

114 學年度資訊學科能力競賽臺南一中校內複選 試題本

競賽規則

1. 競賽時間：2025/9/25 13:00 ~ 17:00，共 4 小時。
2. 本次競賽試題共 6 題，每題皆有子任務。
3. 為了愛護地球，本次競賽題本僅提供電子檔，不提供紙本。
4. 每題的分數為該題所有子任務得分數加總；單筆子任務得分數為各筆繳交在該筆得到的最大分數。
5. 本次初選提供記分板，複選不提供記分板。
6. 全部題目的輸入皆為標準輸入。
7. 全部題目的輸出皆為標準輸出。
8. 所有輸入輸出請嚴格遵守題目要求，多或少的換行及空格皆有可能造成裁判系統判斷為答案錯誤。
9. 每題每次上傳間隔為 120 秒，裁判得視情況調整。
10. 所有試題相關問題請於競賽系統中提問，題目相關公告也會公告於競賽系統，請密切注意。
11. 如有電腦問題，請舉手向監考人員反映。
12. 如有如廁需求，須經過監考人員同意方可離場。
13. 不得攜帶任何參考資料，但競賽系統上的參考資料可自行閱讀。
14. 不得自行攜帶隨身碟，如需備份資料，請將資料儲存於電腦 D 槽。
15. 競賽中請勿交談。請勿做出任何會干擾競賽的行為。
16. 如需使用 C++ 的 `std::cin` 或 `std::cout` 可將以下程式碼插入 `main function` 以及將 `endl` 取代為 `'\n'` 來優化輸入輸出速度。唯須注意不可與 `cstdio` 混用。

```
std::ios::sync_with_stdio(false);  
std::cin.tie(nullptr);
```

A. 部落衝突

Problem ID: tribe

Time Limit: 1.0s

Memory Limit: 512MiB

— 題目描述 —

細胞國內有 N 棟房子，第 i 棟房子隸屬於編號 C_i 的部落，這些部落因為沒有手機而時常發生部落衝突。

野豬騎士身為細胞國的首領，為了減少部落衝突發生，打算在這 N 棟房子間鋪設 $N - 1$ 條道路，使得**任兩棟房子間都能通過一或多條道路互相抵達**，如此一來，可以增進各個部落間的友誼，使得細胞國變為超級細胞國。

為了避免鋪設完道路後遇到不受控的野蠻人抗議，野豬騎士想要讓這 $N - 1$ 條道路中，恰好 K 條道路（不能多也不能少）兩端的房子隸屬於**不同**部落。身為野豬騎士，他還忙著用橡皮筋狩獵山豬，因此將設計道路的重責大任交給了你，請你告訴野豬騎士，是否存在一種方法可以滿足條件，如果有，你還必須告訴他要怎麼鋪設。

我們說兩棟房子 a, b 能經過道路互相抵達，代表存在一系列相異的房子 p_0, p_1, \dots, p_t 滿足 $p_0 = a, p_t = b$ ，且所有 1 到 t 之間的整數 i ，第 p_{i-1} 棟房子和第 p_i 棟房子之間皆有鋪設道路。換句話說，由房子作為點而道路作為邊的圖上，兩個點是連通的。

— 輸入 —

第一行有兩個整數 N, K

第二行有 N 個以空白分開的正整數，第 i 個為 C_i 。

— 輸出 —

如果有滿足條件的鋪設道路方式，第一行輸出 “Yes”，接著輸出 $N - 1$ 行，第 i 行兩個數字 U_i, V_i ，代表要將第 U_i 棟房子和第 V_i 棟房子之間鋪設一條道路，如果有多種鋪設道路的方式都可以滿足條件，回答其中一種即可。

如果無法滿足條件，輸出 “No”。

— 輸入限制 —

- $2 \leq N \leq 10^5$
- $1 \leq C_i \leq N$
- $0 \leq K \leq N - 1$

— 子任務 —

編號	分數	額外限制
1	0	範例輸入輸出
2	3	$K = 0$
3	13	$K = N - 1$
4	47	$C_i \in \{1, 2\}$
5	37	無額外限制

