A Удалить наименьшее

Нужно отсортировать значения. Дальше пройтись по всему массиву и проверить, чтобы разность соседей не превышала единицы.

Можно использовать set. Нужно проверить, чтобы максимальное значение минус минимальное не превышало размера set.

Решение C++

```
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
	int tst;
	cin >> tst;
	while (tst--){
		int n;
		cin >> n;
		int mas[111];
		for (int i = 0; i < n; i++){
			cin >> mas[i];
		}
		sort(mas, mas + n);
		bool ok = true;
		for (int i = 1; i < n; i++)
			if (mas[i] - mas[i - 1] > 1)
				ok = false;
		if (ok)
			cout << "yes" << endl; else
			cout << "no"<< endl;
	}
	return 0;
}
```

Решение C++ (set)

```
#include <bits/stdc++.h>
using namespace std;
int main() {
	int t;
	cin >> t;
	while(t--) {
		int n;
		cin >> n;
		set<int>s;
		for(int i=0;i<n;i++) {
			int x;
			cin >> x;
			s.insert(x);
		}
		if(*s.rbegin() - *s.begin() + 1 == s.size()) {
			cout << "YES\n";
		}
		else {
			cout << "NO\n";
		}
	}
}
```

Решение Python

```python
for _ in range(int(input())):
    n = int(input())
    a = set(map(int, input().split()))
    print('YES' if max(a)-min(a) < len(a) else 'NO')
```

B Missing Coin Sum

Решение перебором или методом динамического программирования не пройдут по времени.

Сначала отсортируем все числа по возрастанию.

Будем идти по всем числам по порядку и поддерживать диапазон значений, которые мы можем составить из предыдущих монет.

Вначале, если у нас нет 1, то ответ 1. Иначе мы можем составить любую сумму от 0 до 1.

Далее, если у на следующее число больше 2, то ответ 2. Иначе мы можем добавить новое число к любому значению суммы, полученному ранее. Таким образом на каждом шаге мы можем составить любую сумму от 0 до res. Если следующее число больше, чем res+1. То ответом будет res+1

Решение C++

```cpp
#include <iostream>
#include <algorithm>


using namespace std;

int arr[100100];

int main() {
	int res = 0;
	int n;
	cin >> n;
	for (int i = 0; i < n; i++)
		cin >> arr[i];

	sort(arr,arr+n);

	for (int i = 0; i < n; i++){
		if (arr[i] > res + 1){
			cout << res + 1;
			return 0;
		}
		res += val;
	}
	cout << res + 1;

	return 0;
}
```

C Movie Festival

Это классическая задача на поиск максимального количества непересекающихся отрезков на прямой.

Есть жадное решение. Отсортируем все отрезки по концу. Пройдем по всем отрезкам. И если новый отрезок начинается позже, чем ранее взятый, то берем его. Первый берем всегда. Попробуйте доказать, почему это всегда выгодно.

Для реализации можно использовать pair. Только первым значением будет конец отрезка, а вторым начало. Тогда при сортировке сравнением сначала будет автоматически производиться по первому значению.

Решение C++ (используются стандартные пары)

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#define ll long long

using namespace std;

ll t, n, x, y, a, b, c;
vector<pair<ll, ll> > v;

int main() {
	cin >> n;
	for (int i = 0; i < n; i++) {
		cin >> a >> b;
		v.push_back(pair<ll, ll>(b, a));
	}
	sort(v.begin(), v.end());
	c = 0;
	for (int i = 0; i < n; i++) {
		if (v[i].second >= c) {
			x++;
			c = v[i].first;
		}
	}
	cout << x;
}
```

Можно также использовать свою структуру данных и компаратор к ней.

Решение C++

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

struct Node {
	int a, b;
} a[200010];

int n, s, k;

bool cmp(Node a, Node b) {
	return a.b < b.b;
}
int main() {
	cin >> n;
	for (int i = 1; i <= n; ++i)
		cin >> a[i].a >> a[i].b;
	sort(a+1, a+n+1, cmp);
	s = 1;
	k = a[1].b;
	for (int i = 2; i <= n; ++i)
		if (a[i].a >= k) {
			k = a[i].b;
			++s;
		}
	cout << s << endl;
	return 0;
}
```

D Stick Lengths

Нужно было догадаться, что нам всегда нужно сводить длины всех отрезков к медиане. Медиана – отрезок, которые окажется в середине, после сортировки. Если отрезков четное число, то в качестве медианы можно выбрать любое значение между двумя отрезками в середине.

Попробуйте для начала рассмотреть пример с двумя отрезками. Потом перейдите к большему числу.

