

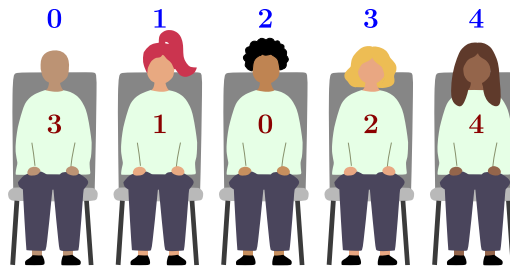
D. Seating Plan (seatingplan)

الحفل الختامي لـ EGOI دي هيحضره N ضيوف مهمين. كلهم محتاجين يقعدوا في الصف الأول بترتيب معين جداً عشان يناسب كل تفاصيل البروتوكول الدبلوماسي. تحديد ترتيب القاعد الصح أخذ من نوبي ليلتين من غير نوم.

فيرونيكا بتشرف على الحفل الختامي. واحدة من مسؤولياتها الكثير هي إنها تتأكد إن كراسي الصف الأول عليها فقط الأسماء الصح. بس فيه مشكلة صغيرة: نوبي مقالتلهاش على ترتيب القاعد الصح، ودلوقتي نوبي مش لاقينها في أي حطة. لحسن الحظ، دوركا المصورة معاها أبلكيشن ممكن يكون مفيد.

دوركا كان لازم تجهز كاميراتها عشان تقدر تاخذ صور معينة للضيوف اللي في الصف الأول. وعشان التجهيز، كانت محتاجة تعرف عرض كل صورة هيكون قد إيه، عشان كده نوبي عملتها أبلكيشن بيطلع المعلومات اللي هي محتاجها بسرعة. فيرونيكا دلوقتي عايزة تستخدم الأبلكيشن عشان تلاقى ترتيب القاعد الصح.

الـ N ضيوف المهمين مترقمين من 0 لحد $N - 1$. الكراسي في الصف الأول برضه مترقمة من 0 لحد $N - 1$ ، من الشمال لليمين. لكل I ($0 \leq I \leq N - 1$)، هنفترض إن g_I بيمثل الضيف اللي المفروض يقعد في الكرسي I ، وهنفترض إن s_I بيمثل الكرسي اللي الضيف I المفروض يقعد فيه.



شكل 1: صف فيه خمس ضيوف. للصف ده، $g = [3, 1, 0, 2, 4]$ و $s = [2, 1, 3, 0, 4]$.

الأبلكيشن بيشتغل كالتالي:

- دوركا بتدخل الأرقام I, J, K بتوع ثلاث ضيوف مختلفين بالظبط.
- الأبلكيشن بيقولها أقل عدد من الضيوف هيبقوا باينين لو الثلاث ضيوف اللي اختارهم موجودين في الصورة. بشكل رسمي، الأبلكيشن هيعرض القيمة $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

على سبيل المثال، شوف الموقف المتوضح في شكل 1:

- الضيوف $I = 0, J = 2$ ، و $K = 4$ قاعدين في الكراسي $s_I = 2, s_J = 3, s_K = 4$. لو دوركا اختارهم، الأبلكيشن هيعرض القيمة $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

بمعنى ثاني، أضيق صورة بتجمع الضيوف 0، 2، و 4 فيها الثلاث ضيوف دول بس.

- الضيوف $I = 0, J = 4$ ، و $K = 3$ قاعدين في الكراسي $s_I = 2, s_J = 4, s_K = 0$. لو دوركا اختارهم، الأبلكيشن هيعرض القيمة $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

بمعنى ثاني، أي صورة بتجمع الثلاث ضيوف دول لازم يكون فيها كل الـ 5 ضيوف.

ساعد فيرونيكا تحدد ترتيب القاعد الصح باستخدام أبلكيشن دوركا. بشكل أدق، الكود بتاعك المفروض يحدد ويطلع التسلسل g_0, g_1, \dots, g_{N-1} دايماً بيكون فيه إجابتين صح بالظبط (واحدة عكس الثانية)، وتقدر تطبع أي واحدة فهم. السكور بتاعك هيعتمد على عدد الأسئلة (أو الـ queries) للأبلكيشن اللي حلك بيعملها.

