2020 網際網路程式設計全國大賽
高中組決賽

- 本次比賽共 8 題，含本封面共 22 頁。
- 全部題目的輸入都來自**標準輸入**。輸入中可能包含多組輸入，以題目敘述為主。
- 全部題目的輸出皆輸出到螢幕 **(標準輸出)**。
  輸出和裁判的答案必須完全一致，英文字母大小寫不同或有多餘字元皆視為答題錯誤。
- 比賽中上傳之程式碼，使用 C 語言請用 `.c` 為副檔名；使用 C++ 語言則用 `.cpp` 為副檔名。
- 使用 `cin` 輸入速度遠慢於 `scanf` 輸入，若使用需自行承擔 Time Limit Exceeded 的風險。
- 部分題目有浮點數輸出，會採容許部分誤差的方式進行評測。一般來說「相對或絕對誤差不超過 ϵ 皆視為正確」，ϵ 值以題目敘述為主。

  舉例來說，假設 $\epsilon = 10^{-6}$ 且 a 是正確答案，b 是你的答案，如果符合 $\frac{|a-b|}{\max(|a|,|b|,1)} \leq 10^{-6}$，就會被評測程式視為正確。

Problem	Problem Name	Time Limit	Memory Limit
A	城市分類	2 s	1024 MB
B	旅行銷售員	1 s	1024 MB
C	貓咪排隊買早餐	1 s	1024 MB
D	旅遊	8 s	1024 MB
E	排列	7 s	1024 MB
F	漣漪	1 s	1024 MB
G	反抗軍	4 s	8 MB
H	翻轉隊伍	5 s	1024 MB

2020 網際網路程式設計全國大賽
輸入輸出範例

C 程式範例：

```c
#include <stdio.h>
int main()
{
    int cases;
    scanf("%d", &cases);
    for (int i = 0; i < cases; ++i)
    {
        long long a, b;
        scanf("%lld %lld", &a, &b);
        printf("%lld\n", a + b);
    }
    return 0;
}
```

C++ 程式範例：

```cpp
#include <iostream>
int main()
{
    int cases;
    std::cin >> cases;
    for (int i = 0; i < cases; ++i)
    {
        long long a, b;
        std::cin >> a >> b;
        std::cout << a + b << std::endl;
    }
    return 0;
}
```

A. 城市分類

Problem ID: cities

NPSC 王國有 N 個城市以及 M 條道路，城市編號為 1 到 N ，而一條道路連接兩個相異的城市，使得任兩個城市皆可以透過若干條道路抵達彼此。為了交通的穩定性，NPSC 王國的道路設計還有另一個特點：對於任一條道路（假設它連接城市 x 與 y ），則如果從 x 開始經過這條道路抵達 y ，那麼至多只有一種方法可以再從 y 走回 x 而途中不經過重複的城市與道路（包含 (x, y) 這條道路）。

雖然這種道路的設計大量的減少了人們在同一些城市之間不斷繞來繞去的現象，由於 NPSC 國沒有設置良好的路標，因此還是時常發生有人迷路的事件。身為交通部部長的你，決定將所有城市分成若干類，使得每一條道路都連接兩個不同類的城市。如此一來，人們就能比較方便的確認自己行駛的方向。

當然，如果城市被分成太多類的話也會造成反效果，使得交通系統變得太複雜。因此，你希望將城市在分成最少類的同時，也達到上述的效果。

Input

輸入第一行有兩個非負整數 N 和 M ，分別代表城市以及道路的數量。接著 M 行，第 i 行有兩個正整數 u_i, v_i ，代表第 i 條道路連接編號 u_i 以及編號 v_i 的城市。

- $1 \leq N \leq 5 \times 10^5$
- $1 \leq u_i, v_i \leq N$
- 保證 $u_i \neq v_i$ ，且任兩個城市之間至多只有一條道路