— 範例輸入 1 —

5 3
1 1 5 3 3

— 範例輸出 1 —

Yes
2 5
5 3
1 5
4 5

— 範例輸入 2 —

5 2
1 1 1 1 1

— 範例輸出 2 —

No

B. 分蛋糕 2

Problem ID: cake2

Time Limit: 1.0s

Memory Limit: 512MiB

— 題目描述 —

由於上次製作的蛋糕受到朋友們的一致好評，巴漆又製作了一條長度為 N 的蛋糕，由左到右編號為第 1 段到第 N 段，巴漆利用神秘的蛋糕透視儀器得到了每一段蛋糕的真實好吃程度，第 i 段的好吃程度為 C_i 。

巴漆為了讓朋友們品嚐他親手做的蛋糕，打算將蛋糕切成一些區間後送給他們。蛋糕有限，所以每段蛋糕最多只能送給一位朋友，並且由於巴漆吃膩蛋糕了，他希望**每一段蛋糕都能送給其中一個朋友**。

俗話說「朋友多蛋糕少」，所以巴漆一定找的到朋友送給他蛋糕。而每個朋友可以收到一個**連續區間**的蛋糕。

對於每個有收到蛋糕的朋友，都會對巴漆有好感值，這次計算方式為：如果那個朋友收到小於 K 段蛋糕，則好感值為 0，否則，那個朋友對巴漆的好感值為他收到所有蛋糕的好吃程度的**最小值**

巴漆想要找到一種分蛋糕的方式，使得所有朋友對他的好感值總和最大，而巴漆不會寫程式，因此請你來幫助他算出答案。

再更精確地來說，我們定義一個合法的切蛋糕方式為一系列的區間 $(l_1, r_1), (l_2, r_2), \dots, (l_k, r_k)$ 滿足 $l_1 = 1, r_k = n$ ，並且對於所有 $1 \leq i \leq k$ 都有 $l_i \leq r_i$ ，且對於所有 $1 \leq i < k$ ，都有 $r_i + 1 = l_{i+1}$ 。而這個切蛋糕方式所得到的好感值總和為

$$\sum_{t=1}^k \begin{cases} 0 & \text{if } r_t - l_t + 1 < K \\ \min_{l_t \leq i \leq r_t} C_i & \text{otherwise} \end{cases}$$

我們要找出所有合法的切蛋糕方式中好感值總和最大的是多少。

— 輸入 —

第一行有兩個整數 N, K 。

第二行有 N 個以空白分開的整數，第 i 個為 C_i 。

— 輸出 —

輸出所有朋友對巴漆的好感值總和最大可能是多少。

— 輸入限制 —

- $1 \leq K \leq N \leq 2 \times 10^5$
- $-10^9 \leq C_i \leq 10^9$

— 子任務 —

編號	分數	額外限制
1	0	範例輸入輸出
2	4	$N \leq 3000$
3	36	$K > 1$
4	21	$ C_i \leq 1$
5	39	無額外限制

— 範例輸入 1 —

10 3
2 2 3 3 3 2 -1 9 9 -2

— 範例輸出 1 —

4

— 範例解釋 1 —

可以將蛋糕切成 $[2, 2, 3]$, $[3, 3, 2]$, $[-1]$, $[9, 9]$, $[-2]$, 其中 $[2, 2, 3]$ 產生的好感值為 2, $[3, 3, 2]$ 產生的好感值為 2, $[-1]$, $[9, 9]$, $[-2]$ 產生的好感值皆為 0, 好感值總和為 4, 可以證明沒有總和大於 4 的分法。

— 範例輸入 2 —

6 1
-3 2 -3 5 7 -2

— 範例輸出 2 —

7

— 範例解釋 2 —

可以將蛋糕切成 $[-3, 2, -3]$, $[5]$, $[7]$, $[-2]$, 其中 $[-3, 2, -3]$ 產生的好感值為 -3, $[5]$ 產生的好感值為 5, $[7]$ 產生的好感值為 7, $[-2]$ 產生的好感值為 -2, 好感值總和為 7。