Решение C++

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

const int mxN = 1e6+5;
int a[mxN];

int main() {
    int n;
    cin >> n;
    for(int i = 0; i < n; i++)
        cin >> a[i];

    sort(a, a+n);

    long long md = a[n/2], ans = 0;
    for(int i = 0; i < n; i++)
```

```
            ans += abs(md-a[i]);

    cout << ans << "\n";

    return 0;
}
```

Решение Python

```
n = int(input())
a = sorted([int(i) for i in input().split()])
k = a[(n + 1) // 2 - 1]
ans = 0
for i in a:
    ans += abs(i - k)
print(ans)
```

E Reading Books

Пусть первый читает самую долгую книгу. А второй все остальные.

Далее 2 варианта развития событий. Если второй не успел дочитать все остальные книги до конца, то ответом будем просто сумма всех чисел. То есть они могут распределить так книги между собой, что не потребуется дополнительного времени ожидания.

Второй вариант, когда второй успел дочитать все остальные книги. То есть длительность чтения самой большой книги больше длительности всех остальных книг. Тогда ответом будет 2*max_element.

Решение Python

```
n=int(input())
l=list(map(int, input().split()))
x=sum(l)
y=max(l)
if 2*y>x:
    print(2*y)
else:
    print(x)
```

F Collecting Numbers

Для каждого числа запомним в отдельном ассоциативном массиве его позицию.

Далее будем моделировать процесс. Пройдем по всем числам по порядку. Если следующее число находится левее чем предыдущее, то увеличиваем ответ.

Решение C++

```
#include <bits/stdc++.h>
using namespace std;
unsigned long long s;
int n, a[200010], b[200010];
int main() {
      cin>>n;
      for(int i =1; i <=n; i++) {
            cin>>a[i];
            b[a[i]]=i;
      }
      for(int i=2; i<=n; ++i)
            if (b[i]<b[i-1])
                  s++;

      cout << s+1<< endl;

      return 0;
}
```

G Array Division

Бинарный поиск по ответу.

Перебираем бинарным поиском максимальную сумму. Для фиксированной суммы жадно производим разбиение массива на максимальное число частей и сравниваем с заданным количеством k.

Решение C++

```
#include <iostream>
#include <vector>

int main()
{
    size_t n;
    std::cin >> n;

    unsigned k;
    std::cin >> k;

    std::vector<unsigned> x(n);
    for (size_t i = 0; i < n; ++i)
        std::cin >> x[i];

    const auto f = [&](uint64_t b) -> uint64_t {
        uint64_t r = 0, c = 0;
        for (const unsigned d : x) {
```

```
            if (d > r) {
                ++c;
                r = b;
            }
            if (d > r)
                return k + 1;

            r -= d;
        }
        return c;
    };

    uint64_t l = 0, r = ~0ull;
    while (l+1<r) {
        const uint64_t mid = (l+r)/2;
        if (f(mid) > k)
            l = mid;
        else
            r = mid;
    }

    std::cout << l << '\n';

    return 0;
}
```

H Sum of Four Values

Можно применить метод meet-in-the-middle. Тогда вместо полного перебора за $O(N^4)$ получится решение с асимптотикой $O(N^2 \log N)$

https://neerc.ifmo.ru/wiki/index.php?title=Meet-in-the-middle

Решение C++

```
#include<cstdio>
#include<map>
#include<iostream>
using namespace std;

map<int, pair<int, int> >mp;
int a[1005];

int main(){
      int n, i, j, m;
      map<int, pair<int, int> >::iterator it;

      scanf("%d%d", &n, &m);

      for(i = 1;i <= n;i++)
            scanf("%d", &a[i]);

      for(i = 1;i <= n;i++){
            for(j = i + 1;j <= n;j++){
                  it = mp.find(m - a[i] - a[j]);
                  if(it != mp.end()){
```

```
                        printf("%d %d %d %d", it->second.first, it-
>second.second, i, j);
                        return 0;
                    }
                }
                for(j = 1;j < i;j++)mp[a[i] + a[j]] = make_pair(j, i);
        }

        printf("-1");

        return 0;
}
```

I Collecting Numbers II

Решение C++

```
#include <bits/stdc++.h>
using namespace __gnu_pbds;
using namespace std;
using ld = long double;
using ll = long long;

int n, m, pos[200001];
pair<int, int> x[200001];

int check(int a, int b){
    if(a > b) swap(a, b);
    int ret = (x[a].second < x[a - 1].second);
    ret += (x[b + 1].second < x[b].second);
    ret += (x[b].second < x[b - 1].second);
    ret += (a + 1 != b && x[a + 1].second < x[a].second);
    return ret;
}

int main(){
    //IO("input.txt", "output.txt");
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cin >> n >> m;
    for(int i = 1; i <= n; i++){
        cin >> x[i].first, x[i].second = i;
        pos[i] = x[i].first;
    }
    sort(x, x + n + 1);
    int ans = 1;
    for(int i = 1; i <= n; i++)
        ans += (x[i].second < x[i - 1].second);
    for(int i = 0; i < m; i++){
        int a, b;
        cin >> a >> b;
        int va = pos[a], vb = pos[b];
        ans -= check(va, vb);
        swap(pos[a], pos[b]);
        x[va].second = b, x[vb].second = a;
        ans += check(va, vb);
        cout << ans << '\n';
    }
    return 0;
}
```