Implementation

دي مسألة تفاعلية (interactive problem). الكود بتاعك هيسخدم الـ standard input والـ output عشان يتواصل مع الجريدر (grader) بالشكل الموصوف تحت. ➡

الكود بتاعك لازم يبدأ بقراءة سطر واحد من الدخول فيه رقم صحيح موجب T ، عدد الـ test cases اللي جاية.

لكل test case، الكود بتاعك لازم يبدأ بقراءة سطر واحد من الدخول فيه رقم صحيح موجب N ، عدد الكراسي، والتي هو برضه عدد الضيوف.

عشان تعمل query، الكود بتاعك المفروض يطبع سطر بالشكل ده $I \quad J \quad K$ ؟ ، بحيث $0 \leq I, J, K \leq N - 1$ هما ثلاث أرقام مختلفين.

بعد ما تعمل query، الكود بتاعك لازم يقرأ سطر واحد فيه رقم صحيح موجب واحد، وهو الإجابة على الـ query بتاعتك.

عشان تجاوب بترتيب القواعد الصح، الكود بتاعك المفروض يطبع سطر بالشكل ده: $g_0 \dots g_{N-1}$!

بعد ما تحل كل الـ test cases T الكود بتاعك لازم يقفل بشكل طبيعي.

خد بالك إن الجريد الرسمي المستخدم في الـ CMS عشان يتست حلك ممكن يكون **adaptive** (أو متكيف). ده معناه إن في بعض الـ test cases، ترتيب الضيوف مبيكونش متحدد من الأول. بدال كده، الجريد ممكن يغير تصرفاته على حسب الـ queries المعينة اللي الكود بتاعك بيسألها.

الـ **Flushing**. لو مش بتستخدم الـ templates اللي متوفرة، اتأكد إنك تعمل flush لا standard output بعد ما تطبع كل سطر، وإلا ممكن الكود بتاعك يتحسب Not correct. في Python، ده بيحصل تلقائي لو استخدمت `input()` عشان تقرأ السطور، وتقدر تستخدم `print(..., flush=True)` عشان تجبره يعمل flush. في C++، أمر `cout << endl; flush` بيعمل flush بالإضافة لأنه بيطلع سطر جديد؛ لو بتستخدم `printf`، استخدم `fflush(stdout)`.

Constraints

- $1 \leq T \leq 10$.
- N هيكون 5 (كمثال بس)، 8، 40، أو 2000.
- لكل test case، تقدر تعمل كحد أقصى 10000 queries.

Scoring

الكود بتاعك هيتختبر على كذا test case متقسمين لـ subtasks. عشان تاخذ السكور بتاع أي subtask، لازم تحل كل الـ tests اللي جواه صح.

- المهمة الفرعية 0 [0 نقاط]: مثال ($N = 5$).
- المهمة الفرعية 1 [9 نقاط]: $N = 8$.
- المهمة الفرعية 2 [11 نقاط]: $N = 2000$ ، والضيوف 0 و 1 قاعدين جنب بعض.
- المهمة الفرعية 3 [15 نقاط]: $N = 40$.
- المهمة الفرعية 4 [65 نقاط]: $N = 2000$.

في الـ subtasks 1 و 2 أي حل بيحل كل الـ test cases صح هياخد كل النقط.

في الـ subtasks 3 و 4 حلك لازم يحل كل الـ test cases صح عشان تاخذ أي نقط، والسكرور بتاعك هيعتمد على Q_s ، والتي هو أكبر عدد من الـ queries حلك احتاجه عشان يحل test case. بفرض إن $X_s = \max(1, Q_s/N)$ ، السكرور للـ subtasks 3 و 4 بيتحسب كالتالي:

$$\text{score}_3 = \min\left(15, 3 + \frac{19}{X_s^{1.5}}\right), \quad \text{score}_4 = \min\left(65, 3 + \frac{91}{X_s^{1.5}}\right)$$

قيمة الـ score_s بتتقرب لأقرب رقم صحيح لكل subtask. والسكرور الإجمالي بتاعك هو مجموعهم. عشان تاخذ السكرور الكامل، لازم تحل subtask 3 في 55 query بالكثير و subtask 4 في 2597 query بالكثير. أمثلة لقيم Q_s والسكرور للـ subtasks 3 و 4 متوضحة تحت.