— 範例輸入 3 —

6 1
-7 5 -4 5 -8 6

— 範例輸出 3 —

-2

C. 手錶

Problem ID: clock

Time Limit: 1.0s

Memory Limit: 512MiB

— 題目描述 —

矮歐哀是著名的手錶收藏家，他有 N 個**不會動**的手錶，每個手錶有 M 個刻度，第 i 個手錶的長針指向刻度 A_i ，短針指向刻度 B_i 。

矮歐哀發現有些手錶長針指向的刻度竟然比短針還小，他覺得這不是好兆頭，於是他打算將手錶拿去廠商調整，他希望調整後所有長針指向的刻度都大於等於短針指向的刻度，也就是說，調整後需滿足所有的 $A_i \geq B_i$ 。

而每次調整可以將一些手錶的長針同時順時針轉相同的距離，也就是說，每次調整可以選擇一個整數 k 和一些手錶，並將這些手錶的長針從 A_i 變為 $(A_i + k) \bmod M$ ，而短針對矮歐哀有獨特的意義，因此短針是不能被調整的。

然而，每次調整手錶需要花費一些時間，矮歐哀已經等不及了，因此**每支手錶最多只能被送去調整一次**。並且調整手錶需要高額的費用，矮歐哀不想花太多錢，所以希望**花越少次調整手錶越好**，如果不是花最少次調整，你可能只能拿到一點分數，詳情請看下方子任務部分。你，身為矮歐哀的財務管理員，幫矮歐哀分配一下要怎麼將手錶送去給廠商調整才能滿足條件。

— 輸入 —

第一行有兩個整數 N M 。

接著有 N 行，第 i 行有兩個整數 A_i, B_i ，分別代表矮歐哀第 i 支手錶長針和短針指向的刻度。

— 輸出 —

第一行輸出一個整數 T ，代表打算花 T 次調整手錶。

接著對於第 i 次調整，輸出兩個整數 t_i k_i ，接著下一行輸出 t_i 個整數，第 j 個整數為 $v_{i,j}$ 。代表這次要將這 t_i 個手錶的長針從 $A_{v_{i,j}}$ 變為 $(A_{v_{i,j}} + k_i) \bmod M$ 。

輸出需要滿足

- $1 \leq t_i \leq N$
- $0 \leq k_i \leq M - 1$
- $1 \leq v_{i,j} \leq N$
- $v_{i,j}$ 兩兩相異

若有多種方式皆可滿足條件，輸出任一種皆可。

— 輸入限制 —

- $1 \leq N \leq 2 \times 10^5$
- $1 \leq M \leq 10^9$
- $0 \leq A_i, B_i < M$

— 子任務 —

編號	分數	額外限制
1	0	範例輸入輸出
2	5	A_i 皆相同
3	26	$N \leq 2000$
4	45	B_i 皆相同
5	24	無額外限制

在每個子任務中，如果輸出符合上述輸出格式，且調整後滿足所有 $A_i \geq B_i$ ，但輸出的 T 並非最小，你可以獲得那個子任務的總分乘以 0.3 的分數。

— 範例輸入 1 —

```
5 10
1 4
5 8
0 9
2 1
9 0
```

— 範例輸出 1 —

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

— 範例解釋 1 —

經過調整後，第 1 到 5 個手錶的 (A_i, B_i) 分別變為 $(5, 4)$, $(9, 8)$, $(9, 9)$, $(2, 1)$, $(9, 0)$ 。

— 範例輸入 2 —

```
5 5
0 4
1 4
2 4
3 4
4 4
```

— 範例輸出 2 —

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

D. 隱藏的排列 2

Problem ID: permutation2

Time Limit: 1.0s

Memory Limit: 512MiB

本題為**互動題**

— 問題描述 —

Alice 和 Bob 正在遊玩猜謎遊戲，由 Alice 負責出題目、Bob 負責猜謎。

遊戲過程如下：

- Alice 會先在心中想好一個 $1 \sim n$ 的排列。也就是說，Alice 心中已經有一個隱藏的序列 p_1, p_2, \dots, p_n ，滿足這些數字都介在 $1 \sim n$ 之間，且每個數字出現恰好一次。
- 接著，Bob 可以詢問 q 個問題，每個問題都會是以「請問有多少數對 (i, j) 滿足 $l \leq i < j \leq r$ ，且 $p_i > p_j$ 」這種形式呈現。Alice 在收到問題後，必須如實回答。
- 在 Bob 問完所有問題後，Alice 會問 Bob k 個問題，每個問題都會是以「請問 p_i 是多少？」這種形式呈現。Bob 在收到問題後，必須給出回答。

