Асансьор

Време: 10 seconds
Памет: 64 MiB (на процес)

Това е комуникационна задача.

В една сграда има етажи, номерирани от 0 до N. Етаж 0 е приземният етаж, над етаж N е покрива на сградата. Така, включвайки покрива, сградата се разделя на общо $N + 2$ нива.

На всеки етаж f ($0 \leq f \leq N$) живее точно един жител, който има тайно число v_f ($0 \leq v_f \leq 3$), което само той знае. Допълнително, на покрива живее Карлсон. Неговата цел е да научи, колкото се може повече от стойностите $v_0, v_1, \ldots, v_N$ с помощта на жителите на долните етажи, които ще му помагат с това намерение.

За да комуникират, жителите ще ползват асансьора по следния начин.

Асансьорът тръгва от етаж 0 и пътува единствено нагоре. В асансьора има бутони за всеки етаж от 1 до N. В началото няма натиснати бутони. Веднъж натиснат, бутон остава натиснат завинаги Асансьорът спира на етаж f и отваря вратите си, единствено ако $f = 0$ или бутонът за етаж f е бил натиснат по-рано. След като посети най-високия етаж, чийто бутон е натиснат (или етаж 0, ако няма натиснати бутони), асансьора продължава директно до покрива без да спира, който и да е оставащ етаж.

Когато асансьора спре на етаж f се случва следното:

1. Жителят на този етаж вижда всички бутони, които са натиснати до момента;
2. Знаейки само тази информация и стойността на v_f, той избира някои от досега ненатиснатите бутони за етажи **стриктно над** неговия. Жителят натиска всички така избрани бутони.
3. Асансьорът продължава да се движи нагоре до следващия етаж, чийто бутон е натиснат, или покрива, ако няма такъв етаж.

Ако асансьора не спре на някой етаж, съответният жител не може да натисне бутони в него.

Когато асансьорът достигне покрива, Карлсон ще види за всеки бутон дали някога е бил натиснат от жител по-долу. Ползвайки единствено тази информация, той се опитва да възстанови, колкото се може повече стойности $v_0, v_1, \ldots v_N$.

Стойностите на v са фиксирани преди началото на програмата и не се променят.

Детайли по имплементацията

Ще трябва да решите T тестови случая.

Ще трябва да изпратите един файл, в който са имплементирани две функции.

Първата функция, която трябва да имплементирате, е:

```
std::vector<int> press_buttons(int subtask, int N,
                               int f, int v, std::vector<int> p)
```

- subtask: подзадачата, към която принадлежи тестовия случай, където $0 \leq \text{subtask} \leq 4$;
- N: номера на последния етаж;
- f: текущият етаж, където $0 \leq f \leq N$;
- v: стойността на v_f за етаж f;
- p: натиснатите бутони, подредени в нарастващ ред.

Тази функция се извиква, когато асансьора спира на етаж f. Това се случва ако $f = 0$ или ако съответния бутон е бил натиснат по-рано. Функцията не се извиква за етажи, на които асансьорът не е спирал.

Функцията връща списък от бутоните, които жителя на текущия етаж натиска. Подредбата на стойностите в списъка не е важна. Всеки натиснат бутон x трябва да изпълнява следните условия:

- $f < x \leq N$;
- x не принадлежи на p и x се среща най-много веднъж във върнатия списък.

Втората функция, която трябва да имплементирате, е:

```
std::vector<int> answer(int subtask, int N, std::vector<int> p)
```

- subtask: подзадачата, към която принадлежи тестовия случай, където $0 \leq \text{subtask} \leq 4$;
- N: номера на последния етаж;
- p: финалното множество от натиснати бутони, подредени в нарастващ ред.

Тази функция се извиква веднъж на тест, когато асансьора достигне покрива.

Върнатият списък трябва да има дължина точно $N + 1$, където i-тият елемент трябва да е равен на стойността на v_f или на -1, ако Карлсон **не е успял** да възстанови стойността на v_f.

