解题报告

1. 相等字串(equal.cpp)

【考察算法】

模拟。

【解题思路】

由于可以任选子序列改成相同的一个字母，所以把要改成同一个字母的位置一次性改动肯定最优，最终答案即为改动的次数。

记录vis_i为是否需要把 A 中某一个非 i 的字母改为 i，对比 A 与 B 的每一位，若第 j 位不一样，则令$vis_{B_j} = 1$，所有vis数组之和即为答案。

时间复杂度$O(n)$

【参考代码】

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

char s1[100005], s2[100005];
int vis[26];

int main() {
    freopen("equal.in", "r", stdin);
    freopen("equal.out", "w", stdout);
    int T; cin >> T;
    while(T--) {
        for(int i = 0; i < 26; i++)  vis[i] = 0;
        scanf("%s", s1 + 1);
        scanf("%s", s2 + 1);
        int n = strlen(s1 + 1);
        for(int i = 1; i <= n; i++) {
            if(s1[i] != s2[i])  ++vis[s2[i] - 'a'];
        }
        int ans = 0;
        for(int i = 0; i < 26; i++)
            ans += (vis[i] > 0);
        cout << ans << endl;
    }
    return 0;
}
```

2. 石子染色(color.cpp)

【考察算法】

背包 DP

【解题思路】

当$n \le 20$时我们暴力枚举 S，再统计 f(S)，时间复杂度$O(n2^n)$

取走一个 S 后我们知道 f(S)=|R-B|，石子总数 X=R+B，那么 f(S)可以看作|X-2B|，假设所有元素和为 i 的 S 有s_i个，那么最终答案即为$\sum_{i=1}^{x} s_i|x-2i|$

考虑统计s_i，用背包处理即可，转移方程式$s_i = \sum_{a_k \le i} s_{i-a_k}$

时间复杂度$O(nS)$

【参考代码】

```
#include<bits/stdc++.h>
#define Mod 998244353

using namespace std;
int x, n, f[2005];

