TỔ CHỨC THE GIFTED BATTLEFIELD
DỰ ÁN INTELLECTUAL SHOWDOWN

Thi thử HSGTP Lớp 9 - Lần 1
NĂM HỌC 2023 – 2024

Ngày thi: **03/03/2024**
Thời gian làm bài: **180 phút** (không kể thời gian phát đề)
Đề thi gồm 05 trang, 03 bài

Tổng quan đề thi

	Tên bài	Tập tin dữ liệu	Tập tin kết quả	Hạn chế thời gian	Hạn chế bộ nhớ
Bài 1	Hệ tam phân	TAMPHAN.INP	TAMPHAN.OUT	2 giây	512MB
Bài 2	Tặng quà	GIFT.INP	GIFT.OUT	2 giây	512MB
Bài 3	Trò chơi điện tử	GAME.OUT	GAME.INP	2 giây	512MB

Bài 1. Hệ tam phân - TAMPHAN

Subtask 1

Xét tất cả giá trị tổng có thể được tạo ra (chỉ nằm trong miền $[2, 8]$), xét trường hợp cho từng giá trị để in ra số lượng chữ số bé nhất trong biểu diễn tam phân của các giá trị đó.

Subtask 2

Duyệt qua $n \times (n-1)/2$ cặp phần tử (a_i, a_j), tìm giá trị $f(a_i + a_j)$ nhỏ nhất.
Bước xây dựng hàm $f(x)$: Với mọi số nguyên dương x, đáp án của hàm $f(x)$ chính là số nguyên dương k bé nhất sao cho $3^k > x$ (nói cách khác, k là giá trị sao cho $3^k > x$ nhưng $x \geq 3^{k-1}$).

Ví dụ: $f(5) = 2$ vì $3^2 > 5$ và 2 là giá trị nhỏ nhất thỏa bất phương trình trên (biểu diễn tam phân của 5 là "12"). Tương tự, $f(25) = 3$ (biểu diễn tam phân của 25 là "221").
Lý do: số có y chữ số trong hệ tam phân sẽ biểu diễn được các giá trị trong khoảng $[3^{y-1}, 3^y - 1]$. Tức là, với một giá trị x có đúng y chữ số trong biểu diễn tam phân của nó, x sẽ thỏa: $3^{y-1} \leq x < 3^y$. $\Rightarrow$ Thỏa điều kiện trước đó.
Độ phức tạp: $O(n^2)$

Subtask 3

Nhận xét: Nếu $A \leq B$ thì $f(A) \leq f(B)$. $\Rightarrow f(a_x + a_y)$, với tổng $a_x + a_y$ là nhỏ nhất trong tất cả các cặp $a_i + a_j$ $(\forall i, j : 1 \leq i, j \leq n$ và $i \neq j)$, sẽ là đáp án của bài toán. Nói cách khác, ta cần phải tìm được cặp phần tử có tổng nhỏ nhất trong mảng a. Ta nhận thấy rằng, tổng của phần tử nhỏ nhất và nhỏ thứ hai trong mảng a chính là giá trị cần tìm trên. Để kiếm được hai phần tử trên, một cách làm là: sắp xếp lại mảng a theo thứ tự từ bé đến lớn, phần tử a_1 và a_2 lần lượt chính là phần tử nhỏ nhất và nhỏ nhì của mảng a.
Như vậy, bước cuối cùng sẽ là xây dựng hàm $f(x)$ - trả về số lượng chữ số trong biểu diễn tam phân của số nguyên dương x (xem phần "xây dựng hàm $f(x)$"ở mục subtask 2).
Độ phức tạp: $O(n)$.

Code mẫu:

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

int n;
```

```cpp

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);
  cin >> n;
  int min_1 = 1e9, min_2 = 1e9;
  for (int i = 1; i <= n; i++) {
    int a; cin >> a;
    if(a <= min_1) {
        min_2 = min_1;
        min_1 = a;
    } else min_2 = min(min_2, a);
  }

  int cnt = 0, sum = min_1 + min_2;
  while (sum > 0) {
    sum /= 3;
    cnt++;
  } //cach lam tren van cho ra ket qua tuong tu nhu huong dan giai: cnt be nhat sao cho 3^
    cnt > sum

