beCP
2026
Tâche 1.1: Les Serrures Tournantes (rotation)

Auteur: Bruno Ploumhans

Limite de temps: 1 s Limite mémoire: 16 MB

Note: Cette tâche est interactive. Veuillez consulter les instructions spéciales pour implémenter, compiler et tester votre programme. N'hésitez pas à demander de l'aide à un surveillant.

Dans l'ancienne ville fortifiée de Khiva, en Ouzbékistan, des archéologues ont découvert une voûte contenant N serrures disposées en cercle, numérotées de 1 à N autour de l'anneau. Maintenant, ils demandent votre aide. Voici les informations qu'ils ont recueillies. Autrefois, chaque serrure avait reçu une valeur secrète distincte. Ces valeurs sont simplement les nombres de 1 à N, mais elles ont été tournées ; c'est-à-dire que la séquence 1, 2, 3, ..., N a été placée sur le cercle et ensuite tournée selon un décalage inconnu k. Ainsi, la serrure numéro 1 contient la valeur $k+1$, la serrure numéro 2 contient $k+2$, et ainsi de suite, en revenant à 1 après N...

La voûte s'ouvre lorsque vous identifiez le décalage k auquel le cercle a été tourné. Malheureusement, les gardiens locaux ne vous permettent pas d'inspecter directement la valeur d'une serrure, prétendant que vous pourriez endommager le mécanisme. Au lieu de cela, vous ne pouvez demander à un gardien de répondre qu'à des questions de comparaison : vous lui donnez deux positions de serrure i et j, et il vous indique laquelle des deux contient la plus petite valeur. Les gardiens ont une patience limitée, donc avec un nombre limité de questions, votre tâche consiste à déterminer de combien le cercle a été tourné.

Protocole d'interaction

Vous devez implémenter une fonction `solve(int N)` qui sera appelée par le correcteur. Votre fonction `solve` recevra l'entier N en argument. Dans cette fonction, vous pouvez utiliser la fonction suivante fournie par la bibliothèque d'interaction :

- **`int ask_guardian(int position1, int position2)`**.
  - Appelez cette fonction avec deux positions. Les deux positions doivent être comprises entre 1 et N, inclus, et différentes l'une de l'autre.

— La fonction retourne l'une des deux positions données : celle de la serrure qui contient la plus petite valeur.

— Vous pouvez appeler cette fonction au maximum **60** fois.

Une fois que vous avez déterminé le décalage k par lequel les nombres sur les serrures ont été tournés, votre fonction `solve` doit le retourner pour signaler votre réponse. Vous devez vous assurer que $0 \leq k < N$.

Limites générales

— $2 \leq N \leq 100\,000\,000$

— $0 \leq k < N$

— Vous pouvez appeler `ask_guardian` au maximum 60 fois.

Contraintes supplémentaires

Sous-tâche	Points	Contraintes
A	27	$2 \leq N \leq 60$.
B	24	$2 \leq N \leq 100$.
C	49	Pas de contrainte supplémentaire.

Exemple d'interaction

Voici un exemple d'interaction pour $N = 5$ et $k = 4$. Dans ce cas, les serrures ont les valeurs suivantes :

Position de la Serrure	**Valeur de la Serrure**
1	5
2	1
3	2
4	3
5	4

Supposons que nous décidions d'effectuer la série suivante d'appels :

Votre Appel	Valeur de Retour	Rapport du Gardien
`ask_guardian({1, 2})`	`2`	La plus petite des deux serrures est à la position 2.
`ask_guardian({2, 3})`	`2`	La plus petite des deux serrures est à la position 2.
`ask_guardian({3, 4})`	`3`	La plus petite des deux serrures est à la position 3.
`ask_guardian({2, 5})`	`2`	La plus petite des deux serrures est à la position 2.

À partir de cette interaction, nous déduisons que puisque la serrure à la position 2 est plus petite que toutes les autres elle doit avoir une valeur de 1. Nous pouvons donc déduire que les nombres ont été tournés de $k = 4$.

Test Local avec le Correcteur d'Exemple

Comment l'utiliser

1. Sauvegardez les fichiers `rotation.h` et `grader.cpp` dans le même répertoire que votre fichier de solution (qui doit être nommé `rotation.cpp`).
2. Compilez le correcteur d'exemple avec votre solution en utilisant :

```
g++ -std=gnu++11 -O2 grader.cpp rotation.cpp -o rotation
```

3. Exécutez le programme compilé :

```
./rotation
```

4. Le correcteur d'exemple vous invitera à entrer la valeur de N et k (par exemple, `5 4`). Entrez ces valeurs séparées par des espaces.
5. Votre fonction `solve` sera alors appelée et interagira avec l'implémentation de `ask_guardian()` du correcteur d'exemple. Le correcteur d'exemple affichera le résultat de votre soumission.
6. Alternativement, vous pouvez utiliser le fichier d'exemple :

```
./rotation < sample1.in
```

Remarques importantes :

— Le correcteur d'exemple est destiné uniquement aux tests locaux et pourrait ne pas appliquer toutes les contraintes du correcteur officiel (par exemple, la limite exacte de requêtes).

— Vous ne devez soumettre que votre fichier `rotation.cpp` au CMS.