int main() {
    freopen("color.in", "r", stdin);
    freopen("color.out", "w", stdout);
    int T; cin >> T;
    while(T--) {
        memset(f, 0, sizeof(f)); f[0] = 1;
        cin >> x >> n;
        for(int i = 1; i <= n; i++) {
            int k; cin >> k;
            for(int j = x; j >= k; j--)
                f[j] = (f[j] + f[j - k]) % Mod;
        }
        long long ans = 0;
        for(int i = 0; i <= x; i++)
            ans = (ans + abs((long long)(x - 2 * i)) * f[i] % Mod) % Mod;
        cout << ans << endl;
    }
    return 0;
}
```

3．城市游历(tour.cpp)

【考察算法】

最小生成树，前缀和，离散化

【解题思路】

当$n \le 100$时我们暴力对每个询问的每个接受程度跑一遍即可，时间复杂度$O(n^3)$。

先跑出最小生成树，很明显在最小生成树上走最优。

当$l, r, k \le 10^5$时，考虑预处理f_i代表接受程度为 i 时能去到的不同种类数的景点，显然有$f_i \ge f_{i-1}$，答案即为$\sum_{i=l}^{r} f_i$，用前缀和计算即可。

由于 i 的范围过大，以 i 为下标存不下，由题目条件$c_i \le 600$，因此考虑离散化记录每一个$f_i < f_{i+1}$的转折点，在计算答案时稍加判断即可。

考虑f_i的统计方式，从起点 x 遍历整棵树，记录下 x 到每个点 j 路程中的最大困难系数 k，若走困难系数$< k$的边不会出现

与c_j相同的特征值，则 k 为转折点之一，将$f_k + 1$即可。

时间复杂度$O(q\, max\{c_i\})$

【参考代码】

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

int first[1000005], to[1000005], nxt[1000005], w[1000005], tot;

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

struct Edge {
    int x, y, z;
    bool operator < (Edge A) const {return z < A.z;}
}e[500005];

int n, m, q, x, opt, Mod, c[500005], lj[605], minn[605], fa[500005], mp[605];

int findfa(int x) {return fa[x] == x ? x : fa[x] = findfa(fa[x]);}

queue<int> q2;

void dfs(int u, int fa, int mx) {
    minn[c[u]] = min(minn[c[u]], mx);
    for(int e = first[u]; e; e = nxt[e]) {
        int v = to[e]; if(v == fa)  continue;
        dfs(v, u, max(mx, w[e]));
    }
}
```

```
int main() {
    freopen("tour.in", "r", stdin);
    freopen("tour.out", "w", stdout);
    memset(mp, 0x7f, sizeof(mp));
    memset(minn, 0x7f, sizeof(minn));
    scanf("%d %d %d %d",&n,&m,&q,&x) ;
    for(int i = 1; i <= n; i++)  scanf("%d", &c[i]), fa[i] = i;
    for(int i = 1; i <= m; i++)  scanf("%d%d%d", &e[i].x, &e[i].y, &e[i].z);
    sort(e + 1, e + m + 1);
    for(int i = 1; i <= m; i++) {
        int fx = findfa(e[i].x), fy = findfa(e[i].y);
        if(fx == fy)  continue;
        Add(e[i].x, e[i].y, e[i].z); Add(e[i].y, e[i].x, e[i].z);
        fa[fx] = fy;
    }
    dfs(x, 0, 0);
    int pos = 600;
    sort(minn + 1, minn + 601);
    for(int i = 1; i <= 600; i++) {
        if(minn[i] < 1E9)  ++lj[i], mp[i] = minn[i];
        else  {pos = i - 1; break;}
    }
    for(int i = 1; i <= 600; i++)  lj[i] += lj[i - 1];
    long long ans = 0;
    for(int i = 1; i <= q; i++) {
        int l, r;
        scanf("%d%d", &l, &r);
        if(l > r)  swap(l, r);
        ans = 0;
        for(int j = 1; j <= 600; j++) {
            if(mp[j + 1] >= l && mp[j] < l)  ans += (long long)(min(mp[j + 1], r + 1) - l) * lj[j];
            if(mp[j] >= l && mp[j] <= r) {
                ans += (long long)(min(mp[j + 1], r + 1) - mp[j]) * lj[j];
            }
        }
        printf("%lld\n", ans);
    }
    return 0;
}
```

4. **游戏闯关(game.cpp)**

【考察算法】

贪心，并查集

【解题思路】

由于玩家攻击力不变，故回合数固定，1 防减伤始终为回合数，我们令怪物 i 的一防减伤为e_i。

先考虑每个点度数为 1 的情况，这代表每次打怪不会出现必须先打 i 才能打 j 的情况。

假设打怪序列为$s_1, s_2, \ldots, s_n$，我们考虑交换s_i, s_{i+1}会产生的影响。

s_i前序列不变，s_{i+1}后防御不变，伤害不变，变化的只有s_i, s_{i+1}的伤害。

设打完s_{i-1}后s_i的伤害为 A，s_{i+1}的伤害为 B，那么交换前两怪总伤为$A + B - d_i \times e_{i+1}$，交换后为$A + B - d_{i+1} \times\ e_i$，若交换后更优，则$d_i \times e_{i+1} < d_{i+1} \times e_i$，即$\frac{e_i}{d_i} > \frac{e_{i+1}}{d_{i+1}}$。

所以我们计算出每个怪物的$\frac{e_i}{d_i}$的值，按$\frac{e_i}{d_i}$从小到大打即可，若相等则随便打。

我们定义一个怪物的特征值$f_i = \frac{e_i}{d_i}$

根据上述内容我们有性质 1：对于两个点 A,B，顺序 AB 优于 BA 的充要条件是$f_A < f_B$

接下来考虑每个点度数至多为 2 的情况，若一个点有后置点，那么打完它以后要么直接打后置点，要么打其它一些点以后再打后置点。

性质 2：如果 B 是 A 的后置点，且$f_B \le f_A$，则打完 A 后立即打 B 一定最优。

证明如下：如果最优顺序是 ACB，C 为一些怪的集合$\{s_1, s_2, \ldots, s_m\}$，那么由性质 1 可知$f_{s_k} \ge f_A \ge f_B$，那么 B 应该在 C 之前打更优，矛盾。