Output

輸出兩行，第一行有一個正整數 K，代表最少所需的城市類別。接著一行輸出 N 個由空白隔開且位於 1 到 K 之間的正整數，第 i 個代表編號為 i 的城市被歸為哪一類。如果有多種符合條件且類別數最少的分類方法，只要輸出任一種分類方法即可。

Sample Input 1

```
4 3
1 2
2 3
3 4
```

Sample Output 1

```
2
1 2 1 2
```

Sample Input 2

```
4 4
1 2
2 3
3 4
4 1
```

Sample Output 2

```
2
1 2 1 2
```

Sample Input 3

```
5 5
1 2
2 3
3 4
4 5
5 1
```

Sample Output 3

```
3
1 2 1 2 3
```

B. 旅行銷售員

Problem ID: salesman

你有聽過有名的**旅行銷售員**問題嗎？問題大概可以這樣描述：

給定一系列城市以及在每對城市之間移動所需要的時間，求解拜訪所有城市恰一次並且回到起始城市所需花費的最短時間。

現在你就是一位銷售員，而你的老闆需要你去每一座城市推銷商品，但你的情境跟**旅行銷售員**有些不同，你的情境是這樣的：

- 你不只是需要花旅行的時間，當你在城市 i 時，你還必須要花 t_i 的時間在這座城市內推銷你的商品，你一開始出發的城市也必須推銷。
- 當你在每一座城市都推銷完商品後，你就會直接在最後一個城市下班，不用回到你出發的城市。
- 剛出發的你活力旺盛，但是你知道長途旅行是會讓人疲累的，因此你給自己一個推銷的策略。你會先選擇一個推銷所需時間非遞減（遞增或持平）的路線，而當你開始疲累的時候，你會轉而走向推銷所需時間非遞增（遞減或持平）的路線。舉例而言，現在有 a, b, c, d, e 五座城市，而這個順序恰好也是你的推銷路線。假設你是在城市 c 開始感到疲倦，那麼 $t_a \leq t_b \leq t_c$ 以及 $t_c \geq t_d \geq t_e$ 必須成立。注意，你可以一出發就感到疲倦，或是到結束時都不感到疲倦。
- 所有城市之間都是互相連通的，當你從城市 i 移動到城市 j 時，你所需要花的旅行時間是 $t_j - t_i$。這是一個時光可以倒流的世界，因此時間可以是負的。

你很想要趕快下班，因此你想要在最短的時間內拜訪每一座城市。你能幫自己規劃一個最好的推銷策略嗎？

Input

輸入共有兩行，第一行是一個正整數 N ，代表城市的數量。方便起見，每座城市都有一個 1 到 N 之間的獨特編號。第二行有 N 個用空白分隔的非負整數，其中第 $i(1 \leq i \leq N)$ 個數字 t_i 代表在第 i 座城市推銷所需要的時間。

你的出發點是第 1 號城市。

- $1 \leq N \leq 10^6$
- $0 \leq t_i \leq 10^9$，對於所有 $1 \leq i \leq N$

Output

請輸出一個整數，代表你拜訪完每一座城市所需要花費的最短時間。

Sample Input 1

```
2
100 200
```

Sample Output 1

```
400
```

Sample Input 2

```
6
4 3 1 6 5 0
```

Sample Output 2

```
15
```

Sample Input 3

```
10
975 187 0 7912 23 456 90 1 975 12
```

Sample Output 3

```
9656
```

C. 貓咪排隊買早餐

Problem ID: cats

最近喵喵來到了貓咪國旅行，這裡充滿各種品種的可愛貓咪，如 bb 貓、加菲貓、電喵、小肥貓……等，只要你能想到的，這邊都能夠看到。

在這邊待了幾天的喵喵，發現貓咪國有一家非常熱門的早餐店 ── NPSC。每天早上，都會有數以萬計的貓咪在店門口排隊，想要吃 NPSC 的早餐。

看到這麼多貓咪在吃 NPSC 的早餐，難道喵喵不會想吃吃看嗎？

喵喵還真的不想吃！比起吃 NPSC 的早餐，喵喵更喜歡觀察那些排在隊伍中的可愛貓咪們！

現在，喵喵發現有 N 隻貓咪正在排隊，從左到右的品種分別為 $a_1, a_2, \ldots, a_N$ 。為了方便，喵喵告訴你的品種都已經換成正整數來表示了，也就是說 $a_1, a_2, \ldots, a_N$ 皆為不超過 10^7 的正整數。

