1904B - Collecting Game

Idea: **oursaco**
Preparation: **oursaco**
Analysis: **lunchbox**
Solution

Let's sort array a. The answer for the largest element is $n-1$ because the score, which is a_n, cannot be smaller than any of the other elements. Now, consider the second largest element. The answer is at least $n-2$ because every element that is not greater than a_{n-1} can be taken. Then, we check if the score is at least a_n. This inspires the following solution: first, we find the prefix sum p of array a. We calculate the answer in decreasing order of a_i. To calculate the answer for an a_i, we find the largest j such that $p_i \geq a_j$ and set the answer for i equal to the answer of j.

Code

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;

#define pb push_back
#define ff first
#define ss second

typedef long long ll;
typedef long double ld;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
typedef pair<ld, ld> pld;

const int INF = 1e9;
const ll LLINF = 1e18;
const int MOD = 1e9 + 7;

template<class K> using sset =  tree<K, null_type, less<K>, rb_tree_tag,
tree_order_statistics_node_update>;

inline ll ceil0(ll a, ll b) {
    return a / b + ((a ^ b) > 0 && a % b);
}

void setIO() {
    ios_base::sync_with_stdio(0); cin.tie(0);
}

int main(){
    setIO();
    int T;
```

```
    cin >> T;
    for(int tt = 1; tt <= T; tt++){
        int n;
        cin >> n;
        pii arr[n + 1];
        for(int i = 1; i <= n; i++) cin >> arr[i].ff, arr[i].ss = i;
        sort(arr + 1, arr + n + 1);
        int nxt[n + 1];
        ll sum[n + 1];
        int ans[n + 1];
        nxt[0] = sum[0] = 0;
        for(int i = 1; i <= n; i++){
            if(nxt[i - 1] >= i){
                nxt[i] = nxt[i - 1];
                sum[i] = sum[i - 1];
            } else {
                sum[i] = sum[i - 1] + arr[i].ff;
                nxt[i] = i;
                while(nxt[i] + 1 <= n && sum[i] >= arr[nxt[i] + 1].ff){
                    nxt[i]++;
                    sum[i] += arr[nxt[i]].ff;
                }
            }
            ans[arr[i].ss] = nxt[i];
        }
        for(int i = 1; i <= n; i++) cout << ans[i] - 1 << " ";
        cout << endl;
    }
}
```