這個遊戲的目的就是要讓 Bob 詢問的問題數量 q 儘量小來使得 Bob 能正確回答出 Alice 的所有詢問。請協助 Bob，在 q 儘量小的情況下，正確回答所有 Alice 的 k 個問題。

— 實作細節 —

你需要實作兩個函式 `bob_init()` 與 `query_from_alice()`：

```
void bob_init(int n);
```

- 對於每一筆測試資料，正式評分程式會呼叫你實作的 `bob_init()` 函式恰好 1 次。
- n 代表 Alice 心中想著的排列的長度

```
int query_from_alice(int a);
```

- a 為 1 到 n 之間的整數
- 對於每一筆測試資料，正式評分程式會呼叫你實作的 `query_from_alice()` 函式恰好 k 次。
- 保證在呼叫完 `bob_init()` 後才會呼叫此函式。
- `query_from_alice()` 需要回傳一個整數 x ，代表 p_a 的實際數值。

此外，在實作 `bob_init` 時可以呼叫 `compare_numbers()` 這個函式。

```
int compare_numbers(int l, int r);
```

- l, r 是於 $1 \sim n$ 的整數
- $l \leq r$
- 此函式會回傳有多少數對 (i, j) 滿足 $l \leq i < j \leq r$ ，且 $p_i > p_j$
- **範例評分程式**內的 `compare_numbers()` 實作與**實際評分程式**內的實作完全相同

— 範例程式碼 —

以下是一個可以編譯但保證不會獲得任何分數的範例程式碼：

```
#include<vector>
using namespace std;
int compare_numbers(int l, int r);
int v[1010];
void bob_init(int n){
    v[1] = compare_numbers(1, n);
    v[2] = compare_numbers(2, n);
}
int query_from_alice(int a){
    return v[a];
}
```

— 互動範例 —

若 Alice 所想的序列為 $\{2, 3, 1\}$ ，一個可能被評為 Accepted 的互動例子顯示如下：

評分程式端	參賽者端
呼叫 bob_init(3)	
	呼叫 compare_numbers(1, 3)
回傳 2	
	呼叫 compare_numbers(2, 3)
回傳 1	
	回傳 void()
呼叫 query_from_alice(1)	
	回傳 2
呼叫 query_from_alice(2)	
	回傳 3

— 測資限制 —

- $3 \leq n \leq 1000$
- $1 \leq k \leq 1000$
- $1 \leq p_i \leq n$
- p_i 兩兩相異

— 評分說明 —

對於每一筆測試資料，若你的程式在函式 `bob_init()` 中呼叫 `compare_numbers` 的次數為 x ，則定義 Q 為：

$$Q = \left\lfloor \frac{x}{n} \right\rfloor$$

若你正確回答了所有 Alice 的詢問，根據 Q ，你將得到分數比重 W ：

$$W = \begin{cases} 1 & \text{if } Q = 0 \\ 1 - \frac{\sqrt{Q}}{50} & \text{if } 0 \leq Q \leq 500 \\ 0 & \text{if } Q > 500 \end{cases}$$

本題共有兩組子任務，條件限制如下所示。每一組可有一或多筆測試資料，你在該子任務的得分為所有測試資料中分數比重 W 的最小值，乘以該子任務的總分。

— 子任務 —

編號	分數	額外限制
1	0	範例互動
2	10	$n = 3$
3	90	無額外限制

— 範例評分程式 —

範例評分程式採用以下格式輸入：

$$\begin{array}{c} n \ k \\ p_1 \ p_2 \ \dots \ p_n \\ a_1 \ a_2 \ \dots \ a_k \end{array}$$

