

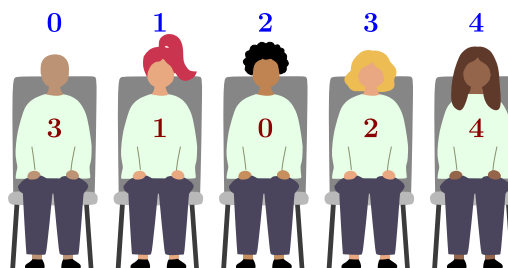
D. תוכנית ישיבה (seatingplan)

לטקס הסיום של ה-EGOI הזה יגיעו N אורחות חשובות. כולן צריכות לשבת בשורה הראשונה בסדר מאוד מסוים שתואם את כל הדקויות של הפרוטוקול הדיפלומטי. קביעת סדר הישיבה הנכון לקחה לנעמי שני לילות ללא שינה.

ורוניקה מפקחת על טקס הסיום. אחת מהאחריות הרבות שלה היא לוודא שעל הכיסאות בשורה הראשונה יש את שלטי השמות הנכונים. יש רק בעיה אחת קטנה: נעמי מעולם לא אמרה לה מהו סדר הישיבה הנכון, ועכשיו אי אפשר למצוא את נעמי בשום מקום. למרבה המזל, לדורקה הצלמת יש אפליקציה שעשויה להיות שימושית.

דורקה הייתה צריכה להכין את המצלמות שלה כדי שתוכל לצלם כמה תמונות ספציפיות של האורחות בשורה הראשונה. בשביל ההכנות, היא הייתה צריכה לדעת כמה רחבה תהיה כל תמונה, אז נעמי הכינה לה אפליקציה שמוציאה במהירות את המידע שהיא צריכה. עכשיו ורוניקה רוצה להשתמש באפליקציה כדי למצוא את סידור הישיבה הנכון.

N האורחות החשובות ממספרות מ-0 עד $N - 1$. הכיסאות בשורה הראשונה ממוספרים גם הם מ-0 עד $N - 1$, משמאל לימין. עבור כל I ($0 \leq I \leq N - 1$), נסמן ב- g_I את האורחת שאמורה לשבת בכיסא I , ונסמן ב- s_I את הכיסא שבו אורחת I אמורה לשבת.



איור 1: שורה עם חמש אורחות. עבור השורה הזו, $g = [3, 1, 0, 2, 4]$ ו- $s = [2, 1, 3, 0, 4]$.

האפליקציה עובדת באופן הבא:

- דורקה מזינה את המספרים I, J, K של בדיוק שלוש אורחות שונות.
- האפליקציה אומרת לה מהו מספר האורחות המינימלי שיופיעו בתמונה אם כל שלוש האורחות שנבחרו יהיו בה. באופן פורמלי, האפליקציה תציג את הערך $1 + \max(s_I, s_J, s_K) - \min(s_I, s_J, s_K)$.

לדוגמה, תסתכלי על המצב שמוצג באיור 1:

- האורחות $I = 0, J = 2, K = 4$ נמצאות בכיסאות $s_I = 2, s_J = 3, s_K = 4$. אם דורקה תבחר אותן, האפליקציה תציג את הערך $3 = \max(2, 3, 4) - \min(2, 3, 4) + 1$.

במילים אחרות, התמונה הכי צרה שמכילה את אורחות 0, 2 ו-4 תכלול בדיוק את שלוש האורחות האלו.

- האורחות $I = 0, J = 4, K = 3$ נמצאות בכיסאות $s_I = 2, s_J = 4, s_K = 0$. אם דורקה תבחר אותן, האפליקציה תציג את הערך $5 = \max(2, 4, 0) - \min(2, 4, 0) + 1$.

במילים אחרות, תמונה שמכילה את שלוש האורחות הנתונות חייבת להכיל את כל 5 האורחות.