  cout << cnt;
}
```

Bài 2. Tặng quà - GIFT

Subtask 1

Nhận xét: Khi tất cả bạn nhỏ đều được phát số quà bằng với số quà của người được nhiều quà nhất sau q lượt phát, mọi người sẽ hài lòng và đó cũng là phương án tối ưu của bác Hai để phát thêm ít quà nhất.

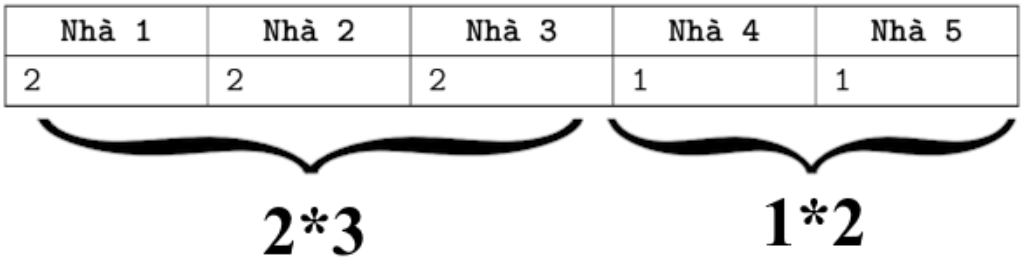
Tạo một mảng số nguyên c với toàn bộ phần tử khởi tạo bằng 0. Với mỗi lượt phát quà, ta chạy vòng lặp từ a đến b để cộng 1 vào các phần tử đi qua. Sau tất cả q lượt phát quà, c_i sẽ mang ý nghĩa là tổng số quà mà bạn thứ i đã nhận được. Tìm phần tử lớn nhất $maxGift$ trong mảng c. Đáp án bài toán chính là $n \times maxGift - sum$, với sum là tổng tất cả giá trị trong mảng c.
Độ phức tạp: $O(n \times q)$.

Subtask 2

Ta có thể áp dụng kiến thức **Mảng hiệu** và **Mảng đánh dấu** cơ bản. Tạo một mảng arr có độ lớn $n+1$, với mỗi truy vấn lượt phát quà, ta cộng 1 vào phần tử thứ a và trừ 1 vào phần tử thứ $b+1$. Nói cách khác, với mỗi cặp (a, b), ta thực hiện thao tác `arr[a]++` và `arr[b+1]-`. Sau đó, chạy vòng lặp với biến i từ 2 đến $n+1$ một lần, với mỗi i, ta thực hiện thao tác `arr[i] += arr[i-1]`.
Sau các bước trên, mảng arr sẽ có ý nghĩa tương tự như mảng c ở subtask 1. Ta sẽ sử dụng những công thức đã được trình bày ở trên để tính ra kết quả.
Độ phức tạp: $O(n+q)$.

Subtask 3

Ta có thể thấy một dãy nhà có số quà giống nhau có thể tính đơn giản hơn bằng cách nhân số quà cần có cho số nhà.



Vì thế, ta có thể cải thiện kĩ thuật **Mảng hiệu** trên về mặt thời gian và bộ nhớ bằng cách sử dụng `map<int, int>` hoặc `vector<pair<int, int»`. Ở ví dụ này, chúng ta sẽ sử dụng `map` để giải quyết bài toán. Cho một `map<int,int> house`. Với mỗi lượt phát quà, ta vẫn cộng cho a và trừ cho $b+1$ (`house[a]++` và `house[b+1]-`), sau đó cũng duyệt qua `house` một lần để biến `house` có chức năng tương tự như mảng c ở subtask 1.
Với ví dụ mẫu trên đề bài, ta có thể tạo ra được một `map`:

Key	**house[1]**	**house[3]**	**house[4]**	**house[6]**
Value	2	0	-1	-1

Sau khi duyệt qua `map` một lần nữa, ta có:

Key	**house[1]**	**house[3]**	**house[4]**	**house[6]**
Value	2	2	1	0

Ta tính được số lượng quà lớn nhất $maxGift$ lúc đó là 2.
Trong lúc duyệt qua q lần đi, ta cũng tính được tổng số quà đã được phát bằng cách cộng $b - a + 1$ vào $sumGift$.
Vì ta biết mọi nhà trong xóm cần có số quà lớn nhất, nên số lượng quà cần còn lại được tính theo công thức:

$$missingGift = maxGift \times n - sumGift$$

...Sau đó tiếp tục cộng dồn vào những vòng lặp còn lại.
Độ phức tạp: $O(q \log)$.

Code mẫu:

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

using namespace std;

int n, q;
long long maxGift = 0, missingGift = 0, sumGift = 0, prevHouse = 0;
map<int, int> house;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    freopen("GIFT.INP", "r", stdin);
    freopen("GIFT.OUT", "w", stdout);

    cin >> n >> q;
    while (q--) {
        int a, b;
        cin >> a >> b;
        // Đánh dấu
        house[a]++;
        house[b + 1]--;
        sumGift += b - a + 1;
    }

    for (auto curHouse: house) {
        // Duyệt qua mảng để xác định số quà
        curHouse.second += prevHouse;
        prevHouse = curHouse.second;
        // Đồng thời tìm số quà lớn nhất
        maxGift = max(maxGift, prevHouse);
    }
    // Tìm số quà còn thiếu
    missingGift = maxGift * n - sumGift;

    cout << missingGift << '\n';
```

```
37
38      return 0;
39  }
```