此外，喵喵還發現，如果品種相同的貓咪排在相鄰的位置，那他們便會開始玩耍起來，進而感受不到因漫長排隊所產生的無聊感，這也是喵喵最喜歡看到的現象。

充滿好奇心的喵喵，看著這 N 隻正在排隊的貓咪，不禁心想，如果他能把至多 M 隻貓咪的品種改變成任何品種，喵喵可以看到「最長連續相同品種的貓咪數量」最大可以是多少呢？

也許你會問，喵喵要如何改變貓咪的品種呢？這個問題簡單！喵喵可以用染毛劑來幫貓咪換顏色，貓咪們便可以假裝改變品種囉！由於喵喵手邊只有 M 個染毛劑，因此至多只能幫 M 隻貓咪改變品種。

其次，你可能會好奇，「連續相同品種的貓咪數量」的意思是什麼？舉個例子，若存在一個區間 $[l, r]$ ，滿足 $a_l = a_{l+1} = \ldots = a_r$ ，那麼 $r - l + 1$ 便是「連續相同品種的貓咪數量」。

Input

輸入第一行有兩個非負整數 N 和 M ，分別代表隊伍中貓咪的數量以及喵喵可以至多改變 M 隻貓咪的品種。

輸入第二行有 N 個正整數 $a_1, a_2, \ldots, a_N$，分別代表從左至右每隻貓咪的品種。

- $1 \leq N \leq 10^6$
- $0 \leq M \leq N$
- $1 \leq a_i \leq 10^6$

Output

輸出一個整數於一行，代表在喵喵可以改變至多 M 隻貓咪品種的情況下，喵喵可以看到「最長連續相同品種的貓咪數量」最大可以是多少。

Sample Input 1	Sample Output 1
3 1 1 2 1	3

Sample Input 2	Sample Output 2
5 1 1 2 1 2 1	3

D. 旅遊

Problem ID: travel

小 N 是一個喜好旅遊的人。他最近發現 NPSC 島十分適合旅遊，因此他決定規劃一個在 NPSC 島旅遊的行程。

經過詳細的調查之後，發現 NPSC 島總共有 N 個主要的城鎮，由 0 編號到 $N-1$，並且每個城鎮都有至少一條通往其它城鎮的單向道路，有些城鎮甚至會有超過一條通往同一個城鎮的道路。為了得到最佳的旅遊體驗，小 N 把從每個城鎮出發的所有道路都排好順序。小 N 計畫了 T 天的行程，每天遊覽一個城鎮，並且要按照自己排的道路順序選擇隔天的要前往的城鎮。

具體來說，小 N 在行程開始前會先用八個幸運數字 $A, B, C, D, E, F, a_0, a_1$ 產生一個數列 $a_0, a_1, \cdots, a_{T-1}$，其中對於所有 $i \geq 2$，$a_i = Aa_{i-1}^2 + Ba_{i-1}a_{i-2} + Ca_{i-2}^2 + Da_{i-1} + Ea_{i-2} + F$。接著，假設小 N 第 i 天在編號 x 的城鎮，且在小 N 的排序之後由該城鎮出發的各個道路分別通往編號 $p_{x,0}, p_{x,1}, \cdots, p_{x,m_x-1}$ 的城鎮，那麼他隔天就會前往編號 $p_{x,q}$ 的城鎮，其中 q 是 a_i 除以 m_x 的餘數。

然而因為小 N 的行程實在是太長了，而且他的數列中數字也都非常大，所以請你幫小 N 寫一個程式，方便他用自己排的道路順序和第 1 天所在的城鎮計算出這 T 天分別會在哪個城鎮。

