J-暴力

首先每种物品可以取很多次，说明普通 dp 会炸。

我们可以考虑第 k 小和第 $k-1$ 小（假设第 $k-1$ 重量为 x)的关系：

当前第 k 小的就是再取了一个物品，当前第 k 小重量就是 $x+a_i(i\in[1,n])$。

所以我们可以用一个 set 去维护整个重量，每次取出这个 set 中最小的数，即第一个数，将这个数加上 $a_i(i\in[1,n])$，在放回 set 中，我们再把最小的数删除。

正确性证明：

我们每次我们是取 set 中最小的数来更新，但是最小的值是不会重复的，所以每次我们都至少能拓展出一个没有出现过的值。

我们将这个操作做 k 次，那么答案一定能拓展出来。

```
void solve(){
    ll n, m;
    cin >> n >> m;
    set<ll> b;
    set<ll> st;
    ll Max = 0;
    for (int i = 0; i < n; i++){
        cin >> x;
        Max = max(Max, x);
        b.insert(x);
        st.insert(x);
    }
    queue<ll> q;
    for (auto i : b){
        q.push(i);
    }
    while (!q.empty()){
        ll tt = q.front();
        q.pop();
        if (tt>=*b.rbegin()&&b.size()>m) continue;
        for (auto i : st){
            if ((b.size() > m && tt + i > *b.rbegin()) || b.count(tt + i)){
                continue;
            }
            if ((b.size() > m && tt + i > *b.rbegin())) break;
            while (b.size() > m){
                b.erase(*b.rbegin());
            }
            Max = *b.rbegin();
            b.insert(tt + i);
            q.push(tt + i);
        }
    }
    while (b.size()>m){
        b.erase(*b.rbegin());

    auto p = b.rbegin();
    cout << *p;
```

```
}
```

L-线段树基础

大力分讨合并时的情况,记录最大/次大的大小及个数

```
// 核心分讨
struct info{
    int l, r;
    T sum1, lazy, Max1, Max2, sum2;
    info operator+(const info &a) const{
        info newone = *this;
        newone.r = a.r;
        if (newone.Max1 > a.Max1){
            if (newone.Max2 > a.Max1){}
            else{
                if (newone.Max2 == a.Max1)
                    newone.sum2 += a.sum1;
                else{
                    newone.Max2 = a.Max1;
                    newone.sum2 = a.sum1;
                }
            }
        }
        else{
            if (newone.Max1 == a.Max1){
                newone.sum1 += a.sum1;
                if (newone.Max2 > a.Max2){}
                else{
                    if (newone.Max2 == a.Max2)
                        newone.sum2 += a.sum2;
                    else{
                        newone.Max2 = a.Max2;
                        newone.sum2 = a.sum2;
                    }
                }
            }
            else{
                info tt = newone;
                tt.Max1 = a.Max1;
                tt.sum1 = a.sum1;
                if (newone.Max1>a.Max2){
                    tt.Max2 = newone.Max1;
                    tt.sum2 = newone.sum1;
                }else {
                    if (newone.Max1==a.Max2){
                        tt.Max2 = a.Max2;
                        tt.sum2 = a.sum2 + newone.sum1;
                    }else {
                        tt.Max2 = a.Max2;
                        tt.sum2 = a.sum2;
                    }
                }
                newone =  tt;
```

```
            }
        }
        return newone;
    }
};
```

参考代码