Tác giả: Kiều Minh Vương, Kiều Gia Huy, Nguyễn Đức Huy 11 Tin, Ngô Viết Tình 10 Tin

CÂU 1

Tóm tắt đề bài: Cho 4 số nguyên dương a, b, n, m. Tính hiệu số của số lượng số nằm trong đoạn [a, b] chia hết cho n và số lượng số nằm trong đoạn [a, b] chia hết cho m.

Subtask 1: a, b $\leq 10^6$.

- Gọi số lượng số chia hết cho n là: x, số lượng số chia hết cho m là: y.

Ta sẽ duyệt qua các số từ a đến b, tăng x với các số chia hết cho n, tăng y với các số chia hết cho m.

Đáp án bài toán sẽ là x – y.

- Độ phức tạp: O(b – a +1) ≈ O(10^6) (trường hợp tệ nhất).
- Code minh họa:

```
#include <bits/stdc++.h>


using namespace std;

int a, b, n, m;

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cin >> a >> b >> n >> m;
    int x = 0, y = 0;
    for(int i = a; i <= b; ++i){
        if (i % n == 0) x++;
        if (i % m == 0) y++;
    }
    cout << x - y << '\n';
    return 0;
```

```
}
```

Subtask 2: $10^6 \leq a, b \leq 2 * 10^9$.

· Để tính các số chia hết cho x trong đoạn [l, r]. Gọi ans(i) là các số chia hết cho x trong đoạn [1, i]. Ta nhận ra đáp án sẽ là ans(r) – ans(l – 1).

Và ans(i) có thể tính dễ dàng: ans(i) = i / x. (Độc giả có thể tự chứng minh).

Áp dụng vào bài toán, ta sẽ dễ dàng tính được:

x = b / n – (a – 1) / n.

y = b / m – (a – 1) / m.

· Độ phức tạp: O(1).

· Code minh họa:

```
#include <bits/stdc++.h>

using namespace std;

int a, b, n, m;

int main(){
	ios_base::sync_with_stdio(0);
	cin.tie(0);
	cin >> a >> b >> n >> m;
	 int x = b / n - (a - 1) / n;
	int y = b / m - (a - 1) / m;
	cout << x - y << '\n';
	return 0;
}
```

CÂU 2

Tóm tắt đề:

Cho mảng a gồm n phần tử và q truy vấn ($n,\ q,\ a_i \leq 10^6$). Mỗi truy vấn gồm 2 số l, r. Yêu cầu mỗi truy vấn tính tổng số lượng ước của các số $a_l,\ a_{l+1},\ a_{l+2}, \dots,\ a_r$.

Subtask 1 : $n \times q \leq 10^4, a_i \leq 100$.

- Với mỗi truy vấn ta duyệt for từ l đến r sau đó for thêm 1 lần nữa từ 1 đến a_i để tính số lượng ước của từng số.
- Độ phức tạp: $O(n \times q \times a_i)$.
- Code minh họa:

```
// Hàm đếm số lượng ước của số nguyên dương x
int count_div(int x) {
  int cnt = 0;
  for (int i = 1; i <= x; ++i) if (x % i == 0) cnt++;
  return cnt;
}

// Mỗi truy vấn
 int l, r;
 cin >> l >> r;
 int res = 0;

 for (int i = l; i <= r; ++i) {
   res += count_div(a[i]);
 }
 cout << res << '\n';
```

Subtask 2: $n \times q \leq 10^4$.

- Với mỗi truy vấn vẫn for từ l đến r sau đó tính số lượng ước của từng số nhưng cải tiến bằng việc sử dụng sàng nguyên tố giúp giảm thời gian tính toán.
- Độ phức tạp: $O(n \log \log n + n \times q)$.
- Code minh họa:

```
 int cnt_div[maxN]; // cnt_div[i] là số lượng ước của số i.

 // sàng để tính số lượng ước của các số trong đoạn 1 đến n.
 for (int i = 1; i <= n; ++i) {
   for (int j = i; j <= n; j += i) cnt_div[j]++;
 }
```

```
// Mỗi truy vấn
int l, r;
cin >> l >> r;
int res = 0;

for (int i = l; i <= r; ++i) {
  res += cnt_div[a[i]];
}
cout << res << '\n';
```

Subtask 3: Không có ràng buộc gì thêm.