Input

第一行有三個以空白隔開的非負整數 N, S, T，代表島上的城鎮數量、小 N 第 1 天所在的城鎮編號與他總共要移動的次數。

第二行有八個以空白隔開的非負整數 $A, B, C, D, E, F, a_0, a_1$，意義如題目所述。

接下來有 N 行，每行有數個以空白隔開的非負整數，第 i 行（從 0 開始編號）代表從第 i 個城鎮出發的所有道路。第一個數字 m_i 代表從該城鎮出發的道路總數，接下來 m_i 個數字依序代表每條道路連向哪一個城鎮，道路的順序即是按照小 N 規劃時決定的順序。

- $0 \leq S < N \leq 3 \times 10^4$
- $T \leq 5 \times 10^6$
- $1 \leq m_i$
- $m_0 + m_1 + \cdots + m_{N-1} \leq 10^5$

- $0 \leq a_0, a_1, A, B, C, D, E, F \leq 10^5$

Output

請輸出 T 行，依序代表小 N 每天所在的城鎮編號（因此第一行一定是 S）。

Sample Input 1

```
4 0 9
1 0 0 0 0 0 1 2
3 1 3 2
1 0
2 0 0
1 0
```

Sample Output 1

```
0
2
0
3
0
3
0
3
0
```

Sample Input 2

```
4 1 8
1 2 3 4 5 6 7 8
2 2 1
2 0 3
2 1 0
2 2 1
```

Sample Output 2

```
1
0
2
1
0
2
1
0
```

E. 排列

Problem ID: permute

$p_0, p_1, \cdots, p_{M-1}$ 是 M 個 $1 \sim N$ 的排列，且對於 $i = 1, 2, 3, \cdots, M-1$ ，p_i 是 p_{i-1} 交換第 x_i 和第 y_i 個數字得到的排列。假設依照字典序排序這 M 個排列後我們有 $p_{a_1} \leq p_{a_2} \leq \cdots \leq p_{a_M}$（若有一樣的排列，則讓下標較小的排列出現在前面），請輸出 $a_1, a_2, \cdots, a_M$。

Input

輸入第一行有兩個正整數分別代表 N 和 M。下一行有 N 個數字代表 p_0。接著 $M-1$ 行中每行有兩個正整數，其中第 i 行的兩個正整數代表 x_i, y_i。

- $2 \leq N \leq 10^5$
- $1 \leq M \leq 10^5$
- p_0 是一個 $1 \sim N$ 的排列
- $1 \leq x_i, y_i \leq N$ 且 $x_i \neq y_i$

Output

輸出只有一行包含 M 個以空白分隔的整數 $a_1, a_2, \cdots, a_M$。

Notes

範例測試一中，$p_0 = [3, 1, 2], x = [1, 1, 2], y = [2, 3, 3]$，將 p_0 的第一個數字（$x_1 = 1$）和第二個數字（$y_1 = 2$）交換，得到 $p_1 = [1, 3, 2]$，接著把 p_1 的第一個數字和第三個數字交換，得到 $p_2 = [2, 3, 1]$，最後類似的得到 $p_3 = [2, 1, 3]$。經過字典序比較後我們知道 $p_1 < p_3 < p_2 < p_0$，所以照順序輸出 $1, 3, 2, 0$ 四個數字。

範例測試二中，$p_0 = [1, 2], p_1 = [2, 1], p_2 = [1, 2]$，注意到 p_0 和 p_2 是一樣的排列，所以我們將 p_0 排到 p_2 前面，也就是 $p_0 \leq p_2 \leq p_1$，所以輸出 $0, 2, 1$ 三個數字。