Важно: Хората в сградата не могат да комуникират помежду си по никакъв начин, освен чрез натискането на бутоните в асансьора. За да симулира това, програмата Ви ще се изпълнява като $N + 2$ отделни процеса, по един за всеки етаж и още един за покрива. Във

всеки процес за етаж ще бъдат изпълнени най-много T извиквания към функцията `press_buttons` и в процеса за покрива – точно T извиквания на функцията `answer`. Заради това програмата ви не трябва да предполага съществуването на обща информация (например глобални променливи) между извикванията на функциите. Всеки процес разполага с 64 MiB памет, която е отделна от паметта, заделена за останалите процеси и всички тези извиквания (най-много $(N + 1) \times T$ `press_button` и точно T `answer`) не трябва да отнемат повече от 10 секунди.

Ограничения

- $N = 60$
- $T \leq 10\,000$
- $0 \leq v_i \leq 3$ за всяко $0 \leq i \leq N$

Пример

Примерният тест има точно един тестов случай със следните стойности на етажите:

```
1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 1 1
1 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 0 1 1
```

Следващото е примерна комуникация:

Участник	**Жури**
	`press_buttons(0, 60, 0, 1, {})`
`return {2}`	
	`press_buttons(0, 60, 2, 0, {2})`
`return {13, 42}`	
	`press_buttons(0, 60, 13, 0, {2, 13, 42})`
`return {}`	
	`press_buttons(0, 60, 42, 1, {2, 13, 42})`
`return {}`	
	`answer(0, 60, {2, 13, 42})`
`return {1, -1, 0, -1, -1, ..., -1}`	

Описаната комуникация успешно възстановява стойностите на етажи 0 и 2. Всички останали стойности са -1, защото Карлсон не успява да ги възстанови.

Подзадачи

Подзадачи	Точки	Допълнителни ограничения	Броят възстановени стойности, необходими за пълен резултат
0	0	Пример.	–
1	15	$0 \le v_i \le 1$ $v_0 = v_N = 1$ $v_i = 1$ или $v_{i+1} = 1$ за всяко $0 \le i < N$	60
2	35	$0 \le v_i \le 1$	40
3	15	$0 \le v_i \le 2$	30
4	35	$0 \le v_i \le 3$	25

Точкуване

Ако програмата върне грешно възстановена стойност или извърши невалидна операция (примерно върне `vector` с грешна дължина), в който и да е тест от подзадача, ще получите 0 точки за тази подзадачи.

Дефинираме *възстановена бройка* за Вашето решение като броя позиции, на които стойността не е -1. Нека K да бъде минималната възстановена бройка сред всички тестови случаи за подзадача (всяка подзадача има точно един тест с най-много $10\,000$ тестови случая). Тогава точките за подзадачи са изчислени предвид долната таблица.

Подзадача	Стойност на K	Точки
0	–	0
1	$K < 60$	$0.25K$
	$K \geq 60$	15
2	$K < 30$	$0.5K$
	$30 \leq K < 40$	$2K - 45$
	$K \geq 40$	35
3	$K < 30$	$0.5K$
	$K \geq 30$	15
4	$K < 20$	$0.7K$
	$20 \leq K < 25$	$4K - 66$
	$K \geq 25$	35

Примерен грейдър

Примерният грейдър извършва всички извиквания на функции за всички тестове в един процес.

Входният формат е следният:

- ред 1: три цели числа – броя на тестовите случаи T, броя на етажите N и номера на подзадачата S;
- ред $1 + i$: $N + 1$ integers $v_0, v_1, \ldots, v_N$, където v_f е стойността, записана на етаж f за текущия тестов случай.

Изходният формат е следният:

- ред i: едно цяло число – броят коректно възстановени стойности на списъка v в тестов случай i;
- ред $T + 1$: финалните точки.

За по-подробна информация, променете стойността на макроса `DETAILED` от `false` на `true` на първия ред на грейдъра.

Може да позволите на грейдъра да генерира стойностите на v_f като промените стойността на макроса `AUTO_GENERATE` от `false` на `true` на втория ред на грейдъра.