为1到i的路径中将某一条边权值变为0并将某一条边权值变为$2w$的最小值。

操作只能进行一次，所以考虑分层图解法，做一次操作就向上走一层，因为高层不能回到低层，所以可以保证每种操作只做一次，而做一次操作才能上一层，所以顶层的点可以保证两次操作都被进行。

然后考虑如何找到最大和最小权值的边，答案很简单，直接跑最短路即可，因为程序一定会在最大权值的边上一层，在最小权值的边再上一层，否则总会有一条更短路存在。

然后分先走 max上第二层和先走 min上第二层建两个分层图讨论一下即可。

最后特判掉走一条边即可从1到i的情况即可，因为此时 min,max走的是同一条边，在分层图上无法实现。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;

int first[2000005], nxt[2000005], to[2000005], tot, w[2000005];

void Add(int x, int y, int z)
{
    nxt[++tot] = first[x]; first[x] = tot; to[tot] = y; w[tot] = z;
}

long long d[500005];
int n, m, vis[500005];

struct node
{
    long long dis;
    int id;
    node(long long A = 0, int B = 0) {dis = A, id = B;}
    bool operator < (node A) const {
        return dis > A.dis;
    }
};
```

```
int main(){
    freopen("path.in","r",stdin);
    freopen("path.out","w",stdout);
    scanf("%d%d",&n,&m);
    for(int i = 1; i <= m; i++){
        int x, y, z;
        scanf("%d %d %d", &x, &y, &z);
        Add(x, y, z); Add(y, x, z);
        Add(x + n, y + n, z); Add(y + n, x + n, z);
        Add(x + 2 * n, y + 2 * n, z); Add(y + 2 * n, x + 2 * n, z);
        Add(x + 3 * n, y + 3 * n, z); Add(y + 3 * n, x + 3 * n, z);
        Add(x, y + n, 0); Add(y, x + n, 0);
        Add(x, y + 2 * n, 2 * z); Add(y, x + 2 * n, 2 * z);
        Add(x + n, y + 3 * n, 2 * z); Add(y + n, x + 3 * n, 2 * z);
        Add(x + 2 * n, y + 3 * n, 0); Add(y + 2 * n, x + 3 * n, 0);
    }
    priority_queue<node> q;
    for(int i = 1; i <= n * 4; i++) {
        vis[i] = 0;
        d[i] = 1E18;
    }
    q.push(node(0, 1));
    d[1] = 0;
    while(!q.empty()){
        int u = q.top().id; q.pop();
        if(vis[u])  continue ;
        vis[u] = 1;
        for(int e = first[u]; e; e = nxt[e]){
            int v = to[e];
            if(d[v] > d[u] + w[e]){
                d[v] = d[u] + w[e];
                q.push(node(d[v], v));
            }
        }
    }
    for(int i = 2; i <= n; i++) {
        printf("%lld ", min(d[i], d[i + 3 * n]));
    }
    return 0;
}
```