Sample Input 1	Sample Output 1
3 4 3 1 2 1 2 1 3 2 3	1 3 2 0

Sample Input 2	Sample Output 2
2 3 1 2 1 2 1 2	0 2 1

F. 漣漪

Problem ID: circles

某個下雨天，你盯著湖面，發現水面上因為下雨而有了一圈圈的漣漪。不知道為什麼的，你突然想知道這片水面上共有幾個不同的產生了建設性干涉的位置。

講的稍微白話一點，假設整個水面是個無限寬廣的二維平面，而有個雨滴在時間 0 落在 (x, y) 的位置，則在時間 t 的時候，這個雨滴所產生的圓就是以 (x, y) 為圓心，半徑為 t 的圓。而產生了建設性干涉的位置就是多於一個圓經過的點。

現在，給定了每個雨滴落在水面上的位置 (x_i, y_i)，並假定每個雨滴都是在時間 0 碰到水面。在這無限可能的所有時間點當中，總會有個時間點 T 使得這個時間點的相異的產生建設性干涉的位置的數量是最多的，你想要知道這個最多的數量會是多少。

然而，因為誰也沒辦法預測雨滴的落點會在哪裡，所以我們會利用以下的 C 程式碼來產生每個雨滴的 x_i, y_i：

```
unsigned s = initSeed;
int left[30] = { 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
    1, 0, 1, 1, 1, 0, 1, 0, 1, 1 };
int right[30] = { 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1,
   1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1 };
int gen() {
  unsigned t = (s & 13) ^ (s << 13) ^ (s >> 16) ^ (s << 14) ^ (s >> 17) ^ (s
      << 3) ^ (s >> 6);
  unsigned r = (t & 880301) ^ (t >> 11) ^ (t << 11) ^ (t >> 12) ^ (t << 12);
  unsigned u = (t & 110029) ^ (t << 21) ^ (t << 14) ^ (t >> 11) ^ (t >> 9);
  unsigned h = (s ^ t ^ r ^ u);
  for (int i = 0; i < 30; i++) if (left[i]) h ^= (h << i);
  for (int i = 0; i < 30; i++) if (right[i]) h ^= (h >> i);
  return s = h % 1000000001;
}

int x[50], y[50];
void generateTestcase(int N) {
  for (int i = 0; i < N; i++) {
    x[i] = gen() ^ i;
    y[i] = gen() ^ gen() ^ i;
  }
}
```

陣列 x 跟 y 中分別存的就是每個點的 x_i 跟 y_i。

Input

輸入共有一行，包含兩個整數，分別是 N 及 $initSeed$，代表雨滴的數量以及給以上程式用的初始種子。

- $1 \leq N \leq 50$
- $0 \leq initSeed \leq 10^9$
- 保證輸入中不會有超過一個雨滴落在同一個位置

Output

輸出共有一行，包含一個整數，為最大的產生建設性干涉的位置的數量。

Sample Input 1	Sample Output 1
1 1	0

Sample Input 2	Sample Output 2
2 998244353	2

G. 反抗軍

Problem ID: resistance

注意此題的記憶體限制非常小，且爲強制在線。此外，由於標頭檔的限制，只能使用 C++ 作答。

> 西元 20xx 年，有個知名的競賽程式 vtuber 團體，致力於提升自己在各大 online judge 的 Elo rating，史稱 EloLive。
> 其中，又有擅長構造題的一幫人，被稱做「兔田建設」（Usada Construction）。

還記得 EloLive 嗎？除了兔田建設以外，EloLive 內還有一群以湊あくあ（Minato Aqua）爲首的人，被稱作 EloLive 反抗軍。這些人有著偏激的思想，認爲用 C++ 寫競賽程式是件毒瘤且糟糕的事情，應當用 Haskell 或 Scala 等 functional programming 語言取而代之。也因此，他們計劃著用 TNT 炸毀現有的 EloLive，創立新秩序，成爲新世界的神。

不過，如同任何反叛組織，EloLive 反抗軍也是有被倒戈背刺的一天。這天，夏色まつり（Natsuiro Matsuri）離開了反抗軍的懷抱，投奔了 EloLive 大前輩時乃空（Tokino Sora）的陣營。