10000	150	100	80	70	60	56	55	Q_s
3	6	8	10	11	13	14	15	score_3

10000	8000	6000	5000	4000	3000	2800	2597	Q_s
11	14	21	26	35	53	58	65	score_4

مقيّم	الحل
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Explanation

الدخل بتاع المثال فيه test case واحدة ($T = 1$) بـ $N = 5$ ضيوف. الترتيب المخفي للضيوف في الـ test case دي بي مطابق شكل 1. أول query اتعملت في حل المثال هي 0, 2, 4. الإجابة 3 للـ query دي بتقولنا إن الضيوف دول قاعدين. بترتيب مش معروف، في ثلاث كراسي ورا بعض جنب بعض.

الإجابة 3 للـ query الثانية بتقولنا نفس الحاجة عن الضيوف 3, 0, 1.

دلوقتي نقدر نستنتج إن الضيف 0 لازم يقعد في النص، مع الضيوف 2 و 4 على ناحية والضيوف 1 و 3 على الناحية الثانية.

بعد الـ query الثالثة، نقدر نكون متأكدين إن الضيوف لازم يكونوا قاعدين يا إما بترتيب [3, 1, 0, 2, 4] أو بالترتيب المعكوس [4, 2, 0, 1, 3]. نقدر نطبع أي ترتيب فهم.

Code Templates و تفاصيل التقييم في الـ CMS

بننصحك بشدة إنك تستخدم الـ code templates اللي متوفرة لـ C++ و Python. دول بيتأكدوا إن التواصل مع الجريدنر نجح وبيقفلوا البرنامج بشكل نظيف لو في مشكلة.

الجريدنر اللي بيتواصل مع حلك يبيلغ عن أول إيروور (error) يلاقيه وبعدين بيقفل. لو مش بتستخدم الـ templates دي، ممكن ده يخلي حلك يعمل كراش (crash) أو يفضل مستني رد. لو ده حصل، غالباً هتاخذ نتيجة تقييم مش مفيدة في الـ CMS زي Execution timed out (wall clock limit exceeded).

بننصحك كمان بأداة التست (شوف تحت) عشان تتست حلك على جهازك (أو locally) قبل ما ترفعه. أداة التست بتراجع المخرجات بتاعت حلك وبتبلغك لو فيه أي مخالفات للبروتوكول.

أداة التست (أو Testing Tool)

عشان نسهل عليك تست التست للحل بتاعك، بنوفر أداة بسيطة تقدر تحملها من الـ CMS. استخدام الأداة دي اختياري. خد بالك إن الجريدنر الرسعي على الـ CMS مختلف عن أداة التست دي.

عشان تستخدم الأداة دي هتحتاج ملف input. تقدر تستخدم مثال الـ input اللي متوفر seatingplan.input0.txt أو تعمل واحد بتاعك. ملف الـ input لازم يبدأ بسطر فيه عدد الـ T بتوع الـ test cases وبعدين لازم يكون فيه سطرين لكل test case: سطر فيه الرقم N وبعدين سطر فيه الأرقام g_0, g_1, \dots, g_{N-1} .

لبرامج Python، مثلاً seatingplan.py (اللي بتشغل عادة بـ pypy3 seatingplan.py) شغل أداة التست كالتالي:

```
python3 testing_tool.py pypy3 seatingplan.py < seatingplan.input0.txt
```

لبرامج C++، الأول اعمل compile للحل بتاعك:

```
g++ -DEVAL -std=gnu++20 -O2 -pipe -static -s -o seatingplan seatingplan.cpp
```

وبعدين شغل أداة التست:

```
python3 testing_tool.py ./seatingplan < seatingplan.input0.txt
```