請注意，正式的評分程式一定不會採用以上格式輸入。**請不要自行處理輸入輸出。**

範例評分程式首先呼叫 `bob_init(n)`，接著範例評分程式會呼叫 k 次 `query_from_alice(a_i)`。接著，若範例評分程式偵測到從 `bob_init` 對 `compare_numbers` 的呼叫有任何不合法、或在 `query_from_alice` 的期間有對 `compare_numbers` 的呼叫，此程式將輸出

Wrong Answer : msg

後並終止程式執行，其中 `msg` 為下列其中之一錯誤訊息：

- Invalid position: $l \ r$: 你的程式傳入 `compare_numbers` 的集合中有不介在 $1 \sim n$ 之間的數字，或是你傳入的 $l > r$ 。
- Invalid call: 你的程式嘗試在 `query_from_alice` 的期間呼叫 `compare_numbers`。

否則，範例評分程式將會以下列格式印在標準輸出中：

$$\begin{array}{c} b_1 \ b_2 \ \dots \ b_k \\ Q \end{array}$$

其中，

- b_i 為第 i 次呼叫 `query_from_alice()` 時你的回傳值。
- Q 為根據你的程式呼叫 `compare_numbers` 的次數得來的數值，詳細定義請見評分說明欄位。
- 請注意，範例評分程式並不會幫助你檢查你回傳的數值是否正確。

下方程式碼可在 pA 下方壓縮檔 permutation2.zip 裡的 grader.cpp 中獲得
請注意，上傳程式碼時請勿直接上傳此範例評分程式，上傳格式請參考上述範例程式碼。

```
#include <cstdlib>
#include <iostream>
using namespace std;
// Functions to be implemented in the solution.
void bob_init(int n);
int query_from_alice(int a);
// Functions to be implemented in the solution.
namespace{
    int N, K, Query_count = 0, P[1005], Ans[1005], Inv[1005][1005];
    bool EndInit = false;
    void WA(const string msg) {
        cout << "Wrong Answer: " << msg << endl;
        exit(0);
    }
}
int compare_numbers(int l, int r){
    if(EndInit) WA("Invalid call");
    if(l <= 0 || l > N || r <= 0 || r > N || l > r)
        WA("Invalid position: " + to_string(l) + " " + to_string(r));
    Query_count++;
    return Inv[l][r];
}
int main() {
    cin >> N >> K;
    for(int i = 1; i <= N; ++i) cin >> P[i];
    for(int i = 1; i <= N; ++i) for(int j = i + 1; j <= N; ++j)
        if(P[i] > P[j]) Inv[i][j]++;
    for(int i = 1; i <= N; ++i) for(int j = 1; j <= N; ++j)
        Inv[i][j] += Inv[i][j - 1];
    for(int j = 1; j <= N; ++j) for(int i = j - 1; i > 0; --i)
        Inv[i][j] += Inv[i + 1][j];
    bob_init(N);
    EndInit = true;
    for(int i = 1, x; i <= K; ++i)
        cin >> x, Ans[i] = query_from_alice(x);
    for(int i = 1; i <= K; ++i) cout << Ans[i] << " \n"[i == K];
    cout << Query_count / N << "\n";
}
```

E. 數城堡

Problem ID: rook

Time Limit: 1.0s

Memory Limit: 512MiB

— 題目描述 —

Hikaru 在下西洋棋時在為了嘲諷中國對手於是根據「剩餘的兵走到底線可以升變的定理」一次變出了 5 隻城堡，眾所皆知城堡只能移動直的或橫的，於是 Hikaru 非常想知道以下問題的答案。

Hikaru 有一個 $N \times N$ 的棋盤和 K 種顏色的的城堡，第 i 種顏色的城堡有 C_i 個。Hikaru 想要知道有多少種方法能把這些城堡**全部**放在這個棋盤上，使得任兩個城堡都不會互相攻擊到。也就是說，如果位置 (x, y) 放了一個城堡，必須滿足對於所有 $1 \leq i \leq N, i \neq y, 1 \leq j \leq N, j \neq x, (x, i), (j, y)$ 都沒有城堡，並且每個格子上最多只能放一個城堡。因為數字可能很大，所以請回答 $\text{mod } M$ 後的數量。