- Gọi b_i là số lượng ước của số a_i, có thể tính bằng sàng nguyên tố một cách nhanh chóng.
- Với mỗi truy vấn bây giờ chính là tính tổng của $b_l + b_{l+1} + ... + b_r$. Có thể dùng mảng cộng dồn để tính trước giá trị này và xử lý truy vấn trong $O(1)$.
- Độ phức tạp: $O(n \log \log n + q)$.
- Code minh họa:

```
int cnt_div[maxN]; // cnt_div[i] là số lượng ước của số i.

// sàng để tính số lượng ước của các số trong đoạn 1 đến n.
for (int i = 1; i <= n; ++i) {
  for (int j = i; j <= n; j += i) cnt_div[j]++;
}

// Xử lý trước
for (int i = 1; i <= n; ++i) b[i] = cnt_div[a[i]];
for (int i = 1; i <= n; ++i) sum[i] = sum[i - 1] + b[i];

// Mỗi truy vấn
int l, r;
cin >> l >> r;
cout << sum[r] - sum[l - 1] << '\n';
```

Code minh họa: https://ideone.com/LYszBM

CÂU 3

Đề bài:

Có 1 seri truyện dài n chương, chương thứ i có a_i. Nhà sản xuất muốn chia n chương thành k tập sao cho số trang của tập dày nhất là ít nhất.

Xác định số trang của tập dày nhất.

Input:

Dòng đầu chứa 2 số nguyên n, k ($1 \leq k \leq n \leq 10^5$).

Dòng thứ 2 chứa n số nguyên $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$, $1 \leq i \leq n$).

Output: số trang của tập dày nhất.

Subtask 1: ($n \leq 60$; k = 1 hoặc k = 2 hoặc k = n, $a_i \leq 10^2$)

* Với k = 1, ta in ra tổng của n chương

* Với k = 2, ta chia n chương thành 2 tập và kết quả là tập có nhiều trang hơn.

. Gọi sum là tổng số trang của n chương.

. dp[i] là tổng tiền tố của n chương.

. Ta chỉ cần 1 vòng for từ 1 đến n và số trang lớn nhất khi ta chia thành 2 tập từ 1 đến i và từ i + 1 đến n là max(dp[i], sum – dp[i]).

* Với k = n, ta in ra chương có số trang lớn nhất.

Độ phức tạp: O(n).

Code minh họa: https://ideone.com/GjTSXu

Subtask 2: ($n < 60$; $k < 4$, $a_i \leq 10^3$)

* Với k = 1 hoặc 2, tương tự như subtask 1.

* Với k = 3, ta thêm 1 mảng pre[i] là hậu tố của n chương.

. Ta chia n chương thành 3 đoạn là từ 1 đến i, i + 1 đến j và từ j + 1 đến n và tập có nhiều trang nhất là max(dp[i], sum – dp[i] – pre[j], pre[j]).

Độ phức tạp: O(n * n) hoặc O(n).

Code minh họa: https://ideone.com/iJAlZb

Subtask 3 + 4: (n = 10^5; k ≈ 10^4; $a_i \leq 10^9$)

. Ta nhận thấy rằng 1 tập tối đa có sum trang và ít nhất maxa_i trang.

. Và sum ≤ 10^{14} nên ta sử dụng phương pháp chặt nhị phân kết quả.

. Hàm check(val) của chúng ta sử dụng thuật toán tham lam để kiểm tra xem với giá trị tập lớn nhất ≤ val, ta có thể chia thành ít hơn hoặc bằng k tập hay không.

Nếu có thể chia ít hơn hoặc bằng k tập thì đồng nghĩa với việc chia thành k tập.

Độ phức tạp: O(Log(1e14) * n).

Code minh họa: https://ideone.com/5lmvwV

CÂU 4

Tóm tắt đề bài: Có X cái kẹo rời và N hộp kẹo ($0 \leq X,\ N \leq 1000$), hộp kẹo thứ i có a_i cái ($1 \leq i \leq N,\ 0 \leq a_i \leq 1000$).Lấy thêm kẹo từ một số hộp kẹo sao cho số kẹo sau khi lấy có đủ Y cái kẹo ($0 \leq Y \leq 1000$) và số hộp kẹo lấy thêm là nhiều nhất có thể. Tìm số hộp kẹo lớn nhất có thể lấy.

Subtask 1: $N \leq 20$

- Ta sử dụng thuật toán quay lui để xét tất cả các trường hợp có thể xảy ra để lấy kẹo.
- Độ phức tạp: $O\left(2^N\right)$.
- Code minh họa:

```
#include<bits/stdc++.h>

using namespace std;

int n, x, y;
int a[100005];

int ans = -1;

void quaylui(int i, int sum, int cnt){
    if (i == n + 1){
        if (sum == y){
            ans = max(ans, cnt);
            return;
```

```
        }
        return;
    }

    quaylui(i + 1, sum, cnt);
    quaylui(i + 1, sum + a[i], cnt + 1);
}

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

    freopen("cau4.inp","r",stdin);
    freopen("cau4.out","w",stdout);

    cin >> n >> x >> y;

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

    quaylui(1, x, 0);

    cout << ans;

    return 0;
```

```
}
```