EloLive 成員們一直都想破獲反抗軍使用的祕密通訊方式。根據夏色まつり的情報，他們得到以下的資訊：

- 訊息被一個區塊大小爲 24-bit 的 block cipher（區塊加密法）使用 ECB mode（電子密碼本模式）（見下頁註）加密
- 加密時所用的金鑰

可惜的是，他們並不知道加密確切使用了什麼演算法。不過，夏色まつり從基地中偷出了一臺機器，能夠使用給定的金鑰加密，並給出相對應的密文。

現在，EloLive 成員們得到了一些密文，想要使用手上的資訊進行解密。但是，因爲夏色まつり把精力都拿來呼吸其他 EloLive 成員了，記憶力並不是特別好，所以在解密全程**只能使用 8 MB 的記憶體**。除此之外，由於她想要盡早回報~~やばい~~不好的訊息，因此她想要在得到一個密文之後，立即得到答案（也就是說此題爲**強制在線**）。

身爲祭絲的你，寫個程式幫助她吧！

註

對於不熟悉密碼學的同學，~~可以選擇不做這一題，一道美味的燒雞就完成了~~以下做一些名詞解釋。

Block cipher（區塊加密法）是一類加密的方式，一次對一個固定大小的區塊，根據金鑰，進行某種確定性且可逆的操作。(在這個情境中，一個區塊的大小便是 24-bit。)

而 ECB mode（電子密碼本模式) 指的是直接一塊一塊地將明文輸入上述 block cipher，並將結果也直接輸出，而不進行更多處理。顯然地，這樣代表對於每塊相同的明文，ECB mode 都會輸出相同的密文。

另外值得注意的是，因爲我們需要能夠解密，加密的算法需要是**可逆**的。也就是說，對任意的密文都存在唯一相對應的明文。(換言之，本題的正確輸出是唯一的。)

總而言之，如果我們固定金鑰，那這個算法是一個可逆函數，把一個 24-bit block 映射到另一個 24-bit block。

機器使用

請使用以下的程式碼來進行加密：

```cpp
#if __has_include("resistance.h")
#include "resistance.h"
#else
#include <cstdint>
class Machine {
 public:
  explicit Machine(uint64_t k) : key(k & ((1 << 24) - 1)) {}
  int Encrypt(int x) { return (x ^ key); }
 private:
  uint64_t key;
};
#endif
```

並以以下的方式呼叫：

```cpp
Machine machine(key);       // 初始化金鑰
machine.Encrypt(message);   // 加密 `message`，回傳 24-bit 密文
```

注意到提交到評測系統上時會自動改用 `resistance.h` 這個標頭檔，不需要進行額外操作。這同時也代表在評測系統上的加密算法並非如此簡單。

另外，你可以假設 `Encrypt` 這個函數是非常快的，一秒能進行約略 4×10^7 個操作（當然，這不算你的程式其它部分的執行時間）。

Input

注意此題爲強制在線！

輸入的第一行包含一個 64-bit 無號整數 K，代表加密時使用的金鑰。

第二行包含一個正整數 Q，表示需要解密的密文個數。

接下來有 Q 行，每行是一個 24-bit 整數 x_i（$1 \leq i \leq Q$）。需要注意的是，你需要透過上一個詢問的答案來取得這個詢問的密文。確切地說，假設第 i 個詢問的答案是 a_i，那麼第 $i+1$ 個詢問的密文便是 $x_{i+1} \oplus a_i$，其中 $\oplus$ 是異或（xor）運算。特別地，第 1 個詢問的密文即是 x_1。

- $0 \leq K < 2^{64}$
- $1 \leq Q \leq 5 \times 10^5$
- $0 \leq x_i < 2^{24}$

Output

對於每筆詢問，請輸出一行，該行包含一個整數表示該詢問對應的明文。

Sample Input 1	Sample Output 1
20180601000722 5 962820 1584920 14730246 10227930 15691788	8610326 1482268 8096008 6965952 584158

This page is intentionally left blank.

