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 $O(n \log n)$.

```
#include <bits/stdc++.h>
 2
    #define int long long
 3
    #define vec vector
    using namespace std;
 5
 6
    signed main() {
        ios::sync_with_stdio(0);
 7
 8
        cin.tie(0);
 9
        int w, n; cin >> w >> n;
10
11
        vec<int> a(n + 1);
12
        for (int i = 1; i \leftarrow n; ++i)
13
             cin \gg a[i];
14
        sort(a.begin() + 1, a.begin() + 1 + n);
15
        int tot = 0, i, j;
16
17
        for (i = 1, j = n; i < j;)
18
             if (a[i] + a[j] <= w) {
19
                 tot++;
20
                 i++; j--;
21
             }
22
             else {
23
                 j--;
24
                 tot++;
25
26
        cout << tot + (i == j) << '\n';
27
        return 0;
28 }
```

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).

```
#include <bits/stdc++.h>
 2
   #define int long long
 3
   #define vec vector
4
   using namespace std;
 5
   signed main() {
 6
7
        ios::sync_with_stdio(0);
 8
        cin.tie(0);
9
10
        int n, c, ans = 0; cin >> n >> c;
```

```
11
        vec<int> a(n + 1);
12
        map<int, int> cnt;
13
        map<int, bool> vis;
        for (int i = 1; i <= n; ++i) {
14
15
             cin >> a[i];
             cnt[a[i]]++;
16
17
        }
18
        for (int i = 1; i <= n; ++i) {
19
            if (!vis[a[i]]) {
20
                 vis[a[i]] = 1;
21
                 ans += cnt[a[i]] * cnt[c + a[i]];
             }
22
23
        }
24
        cout << ans << '\n';</pre>
25
        return 0;
26 }
```

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 $O(n^2)$.

```
#include <bits/stdc++.h>
 2
    #define int long long
    #define vec vector
    using namespace std;
    signed main() {
 6
 7
        ios::sync_with_stdio(0);
8
        cin.tie(0);
9
10
        int n; cin >> n;
11
        vec<array<int, 4>> a(n + 1);
12
        for (int i = 1; i <= n; ++i) {
13
             cin >> a[i][0] >> a[i][1] >> a[i][2];
14
             a[i][3] = i;
15
        }
16
        auto cmp = [](const array<int, 4> &a, const array<int, 4> &b) {
17
18
             if (a[0] == b[0]) return a[3] < b[3];
19
             return a[0] > b[0];
20
        };
21
        sort(a.begin() + 1, a.begin() + 1 + n, cmp);
22
23
        \text{vec}(\text{vec}(\text{int})) \approx \text{ans}(n + 1, \text{vec}(\text{int}));
24
        for (int i = 1; i <= n; ++i) {
25
             int l = a[i][1], r = a[i][2], h = a[i][0];
26
             bool flag = 0;
27
             for (int j = i + 1; j \le n; ++j) {
28
                 if (a[j][0] < h && l > a[j][1] && l < a[j][2]) {
29
                      ans[a[i][3]][0] = a[j][3];
30
                      flag = 1;
```

```
31
                     break;
                 }
32
33
             }
             if (!flag) ans[a[i][3]][0] = 0;
34
35
             flag = 0;
             for (int j = i + 1; j <= n; ++j) {
36
37
                 if (a[j][0] < h && r > a[j][1] && r < a[j][2]) {
38
                     ans[a[i][3]][1] = a[j][3];
39
                     flag = 1;
40
                     break;
41
                 }
42
             }
43
             if (!flag) ans[a[i][3]][1] = 0;
44
        for (int i = 1; i <= n; ++i) {
45
             cout << ans[i][0] << ' ' << ans[i][1] << '\n';</pre>
46
47
48
        return 0;
49
```

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).

```
#include <bits/stdc++.h>
    #define int long long
 3
    #define vec vector
 4
    using namespace std;
 5
 6
    signed main() {
 7
        ios::sync_with_stdio(0);
 8
        cin.tie(0);
 9
10
        int n, m; cin >> n >> m;
        vec<array<int, 3 > a(m + 1);
11
12
        \text{vec} < \text{int} > \text{fa}(n + 1);
13
        for (int i = 1; i <= n; ++i) fa[i] = i;
14
        for (int i = 1; i <= m; ++i) {
15
             cin >> a[i][0] >> a[i][1] >> a[i][2];
16
        }
17
18
        auto cmp = [](const array<int, 3> &a, const array<int, 3> &b) {
19
             return a[2] < b[2];
20
        };
21
22
        auto find = [\&](auto find, int x) -> int {
23
             if (fa[x] == x) return x;
             return fa[x] = find(find, fa[x]);
24
25
        };
26
```

```
sort(a.begin() + 1, a.begin() + 1 + m, cmp);
27
        for (int i = 1; i <= m; ++i) {
28
29
             int aa = find(find, a[i][0]), bb = find(find, a[i][1]);
30
             fa[aa] = bb;
31
             aa = find(find, a[i][0]);
32
             bool flag = 1;
             for (int j = 1; j <= n; ++j) {
33
                 if (find(find, j) != aa) {
34
35
                     flag = 0;
36
                     break;
37
                 }
38
             }
39
             if (flag) {
40
                 cout << a[i][2] << '\n';</pre>
                 return 0;
41
             }
42
43
44
        cout << "-1\n";
        return 0;
45
46
```

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).

```
#include <bits/stdc++.h>
    #define int long long
 3
    #define vec vector
    using namespace std;
 5
 6
    signed main() {
 7
        ios::sync_with_stdio(0);
 8
        cin.tie(0);
 9
        int n, ans = (int)(-1e18), sum = 0; cin >> n;
10
11
        for (int i = 1; i <= n; ++i) {
12
            int x; cin >> x;
13
            if (sum >= 0) sum += x;
14
            else sum = x;
            ans = max(ans, sum);
15
        }
16
17
        cout << ans << '\n';
18
        return 0;
19 }
```