עזרי לוורוניקה למצוא את סדר הישיבה הנכון בעזרת האפליקציה של דורקה. ליתר דיוק, התוכנית שלך צריכה למצוא ולהדפיס את הסדרה g_0, g_1, \dots, g_{N-1} . תמיד יש בדיוק שתי תשובות נכונות (האחת היא ההיפוך של השנייה), ואת יכולה להדפיס כל אחת מהן. הציון שלך יהיה תלוי במספר השאלות לאפליקציה שהפתרון שלך יעשה.

מימוש

זוהי בעיה אינטראקטיבית. התוכנית שלך תשתמש בקלט ופלט סטנדרטיים כדי לתקשר עם ה-grader בפורמט שמתואר למטה. ➡

התוכנית שלך צריכה להתחיל בקריאת שורת קלט אחת שמכילה מספר שלם חיובי T , מספר הטסטים שיבואו אחר כך. עבור כל טסט, התוכנית שלך צריכה להתחיל בקריאת שורת קלט אחת שמכילה מספר שלם חיובי N , מספר הכיסאות, שהוא גם מספר האורחות.

כדי לשאול שאילתה, על התוכנית שלך להדפיס שורה בצורה " $I \ J \ K \ ?$ ", כאשר $0 \leq I, J, K \leq N - 1$ הם שלושה מספרים שונים.

אחרי שאילת שאילתה, התוכנית שלך צריכה לקרוא שורה אחת שמכילה מספר שלם חיובי, התשובה לשאילתה שלך.

כדי לענות עם סדר הישיבה הנכון, התוכנית שלך צריכה להדפיס שורה בצורה " $g_0 \dots g_{N-1} !$ ".

אחרי פתרון כל T הטסטים, התוכנית שלך צריכה להסתיים כרגיל.

שימי לב שה-grader הרשמי שמשמש את מערכת ה-CMS לבדיקת הפתרון שלך עשוי להיות **אדפטיבי** (מסתגל). כלומר, עבור חלק מהטסטים, התמורה (permutation) של האורחות לא נקבעת מראש. במקום זאת, ה-grader יכול להחליט באיזו מהתמורות שנשארו הוא משתמש כתלות בשאילתות שכבר נשאלו על ידי התוכנית שלך.

Flushing. אם את לא משתמשת בתבניות שסופקו, ודאי שאת עושה flush לפלט הסטנדרטי אחרי הדפסת כל שורה, אחרת התוכנית שלך עלולה להשפט כ-*Not correct*. ב-Python, זה קורה אוטומטית אם את משתמשת ב-input() כדי לקרוא שורות, ואת יכולה להשתמש ב-print(..., flush=True) כדי לאלץ אותה לעשות flush. ב-C++, השימוש בפקודה cout << endl; עושה flush בנוסף להדפסת שורה חדשה; אם את משתמשת ב-printf, השתמשי ב-fflush(stdout).

אילוצים

- $1 \leq T \leq 10$.
- N יהיה 5 (רק בדוגמה), 8, 40, או 2000.
- עבור כל טסט, את יכולה לבצע לכל היותר 10000 שאילתות.

ניקוד

התוכנית שלך תיבדק על מספר טסטים שמקובצים לתתי-משימות. כדי לקבל את הציון על תת-משימה, את חייבת לפתור נכון את כל הטסטים שהיא מכילה.

- תת-משימה 0 [0 נקודות]: דוגמה ($N = 5$).
- תת-משימה 1 [9 נקודות]: $N = 8$.
- תת-משימה 2 [11 נקודות]: $N = 2000$, והאורחות 0 ו-1 יושבות זו לצד זו.
- תת-משימה 3 [15 נקודות]: $N = 40$.
- תת-משימה 4 [65 נקודות]: $N = 2000$.

עבור תתי-משימות 1 ו-2 כל פתרון שיפתור נכון את כל הטסטים יקבל את מלוא הנקודות.