Subtask 2: $N \leq 1000$

- Ta sử dụng thuật toán quy hoạch động để ghi lại kết quả của trạng thái quay lui.
- Đặt $M = Y - X$, bài toán trở thành tìm cách lấy nhiều hộp kẹo nhất để có tổng bằng M.
- Gọi $f(i, m)$ là số hộp kẹo nhiều nhất có thể lấy khi xét đến hộp kẹo thứ i và cần phải lấy thêm m cái kẹo. Khi đó ta có nhận xét:
- Ta có thể lấy tiếp hộp kẹo thứ i hoặc bỏ qua nó.
- Nếu lấy tiếp, số kẹo cần lấy thêm là $m - a_i$, chuyển sang trạng thái $f(i + 1, m - a_i)$
- Nếu bỏ qua, số kẹo cần lấy vẫn là m và trạng thái là $f(i + 1, m)$
- Công thức: $f(i, m) = max(1 + f(i + 1, m - a_i), f(i + 1, m))$
- Đáp án của bài sẽ là $f(1, M)$
- Độ phức tạp: $O(N * (Y - X))$.
- Code minh họa:

```
#include<bits/stdc++.h>

using namespace std;

const int oo = 1e9;

int n, x, y;
int a[100005];
```

```
int dp[1005][1005];

int f(int i, int sum){
    if (sum < 0)
        return - oo;
    if (i == n + 1){
        if (sum == 0)
            return 0;
        return - oo;
    }

    if (dp[i][sum] != -1)
        return dp[i][sum];

    int cur = - oo;

    cur = max(cur, f(i + 1, sum));

    cur = max(cur, 1 + f(i + 1, sum - a[i]));

    return dp[i][sum] = cur;
}

signed main(){
```

```
    ios_base::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);

    freopen("cau4.inp","r",stdin);
    freopen("cau4.out","w",stdout);

    cin >> n >> x >> y;

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

    memset(dp, -1, sizeof(dp));

    cout << max(-1, f(1, y - x));

    return 0;
}
```

Subtask 3: $X = Y$

- Khi đó, ta thấy $M = 0$, khi đó ta chỉ có thể lấy được các hộp kẹo rỗng, tức là các hộp kẹo có $a_i = 0$.
- Để số hộp kẹo lấy được là nhiều nhất, ta sẽ lấy tất cả các hộp kẹo rỗng và bỏ qua các hộp kẹo không rỗng.
- Độ phức tạp: $O(N)$
- Code minh họa:

```
#include<bits/stdc++.h>
```

```
using namespace std;

int n, x, y;
int a[100005];

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

    freopen("cau4.inp","r",stdin);
    freopen("cau4.out","w",stdout);

    cin >> n >> x >> y;

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

    int dem = 0;

    for (int i = 1; i <= n; i ++)
        if (a[i] == 0)
            dem ++;

    cout << dem;

    return 0;
}
```

Subtask 4: $N \leq 10^5$

- Đầu tiên, ta tham lam lấy hết các hộp kẹo rỗng trước.
- Vì giới hạn của M và a_i khá bé, có thể mảng đếm phân phối 2 chiều để quản lý nên thay vì quan tâm đến từng hộp kẹo, ta chỉ cần quan tâm số lượng hộp kẹo có số kẹo bằng nhau
- Ta sử dụng thuật toán quy hoạch động để giải quyết bài toán
- Gọi $f(i, sum)$ là số hộp kẹo nhiều nhất có thể lấy khi đã lấy được sum cái kẹo và đang xét đến những hộp kẹo có i cái

- Gọi cnt_i là số hộp kẹo có i cái kẹo
- Ta chạy for j từ 0 đến cnt_i để xét số hộp kẹo có i cái kẹo sẽ lấy
- ĐPT của vòng for kết hợp quy hoạch động rất lớn, tuy nhiên nếu ta thêm điểm dừng của vòng for hợp lý sẽ giảm được ĐPT của vòng for là ĐPT của sàng số nguyên tố vì ta chỉ xét các số j thỏa mãn $sum + i * j \leq M$
- Độ phức tạp: $O(M^2 * log_2 M)$
- Code minh họa:

```
#include<bits/stdc++.h>

using namespace std;

const int oo = 1e9;

int n, x, y;
int a[100005];

int cnt[1005];
int dp[1005][1005];
int m;

int f(int i, int sum){
    if (i == 1001){
        if (sum == m)
            return 0;
        return - oo;
    }

    if (dp[i][sum] != -1)
        return dp[i][sum];
    int cur = - oo;

    for (int j = 0; j <= cnt[i] && sum + j * i <= m; j ++){
        cur = max(cur, j + f(i + 1, sum + j * i));
    }

    return dp[i][sum] = cur;
}
```

```
signed main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);

    freopen("cau4.inp","r",stdin);
    freopen("cau4.out","w",stdout);

    cin >> n >> x >> y;
    m = y - x;

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

    for (int i = 1; i <= n; i ++)
        cnt[a[i]] ++;

    memset(dp, -1, sizeof(dp));

    int res = f(1, 0);

    if (res >= 0)
        cout << res + cnt[0];
    else
        cout << -1;

    return 0;
}
```

Code tham khảo: https://ideone.com/GbAES8