Bài 3. Trò chơi điện tử - GAME

Subtask 1

Vì An phải diệt hết tất cả mọi quái vật, nên đáp án của subtask này chính là $max(x_i) + max(y_i) + max(z_i)$ $(\forall i : 1 \leq i \leq n)$.

Subtask 2

Duyệt hết tất cả cấu hình (u, v, w) có thể của siêu anh hùng trong $O(x \times y \times z)$. Với mỗi siêu anh hùng, duyệt qua mảng quái vật ban đầu trong $O(n)$ để đếm số lượng quái vật mà cấu hình siêu anh hùng (u, v, w) hiện tại có thể chiến thắng, nếu chiến thắng được ít nhất k quái vật thì cập nhật $ans = min(ans, u + v + w)$.
Độ phức tạp: $O(n \times x \times y \times z)$.

Subtask 3

Vì các quái vật đều có chỉ số phòng thủ là 1, siêu anh hùng được chọn cũng chỉ cần chỉ số phòng thủ bằng 1. Khi đó, ta chỉ quan tâm đến giá trị x, y của các quái vật.
Ý tưởng: Mảng cộng dồn 2D.
Giả sử ta có hàm $f(i, j)$ trả về số lượng quái vật có (x, y) sao cho $x \leq i$ và $y \leq j$, ta có thể duyệt qua $x \times y$ cấu hình siêu anh hùng khác nhau để kiểm tra siêu anh hùng đó có chiến thắng được ít nhất k quái vật hay không, và cập nhật kết quả nếu thỏa điều kiện.
Ta có thể xây dựng mảng cộng dồn 2D $f[i][j]$ từ trước, đóng vai trò như hàm $f(i, j)$ để đếm số quái vật có $x \leq i$ và $y \leq j$.
Độ phức tạp: $O(n + x \times y)$.

Subtask 4

Phát triển từ ý tưởng của subtask 3, ta sử dụng **mảng cộng dồn 3D** $f[i][j][l]$ để đếm số lượng quái vật có $x \leq i, y \leq j, z \leq l$, sau đó duyệt qua mọi cấu hình siêu anh hùng (u, v, w) trong độ phức tạp $O(x \times y \times z)$, kiểm tra điều kiện $f[u][v][w] \geq k$ rồi cập nhật kết quả $ans = min(ans, u + v + w)$ nếu thỏa.
Công thức cho bài toán mảng cộng dồn 3D có thể được suy ra từ nguyên tắc bao hàm-loại trừ (đây là kiến thức tương đối nâng cao, các bạn THCS chưa cần quan tâm) hoặc đơn giản từ việc tưởng tượng ra khối hình hộp 3D để suy ra các công thức.
Công thức để xây dựng mảng cộng dồn 3D là:

$$f[i][j][l] = f[i-1][j][k]+f[i][j-1][k]+f[i][j][k-1]-f[i-1][j-1][k]-f[i-1][j][k-1]-f[i][j-1][k-1]+f[i-1][j-1][k-1]$$

Độ phức tạp: $O(x \times y \times z)$.

Code mẫu:

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

const int N = 5e5, M = 3e2;

int n, k;
int cnt[M + 5][M + 5][M + 5];

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);

```

```
  cin >> n >> k;
  for (int i = 1; i <= n; i++) {
    int x, y, z;
    cin >> x >> y >> z;
    cnt[x][y][z]++;
  }

  int res = 1e9;
  for (int x = 1; x <= M; x++) {
    for (int y = 1; y <= M; y++) {
      for (int z = 1; z <= M; z++) {
        cnt[x][y][z] += cnt[x - 1][y][z] + cnt[x][y - 1][z] + cnt[x][y][z - 1] - cnt[x - 1][y - 1][z] - cnt[x - 1][y][z - 1] - cnt[x][y - 1][z - 1] + cnt[x - 1][y - 1][z - 1];
        if (cnt[x][y][z] >= k) res = min(res, x + y + z);
      }
    }
  }

  cout << res;
}
```