那么我们可以从叶子开始，如果一个点与其父节点满足性质 2，那么可以将其合并为一个大点，由于大点内部一定会连续全部打完，所以大点的$f = \frac{\sum e}{\sum d}$，继续往上处理即可。

合并完以后整棵树变成了一个大根堆，按特征值从小到大打即可。

树同样满足性质 2，不过如果 A 的子节点存在 B,C 等多点同时满足性质 2，那么应该优先打特征值更小的点，否则不满足性质 1。处理方式同上，具体实现如下：

暴力每次合并是$O(n^2)$的，不过不需要改变树的形态，只需要合并节点信息即可。

将所有节点的特征值放进一个小根堆，每次取堆顶节点 i，若 i 的父节点 j 还没有打，则合并 i 与 j 的参数，即$d_j \mathrel{+}= d_i, e_j \mathrel{+}= e_i$，过程用并查集实现即可。

若 i 的父节点 j 已经打过了，那么直接对 i 所在大点的内部节点进行模拟即可。

时间复杂度$O(n\,log\,n)$

【参考代码】

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

int first[400005], nxt[400005], to[400005], tot = 0;

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

long long fa[200005], b[200005], a[200005], d[200005], hh[200005], val[200005],
HH[200005], Val[200005], tim[200005];
int vis[200005], sc[200005];
int ffa[1000005];
long long ans;

int findfa(int x) {return (ffa[x] == x) ? x : ffa[x] = findfa(ffa[x]);}

void fight(int x) {
    ans += (a[x] - d[1]) * hh[x];
    d[1] += val[x];
}

void dfs(int u, int F) {
    fa[u] = F;
    for(int e = first[u]; e; e = nxt[e]) {
        int v = to[e];
        if(v == F)  continue;
        dfs(v, u);
    }
}

vector<int> Nxt[200005];

void Do(int u) {
    fight(u); sc[u] = 1;
    for(int i = 0; i < Nxt[u].size(); i++) {
        Do(Nxt[u][i]);
    }
}
```

```
int main() {
    freopen("game.in", "r", stdin);
    freopen("game.out", "w", stdout);
    priority_queue<pair<double, int> > q;
    int n; cin >> n;
    cin >> a[1] >> d[1];
    for(int i = 1; i < n; i++) {
        int x, y;
        scanf("%d%d", &x, &y);
        Add(x, y); Add(y, x);
    }
    dfs(1, 0);
    for(int i = 2; i <= n; i++) {
        scanf("%lld%lld%lld%lld", &a[i], &d[i], &b[i], &val[i]);
        hh[i] = b[i] / (a[1] - d[i]);  HH[i] = hh[i]; Val[i] = val[i];
        if(b[i] % (a[1] - d[i]) == 0)  --hh[i], --HH[i];
        q.push(make_pair(1.0 * val[i] / hh[i], i));
    }
    sc[1] = 1;
    for(int i = 1; i <= n; i++)  ffa[i] = i;
    while(!q.empty()) {
        int u = q.top().second; q.pop();
        if(vis[u])  continue;  vis[u] = 1;
        if(sc[fa[u]])  {Do(u); continue;}
        HH[findfa(fa[u])] += HH[u], Val[findfa(fa[u])] += Val[u];
        Nxt[ffa[fa[u]]].push_back(u);
        ffa[u] = ffa[fa[u]];
        q.push(make_pair(1.0 * Val[ffa[fa[u]]] / HH[ffa[fa[u]]], ffa[fa[u]]));
    }
    cout << ans << endl;
    return 0;
}
```