עבור תתי-משימות 3 ו-4 הפתרון שלך חייב לפתור נכון את כל הטסטים כדי לקבל נקודות, והציון שלך יהיה תלוי ב- Q_s , המספר הגדול ביותר של שאילתות שהפתרון שלך היה צריך לשאול כדי לפתור טסט. נגדיר $X_s = \max(1, Q_s/N)$. הציונים לתתי-משימות 3 ו-4 מחושבים באופן הבא:

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

הערך של score_s מעוגל למספר השלם הקרוב ביותר בכל תת-משימה, והציון הכולל שלך הוא סכום הציונים האלו. כדי לקבל ציון מלא, את צריכה לפתור את תת-משימה 3 בלא יותר מ-55 שאילתות ואת תת-משימה 4 בלא יותר מ-2597 שאילתות. ערכים לדוגמה של Q_s והניקוד לתת-משימות 3 ו-4 מוצגים למטה.

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

דוגמאות

Grader	Solution
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

הסבר

קלט הדוגמה מכיל טסט אחד ($T = 1$) עם $N = 5$ אורחות. סידור האורחות המוסתר בטסט זה תואם לאיור 1. השאילתה הראשונה ששואל פתרון הדוגמה היא 0, 2, 4. התשובה 3 לשאילתה הזו מגלה לנו שהאורחות האלו יושבות, בסדר כלשהו לא ידוע, בשלושה כיסאות סמוכים זו לצד זו. התשובה 3 לשאילתה השנייה אומרת לנו את אותו הדבר על האורחות 3, 0 ו-1. אנחנו יכולות כעת להסיק שאורחת 0 חייבת לשבת באמצע, עם אורחות 2 ו-4 בצד אחד ואורחות 1 ו-3 בצד השני. אחרי השאילתה השלישית, אנחנו כבר יכולות להיות בטוחות שהאורחות חייבות לשבת בסדר [3, 1, 0, 2, 4] או בסדר ההפוך [4, 2, 0, 1, 3]. אנחנו יכולות להדפיס כל אחד מהסדרים האלו.

תבניות קוד ופרטי בדיקה ב-CMS

אנחנו ממליצות מאוד להשתמש בתבניות הקוד (code templates) שסופקו עבור C++ ו-Python. התבניות האלו בודקות האם התקשורת עם ה-grader הצליחה, ומסיימות את הריצה בצורה מסודרת כשהיא נכשלת. אם את לא משתמשת בתבניות שסופקו, במקרים שהפתרון שלך שגוי, ה-CMS עלול להציג את החיווי (verdict) הלא נכון. לדוגמה, במקום "Output isn't correct" את עלולה לקבל "Execution killed by signal" או "Execution timed out" (wall clock limit exceeded).

אנחנו גם ממליצות להשתמש בכלי הבדיקה (testing tool, ראי למטה) כדי לבדוק את הפתרון שלך מקומית (locally) לפני ההגשה. כלי הבדיקה מוודא את הפלטים של הפתרון שלך ומדווח על שימוש בשאילתות לא חוקיות.

כלי בדיקה (Testing Tool)

כדי להקל על בדיקת הפתרון שלך, אנחנו מספקות כלי פשוט שאת יכולה להוריד מ-CMS. השימוש בכלי הוא בגדר רשות. שימי לב שה-grader הרשמי ב-CMS שונה מכלי הבדיקה.

כדי להשתמש בכלי, את צריכה קובץ קלט. את יכולה להשתמש בקלט הדוגמה שסופק seatingplan.input0.txt או ליצור אחד משלך. קובץ הקלט צריך להתחיל עם שורה שבה יש את המספר T של הטסטים, ואז צריכות להיות שתי שורות לכל טסט: שורה אחת עם המספר N ואז שורה אחת עם המספרים g_0, g_1, \dots, g_{N-1} .

עבור תוכניות ב-Python, למשל `seatingplan.py` (שבדרך כלל מריצים כ-`pypy3 seatingplan.py`) הריצי את כלי הבדיקה בצורה הבאה:

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

עבור תוכניות ב-C++, קודם קמפלי את הפתרון שלך:

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

ואז הריצי את כלי הבדיקה:

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