beCP
2018
Tâche 1.1: Énigme japonaise (riddle)

Auteur: Victor Lecomte
Limite de temps: 2 s Limite mémoire: 512 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.

Lors d'une expédition au cœur des montagnes japonaises, vous rencontrez une vieille dame accroupie au bord du sentier. Cette vieille dame vous pose une énigme : elle a en tête un tableau p de taille n qui contient les nombres $0, \ldots, n-1$ chacun exactement une fois, et vous devez le deviner.

En raison des difficultés linguistiques, vous pouvez seulement lui poser deux types de questions. Pour chaque question vous lui donnez deux indices i et j, des entiers entre 0 et $n-1$. Sa réponse dépend du type :

- Type A : Elle vous répond **`true`** si $p[i] < p[j]$, et **`false`** sinon.
- Type B : Elle vous répond **`true`** si $p[i]$ & $p[j]$ est strictement positif (où & est l'opération ET bit-à-bit), et **`false`** sinon.

Pour rappel, le ET bit-à-bit est une opération qui se base sur la représentation binaire des nombres. Le résultat de a & b est un nombre qui à chaque position de sa représentation binaire a un 1 si a et b ont tous les deux un 1 à cette position, et un 0 sinon. Par exemple, $a = 12$ et $b = 5$ sont représentés par `1100` et `0101`, donc le résultat de 12 & 5 est `0100`, c'est-à-dire 4.

Par conséquent, pour les questions de type B, la vieille dame vous répond **`true`** si et seulement si il existe une position à laquelle les représentations binaires de $p[i]$ et $p[j]$ ont toutes les deux un 1.

Bien sûr, si vous pouvez poser autant de questions que vous voulez, c'est trop facile, donc la vieille dame vous limite à $N_{\texttt{askLt}}$ questions de type A et $N_{\texttt{askAnd}}$ questions de type B. Pouvez-vous deviner le tableau sous ces limites ?

1 Fonctions à implémenter

C++	`vector<int> guess(int n)`
Java	`int[] guess(int n)`
	Étant donnée la taille n, fait appel à `askLt(i,j)` et `askAnd(i,j)` pour deviner le tableau p.
`return`	Un tableau de taille n, le tableau p deviné.

2 Fonctions à appeler

C++	`bool askLt(int i, int j)`
Java	`boolean askLt(int i, int j)`
	Pose une question de type A pour les indices i, j.
`return`	La réponse à la question : `p[i] < p[j]`.

C++	`bool askAnd(int i, int j)`
Java	`boolean askAnd(int i, int j)`
	Pose une question de type B pour les indices i, j.
`return`	La réponse à la question : `(p[i] & p[j]) > 0`.

3 Limites générales

— $n = 2^{11} = 2048$ pour tous nos tests, mais nous vous conseillons d'utiliser des valeurs plus petites quand vous débuguez votre programme ;
— n est une puissance de 2 ;
— pour tous nos tests, le tableau p a été tiré au hasard (mais il reste le même d'une soumission à l'autre).

4 Contraintes supplémentaires

Sous-tâche	Points	$N_{\texttt{askLt}}$	$N_{\texttt{askAnd}}$
A	10	5 000 000	5 000 000
B	10	50 000	5 000 000
C	35	200	5 000 000
D	20	200	600 000
E	15	200	100 000
F	10	200	35 000

Chaque sous-tâche contient 10 tests. Vous obtiendrez les points de la sous-tâche uniquement si votre programme passe chacun des tests correctement et en respectant les limites de questions.

5 Exemple d'interaction

La taille est $n = 4$, et le tableau à trouver est $p = \{2, 3, 1, 0\}$. Votre fonction est donc appelée avec `guess(4)`. Elle effectue cette suite d'appels

aux fonctions `askLt(i,j)` et `askAnd(i,j)` :

Appel	Résultat	Commentaires
`askLt(0,1)`	**true**	$2 < 3$
`askAnd(0,1)`	**true**	$2 \,\&\, 3 = 2$
`askAnd(2,3)`	**false**	$1 \,\&\, 0 = 0$
`askAnd(2,0)`	**false**	$1 \,\&\, 2 = 0$
`askLt(3,3)`	**false**	$0 \geq 0$
`askAnd(1,1)`	**true**	$3 \,\&\, 3 = 3$
`askAnd(3,1)`	**false**	$0 \,\&\, 3 = 0$
`askAnd(1,2)`	**true**	$3 \,\&\, 1 = 1$
`askAnd(0,3)`	**false**	$2 \,\&\, 0 = 0$
`askLt(2,0)`	**true**	$1 < 2$
`askLt(0,2)`	**false**	$2 \geq 1$

Les informations obtenues sont suffisantes pour déduire que le tableau est $\{2, 3, 1, 0\}$, donc c'est ce que votre fonction renvoie. Bien sûr ceci n'est qu'un exemple, et vous êtes libre d'utiliser les questions comme bon vous semble.

6 Implémentation, compilation et test

Vous devez implémenter la fonction `guess(n)` qui se trouve dans le fichier `riddle.cpp/java`. Pour compiler puis exécuter, utilisez les commandes suivantes :

C++	`g++ -std=c++11 -Wall grader.cpp riddle.cpp` `./a.out < riddle.in`
Java	`javac grader.java` `java grader < riddle.in`

Pour tester votre code sur plusieurs cas possibles, vous pouvez modifier le fichier `riddle.in`. La première ligne de ce fichier contient n, et la deuxième ligne peut soit contenir le tableau p à deviner, soit être vide, dans quel cas le tableau est tiré au hasard.

`riddle.in (spécifié)`

```
4
2 3 1 0
```

`riddle.in (aléatoire)`

```
16
```

7 Soumission et verdict

— Vous ne devez soumettre que le fichier `riddle.cpp/java`.
— Le verdict "Wrong Answer" peut signifier soit que vous avez utilisé trop de questions d'un type, soit que le tableau que vous renvoyez est incorrect.
— N'imprimez rien vers `stdout` dans votre programme : n'utilisez jamais `cout`, `printf()` ou `System.out.println()`.