H. 翻轉隊伍

Problem ID: reversing

NPSC 高中正在校外教學，其中你恰好是一個有 N 位學生的隊伍領隊，學生們的座號分別是 $1 \sim N$，行動時他們總是照著座號排成一列。

校外教學的活動非常多，每次活動每個隊伍都必須挑選出一些學生形成小組排成一列進行，為了省時間，你決定每次都直接挑選出一段座號連續的學生來成為該次活動的小組。

麻煩的是，這些學生非常喜歡聊天，如果挑選出去的小組太吵而被扣秩序成績那就不好了！

經過調查後，你發現這座號 i 的學生有興趣的話題為 a_i，有趣的是，這些學生只會跟和自己**相鄰且興趣相同**的人聊天，這就代表你只需要在挑選出小組後，盡量讓相鄰且興趣不同的對數最多，整個小組就會最安靜！

不過調整隊伍是一個很麻煩的任務，基於一些問題，在挑選出小組後，你只能不斷再從中選出一段連續的學生，並將他們頭尾翻轉來組合出新的排列，而且由於時間緊迫，你希望**操作次數越少越好**。

舉例來說，假設現在一支 5 個人的小組有興趣的話題編號依序是 $1\ 1\ 1\ 2\ 2$，你可以把位置區間 $[2, 4]$ 的學生翻轉使得有興趣的話題編號變成依序是 $1\ 2\ 1\ 1\ 2$，然後再翻轉 $[4, 5]$ 的學生變成 $1\ 2\ 1\ 2\ 1$，如此一來這個小組就有 4 對相鄰且興趣不同的人，也是最多可能的對數，而 2 次操作也是最少的操作次數。

不巧的是，問題可沒這麼單純。由於高中生們有興趣的話題總是在改變，因為平常行動時學生們總是照著座號排成一列，因此常常會有一段座號連續學生的興趣一口氣發生改變！

請你撰寫一支程式，並支援 Q 次事件，第 i 次事件可能是你挑選了區間 $[l_i, r_i]$ 的學生出來形成小組，而你想同時知道「相鄰且興趣不同的對數最多可能是多少」和「達到這最佳對數最少需要幾次翻轉操作」；也有可能是區間 $[l_i, r_i]$ 的學生的興趣全部變成了 v_i。

注意到就算你挑選了一些學生形成小組並將他們做了重新編排，他們依然會在活動結束後回到原位按照座號排好，但興趣的改變將會是持續的。

Input

輸入的第一行有兩個正整數 N, Q，代表隊伍的人數、事件的個數。

第二行有 N 個整數 $a_1 \sim a_N$，代表第 i 個人有興趣的話題為 a_i。

接下來 Q 行，第 i 行首先會有一個整數 t_i，代表事件的種類：

- 若 $t_i = 1$，接下來會有兩個整數 l_i, r_i，代表你挑選了區間 $[l_i, r_i]$ 的學生出來形成小組。
- 若 $t_i = 2$，接下來會有三個整數 l_i, r_i, v_i，代表區間 $[l_i, r_i]$ 的學生興趣全部變成了 v_i。

所有同一行的數字將會以一個單一空格隔開。

- $1 \leq N, Q \leq 2.5 \times 10^5$
- $0 \leq a_i, v_i \leq N$
- $1 \leq l_i \leq r_i \leq N$
- 保證存在一個 $1 \leq i \leq Q$ 滿足 $t_i = 1$

Output

對於所有 $t_i = 1$ 的事件，依序輸出一行兩個整數，分別代表該小組「相鄰且興趣不同的對數最多可能是多少」和「達到這最佳對數最少需要幾次翻轉操作」。

Sample Input 1	Sample Output 1
5 3 1 1 1 2 2 1 1 5 2 2 5 3 1 1 5	4 2 2 1

Sample Input 2	Sample Output 2
7 7 2 2 3 3 1 1 4 1 1 7 2 5 7 4 1 3 7 2 3 4 2 1 2 6 2 5 7 2 1 1 7	6 2 4 2 4 2 0 0

This page is intentionally left blank.