我們說兩種放城堡的方法不同若且為若兩種方法之間存在至少一個格子放的城堡顏色不一樣，或是在其中一個方法中有放城堡，另一種方法沒放城堡。

— 輸入 —

第一行有三個整數 N K M 。第二行有 K 個以空格分開的整數，第 i 個為 C_i

— 輸出 —

輸出一個正整數，代表答案。

— 輸入限制 —

- $1 \leq K \leq N \leq 2 \times 10^5$
- $2 \leq M \leq 10^9 + 7$
- $1 \leq C_i \leq N$

— 子任務 —

編號	分數	額外限制
1	0	範例輸入輸出
2	6	$K = N$
3	45	保證 M 是質數
4	13	$K = 1$
5	36	無額外限制

－ 範例輸入 1 －

5 1 87
3

－ 範例輸出 1 －

78

－ 範例輸入 2 －

10 3 877
1 2 3

－ 範例輸出 2 －

622

－ 範例輸入 3 －

10 5 877
1 2 3 4 5

－ 範例輸出 3 －

0

F. 部落衝突 2

Problem ID: tribe2

Time Limit: 1.0s

Memory Limit: 512MiB

— 題目描述 —

超級細胞國內有 N 棟房子，第 i 棟房子隸屬於編號 C_i 的部落。經過你鋪設道路後，現在有 $N - 1$ 條道路，第 i 條道路連接房子 U_i 和房子 V_i ，並且任兩棟房子可以經過一或多條的道路互相抵達。

野豬騎士身為超級細胞國的首領，他成功用橡皮筋狩獵到了大山豬了。為了慶祝這項喜事，野豬騎士打算呼叫超級細胞國內的所有居民到某一棟房子慶祝。然而，若慶祝地點選得不好會非常容易引發部落衝突，因此野豬騎士打算問你 Q 個問題。第 i 個問題會以「從 A_i 房子走到 B_i 房子這條路徑上，**第一個**遇到隸屬於 T_i 部落的房子是哪一棟？」這種形式出現，請對於每一次詢問，告訴野豬騎士正確的答案是多少。

我們說兩棟房子 a, b 能經過道路互相抵達，代表存在一系列相異的房子 p_0, p_1, \dots, p_t 滿足 $p_0 = a, p_t = b$ ，且所有 1 到 t 之間的整數 i ，第 p_{i-1} 棟房子和第 p_i 棟房子之間皆有鋪設道路。而上述 p_0, p_1, \dots, p_t 稱為 a 走到 b 的路徑，可以證明在本題限制下，任兩棟房子間只會有唯一一條路徑。若 k 為所有滿足 $C_{p_t} = c$ 中最小的整數 t ，那我們稱 p_k 為這條路徑上第一個隸屬於 c 部落的房子

— 輸入 —

第一行有兩個整數 N, Q 。

第二行有 N 個以空白分開的正整數，第 i 個為 C_i 。

接著有 $N - 1$ 行，第 i 行有兩個整數 U_i, V_i

接著有 Q 行，第 i 行有三個整數 A_i, B_i, T_i

— 輸出 —

輸出 Q 行，第 i 行代表第 i 次詢問的答案。如果這條路徑上沒有編號為 T_i 的部落，輸出 -1 。

— 輸入限制 —

- $2 \leq N \leq 10^5$
- $1 \leq Q \leq 10^5$
- $1 \leq U_i, V_i \leq N$
- $1 \leq A_i, B_i, T_i \leq N$
- $1 \leq C_i \leq N$
- 任兩棟房子可以經過一或多條的道路互相抵達。

— 子任務 —

編號	分數	額外限制
1	0	範例輸入輸出
2	3	$1 \leq N, Q, \leq 2000$
3	23	$U_i = i, V_i = i + 1$
4	37	$C_i \in \{1, 2\}$
5	37	無額外限制

— 範例輸入 1 —

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

— 範例輸出 1 —

```
1
5
```

— 範例輸入 2 —

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

— 範例輸出 2 —

```
3
-1
4
```

114 學年度資訊學科能力競賽臺南一中校內複選 試題本

附錄

祝各位都能破台，GL & HF