





ТРЕНИРОВЪЧНО СЪСТЕЗАНИЕ

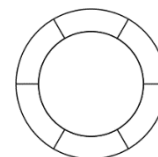
НАЦИОНАЛНА ШКОЛА ПО ИНФОРМАТИКА

Ловеч, 22 август 2023 г.
група С

 : 0,2 сек.
 : 256 MB

Задача Т2. КОРИДОР

Стаята, в която Криси се намира, представлява кръгъл коридор, разделен на N последователни сектора (виж фигурата вдясно) – в началото тя се намира в един тях. Във всеки от тези сектори има по една лампа. Всяка от лампите в началото е включена или изключена. Във всеки момент Криси вижда лампата в сектора, в който се намира (знае дали е включена или изключена), но не вижда нито една от другите лампи в коридора. Тя има право да извършва следните стъпки:



1. Да се премести в следващия по посока на часовниковата стрелка сектор.
2. Да се премести в следващия по посока обратна на часовниковата стрелка сектор.
3. Да включи/изключи лампата в сектора, в който се намира (ако лампата е включена може да я изключи, а ако е изключена – да я включи).
4. Да се опита да познае на колко е равно N .

Целта на Криси е да познае правилно стойността на N . Помогнете ѝ като реализирате алгоритъм, който тя да ползва. Понеже главата ѝ още се мотае, тя не иска да извършва твърде много стъпки и също така не може да помни много на брой неща – вашият алгоритъм ще е ограничен от страна на брой позволени стъпки и памет, която може да ползвате между отделните стъпки (описано подробно по-долу).

Детайли по имплементацията

Напишете програма **corridor**, която да съдържа функцията **solve** със следния формат:
`step solve(char cellValue);`

Тя ще бъде извиквана многократно, симулирайки вашия алгоритъм стъпка по стъпка. Целта ѝ е да върне резултат, който описва каква е следващата стъпка при изпълнението на алгоритъма. При всяко извикване като аргумент ще ѝ бъде подавана стойност '0', ако лампата в сектора на коридора, в който се намирате, е включена, и '1' – ако е изключена. Функцията **solve** трябва да връща като резултат стойност от тип **step**, дефиниран по следния начин (**не трябва да го дефинирате сами**):

```
struct step {  
    char action;  
    int answer;  
};
```



1. За да укажете, че следващата Ви стъпка е преместване в следващия по посока на часовниковата стрелка сектор, трябва да върнете за стойност на **action** - 'l' (**answer** не се взима предвид).
2. За да укажете, че следващата Ви стъпка е преместване в следващия по посока обратна на часовниковата стрелка сектор, трябва да върнете за стойност на **action** - 'r' (**answer** не се взима предвид).
3. За да укажете, че следващата стъпка е включване/изключване на лампата в текущия сектор на коридора, трябва да върнете за стойност на **action** - 't' (**answer** не се взима предвид).
4. За да укажете, че следващата стъпка е опит за познаване на N , трябва да върне за стойност на **action** - 'a', а за стойност на **answer** - верния според Вас отговор. Имате право на един опит за познаване на броя сектори на даден коридор.
5. Счита за грешка, ако **solve** върне резултат, в който **action** не е нито една от описаните по-горе стойности.



ТРЕНИРОВЪЧНО СЪСТЕЗАНИЕ

НАЦИОНАЛНА ШКОЛА ПО ИНФОРМАТИКА

Ловеч, 22 август 2023 г.
група С

 : 0,2 сек.
 : 256 MB

Тестовите ще бъдат групирани в групи от по K коридора. В рамките на един тест, вашето решение ще бъде симулирано върху всеки един от тях, като симулациите на отделните коридори ще бъдат преплетени една с друга – възможно е две последователни извиквания на `solve` да са за различни коридори (например може първите 2 извиквания да са за коридор 1, следващите 2 – за коридор 2, следващите 2 – отново за коридор 1 и т.н.). Всяка група носи точки, ако за всеки от коридорите ѝ е даден верен отговор.

Поради преплитането на симулациите на решението, е предоставен начин да имате памет, локална за всяка от тях. Всяка от симулациите разполага с памет в размер на 5 променливи от тип `int` (в началото на симулация, стойностите им са 0). Във функцията `solve` имате достъп до тази памет чрез функциите `get_memory` и `set_memory` (**не трябва да ги дефинирате сами**):

```
int get_memory (int index);  
void set_memory (int index, int value);
```

Извикването `get_memory(i)`, връща стойността на i -тата (броейки от 0) променлива от паметта за текущата симулация. Извикването `set_memory(i, x)` присвоява стойност на i -тата променлива от паметта за текущата симулация равна на x . Последното ограничение за алгоритъма ви е, че при две негови симулации над еднакви коридори (с еднаква начална конфигурация на лампите и един и същ начален сектор), той трябва да работи по един и същ начин – трябва да прави едни и същи поредици от стъпки и поредици от извиквания на `set_memory` в двете симулации. Ако в рамките на един тест това не е вярно, тестът се счита за грешен. **Това значи, че всякакво съществено ползване на глобална памет би довело до грешно решение.**

Вашата програма `corridor.cpp` трябва да имплементира функцията `solve`. Тя може да съдържа и друг код, и функции, необходими за работата Ви, но не трябва да съдържа главната функция `main`. Също така, не трябва да четете от стандартния вход или да отпечатвате на стандартния изход. Програмата Ви трябва да включва хедър файла `corridor.h` чрез указание към компилатора:



```
#include "corridor.h"
```

Ограничения

- ♣ $1 \leq K \leq 5$
- ♣ $1 \leq N \leq 200$



ТРЕНИРОВЪЧНО СЪСТЕЗАНИЕ
НАЦИОНАЛНА ШКОЛА ПО ИНФОРМАТИКА
Ловеч, 22 август 2023 г.
група С

 : 0,2 сек.
 : 256 MB

Подзадачи

Подзадача	Точки	N	Максимален брой стъпки	Максимален брой извикване на <code>set_memory</code>
1	50	≤ 200	42000	84000
2	20	≤ 50	500	1000
3	20	≤ 200	26000	52000
4	10	≤ 200	20000	40000

Точките за дадена подзадача се пресмятат като се вземе минималната оценка на тест за съответната подзадача и се умножи по броя точки на подзадачата. Всеки тест, по времето на който програмата не е спазила гореописания протокол за комуникация или е намерила грешно N , се счита за неуспешен и получава оценка 0. В противен случай, тестът се счита за успешен и получава оценка $\min(1, \frac{60}{ct})$, където ct е максималният брой пъти, в които изпълняват `action` равно на t за симулацията върху някой коридор в теста.

Локално тестване

За локално тестване са Ви предоставени файловете `corridor.h` и `Lgrader.cpp`. Сложете ги в същата папка, в която е Вашият файл `corridor.cpp` и компилирайте `Lgrader.cpp`. Така ще получите програма, с която ще проверите верността на функцията `Vi`. След стартиране на програмата, въведете на един ред низ, съставен от символите '0' и '1' (началният сектор е първият въведен символ). Програмата ще ви каже дали коректно сте намерили дължината, или не, както и дали не правите прекалено много стъпки или извиквания на `set_memory`.