T1

因为最多两个物品一组，并且价格之和不能超过w，所以考虑贪心，排序后从两端向中间遍历，两端物品能一组就一组，不能就让价格高的单独一组。

复杂度$O(n\log n)$.

```cpp
#include <bits/stdc++.h>
#define int long long
#define vec vector
using namespace std;

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int w, n; cin >> w >> n;
    vec<int> a(n + 1);
    for (int i = 1; i <= n; ++i)
        cin >> a[i];

    sort(a.begin() + 1, a.begin() + 1 + n);
    int tot = 0, i, j;
    for (i = 1, j = n; i < j; )
        if (a[i] + a[j] <= w) {
            tot++;
            i++; j--;
        }
        else {
            j--;
            tot++;
        }
    cout << tot + (i == j) << '\n';
    return 0;
}
```

T2

数据比较大，所以map记一下每个数的个数，然后乘法原理计算即可。

复杂度$O(n)$.

```cpp
#include <bits/stdc++.h>
#define int long long
#define vec vector
using namespace std;

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, c, ans = 0; cin >> n >> c;
    vec<int> a(n + 1);
    map<int, int> cnt;
    map<int, bool> vis;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
        cnt[a[i]]++;
    }
    for (int i = 1; i <= n; ++i) {
        if (!vis[a[i]]) {
            vis[a[i]] = 1;
            ans += cnt[a[i]] * cnt[c + a[i]];
        }
    }
    cout << ans << '\n';
    return 0;
}
```

T3

根据题意进行排序,先按高度从大到小,如果高度一样,把编号小的排前面,然后暴力模拟即可。

复杂度$O(n^2)$.

```cpp
#include <bits/stdc++.h>
#define int long long
#define vec vector
using namespace std;

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n; cin >> n;
    vec<array<int, 4> > a(n + 1);
    for (int i = 1; i <= n; ++i) {
        cin >> a[i][0] >> a[i][1] >> a[i][2];
        a[i][3] = i;
    }

    auto cmp = [](const array<int, 4> &a, const array<int, 4> &b) {
        if (a[0] == b[0]) return a[3] < b[3];
        return a[0] > b[0];
    };

    sort(a.begin() + 1, a.begin() + 1 + n, cmp);
    vec<vec<int> > ans(n + 1, vec<int>(2));
    for (int i = 1; i <= n; ++i) {
        int l = a[i][1], r = a[i][2], h = a[i][0];
        bool flag = 0;
        for (int j = i + 1; j <= n; ++j) {
            if (a[j][0] < h && l > a[j][1] && l < a[j][2]) {
                ans[a[i][3]][0] = a[j][3];
                flag = 1;
```

```cpp
                    break;
                }
            }
            if (!flag) ans[a[i][3]][0] = 0;
            flag = 0;
            for (int j = i + 1; j <= n; ++j) {
                if (a[j][0] < h && r > a[j][1] && r < a[j][2]) {
                    ans[a[i][3]][1] = a[j][3];
                    flag = 1;
                    break;
                }
            }
            if (!flag) ans[a[i][3]][1] = 0;
        }
        for (int i = 1; i <= n; ++i) {
            cout << ans[i][0] << ' ' << ans[i][1] << '\n';
        }
        return 0;
}
```

T4

按修路时间排序，每次并查集合并两个村庄，因为数据比较小，所以合并后暴力检查是否联通即可。（后来发现当MST做就行，可以优化到$O(n\log n)$.）

复杂度$O(mn)$.

```cpp
#include <bits/stdc++.h>
#define int long long
#define vec vector
using namespace std;

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m; cin >> n >> m;
    vec<array<int, 3> > a(m + 1);
    vec<int> fa(n + 1);
    for (int i = 1; i <= n; ++i) fa[i] = i;
    for (int i = 1; i <= m; ++i) {
        cin >> a[i][0] >> a[i][1] >> a[i][2];
    }

    auto cmp = [](const array<int, 3> &a, const array<int, 3> &b) {
        return a[2] < b[2];
    };

    auto find = [&](auto find, int x) -> int {
        if (fa[x] == x) return x;
        return fa[x] = find(find, fa[x]);
    };

    sort(a.begin() + 1, a.begin() + 1 + m, cmp);
    for (int i = 1; i <= m; ++i) {
        int aa = find(find, a[i][0]), bb = find(find, a[i][1]);
        fa[aa] = bb;
        aa = find(find, a[i][0]);
        bool flag = 1;
        for (int j = 1; j <= n; ++j) {
            if (find(find, j) != aa) {
                flag = 0;
                break;
            }
        }
        if (flag) {
            cout << a[i][2] << '\n';
            return 0;
        }
    }
    cout << "-1\n";
    return 0;
}
```

T5

贪心，如果当前子段和为非负数，就把新的数加上；如果当前子段和为负数，那就舍弃前面的子段，从新的数开始。

复杂度$O(n)$.

```cpp
#include <bits/stdc++.h>
#define int long long
#define vec vector
using namespace std;

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, ans = (int)(-1e18), sum = 0; cin >> n;
    for (int i = 1; i <= n; ++i) {
        int x; cin >> x;
        if (sum >= 0) sum += x;
        else sum = x;
        ans = max(ans, sum);
    }
    cout << ans << '\n';
    return 0;
}
```