НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
НАЦИОНАЛЕН КРЪГ 13-15 МАРТ 2026
Група С 7-8 клас

Анализ на задача Маршрути

Кратко условие. Дадена ни е таблица от свободни и блокирани клетки, както и два свързани региона (С и О) в два от краищата ѝ. Търси се максималния брой непресичащи се пътеки от единия до другия регион.

Първа подзадача. Намираме всички пътеки от (N,1) до (1,M) с брут форс. След това правим брут форс по това колко най-много от тях можем да вземем без да се пресичат. Отговорът е до 4 и брут форса няма да се държи прекалено бавно.

Сложност – **Yes**.

Втора подзадача. Допълнителното ограничение, че има най-много една пътека свежда задачата до доста стандартна постановка – търсим дали има някакъв път между два региона. За целта е достатъчно да пуснем едно DFS обхождане от единия и да проверим дали другия е достижим. Най-лесно е да считаме всичко освен X е проходима клетка и началната позиция да е (N,1). За отпечатване на решението трябва да внимаваме по пътеката да няма клетка С или О (евентуално да орежем някаква част от префикса и суфикса ѝ).

Сложност – **O(N×M)**.

Трета подзадача. Тук регионите са два квадрата и няма блокиращи клетки. Лесно можем да видим, че отговорът винаги е теоретичният максимум min(2L,2R), където L и R са размерите на квадратните региони. Самата конструкция на пътеките е симетрична – строим две пътеки по границите на таблицата, след това две пътеки по смалената таблица и така нататък докато по-малкия от двата региона не стане „запушен“.

Сложност – **O(N×M)**.

Четвърта и пета подзадача. Време е да направим основните наблюдения по задачата. Първо ще дефинираме и докажем някои концепции:

Дефиниция 0. Макар пътека да има доста разширена дефиниция в условието, тук ще подразбираме, че пътеката си има естествено обхождане от единия до другия край без разклонения.

Дефиниция 1. Област на дадена пътека между С и О наричаме клетките от таблицата, заградени от пътеката, регионите С и О, лявата стена и горната стена.

Дефиниция 2. Една пътека ще наричаме горна, ако в областта ѝ съществува единствен път между двата региона. Този път е именно горната пътека.

Пример: оригинална таблица (ляво) и горна пътека заедно с областта, която огражда (дясно):

```
..X......X..OO          ..X......X0000
..X.XXX..X..OO          ..X.XXX..X0.OO
X.....X..X....          X00000X..X0...
X.....XXXX....          X0...0XXXX0...
X..X........X.          X0.X.000000.X.
CC....XXXXXXX.          CC....XXXXXXX.
```

Твърдение 1. Съществува единствена горна пътека.

Доказателство. Да приемем, че има две различни горни пътеки T1 и T2. Ако те нямат обща зона, то едната ще се намира в областта на другата – противоречие. Ако имат обща зона, можем да приемем, че имат две общи зони S и T, а между тях нямат (все пак T1 и T2 се различават някъде). Между тези две зони, едната подпътека ще се намира в областа на другата – отново противоречие.

Твърдение 2. Съществува решение с максимален брой пътеки между C и O, в което горната пътека участва.

Доказателство. Нека съществува оптимално решение и вземем пътеката, която е в областта на всички останали (такава има). Сега нека я заменим с горната пътека. Тъй като тя ще се намира изцяло в областта ѝ, няма да се пресича с никоя друга и решението ще остане валидно.

Това ни е достатъчно, за да знаем, че следният алгоритъм ще намери оптимално решение:

1. Избираме горната пътека от C до O, ако такава съществува.
2. Блокираме клетките по намерената пътека.
3. Повтаряме 1, докато можем.

Въпросът е как намираме горна пътека? (Тук може да преминете директно на 6та подзадача, защото бързото решение е учудващо по-лесно).

НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
НАЦИОНАЛЕН КРЪГ 13-15 МАРТ 2026
Група С 7-8 клас

Оказва се, че е доста нетривиално, ако се опитаме да имплементираме дефиницията ѝ директно. Все пак ще покажем примерен алгоритъм, в който основната идея е да „повдигаме“ някоя пътека, докато тя не стане горна.

Нека започнем с произволна пътека T1 от C до O. Сега искаме да проверим дали можем да я „повдигнем“ или формално казано: ако T1 не е горна пътека, да намерим втора пътека в областта ѝ, различна от T1, която свързва C и O. По дефиниция, такава пътека съществува.

За целта можем да направим следното – маркираме досегашната пътека с 1-ци, а всички останали свободни клетки в областта ѝ с нули. Сега търсим най-краткия път от C до O чрез BFS 0-1 / Дейкстра. Ако дължината на най-краткия път съвпада с T1, то няма алтернативна пътека и спираме. Иначе повдигаме T1 до намерения най-кратък път.

Друг важен детайл е как следим, че не излизаме от областта на T1 при намирането на най-кратък път? Логично е да блокираме клетки, които се намират „под пътеката“ или формално, които не се намират в областта ѝ. За целта пускаме DFS от клетките по долната и дясната стена, което спира при клетка от текущата пътека. Това е почти вярно и можем да покажем частния случай с пример:

```
XX...X0        XX111X0        XX111X0
XX.....        XX1.111        XX1.111
XX.....        XX11...        XX11***
XXX....        XXX1...        XXX1***
C......        C111...        C111***
```

Единиците представляват намерена пътека (която е и горната). За следващата итерация на повдигане искаме да блокираме всички клетки извън областта ѝ. Нека ги бележим с *. Ако пуснем нормално DFS от долната и дясната стена, то ще намерите 9-те *. Проблемът е, че и има и още една, която обаче не е достижима по страна. Затова ще променим DFS-то да посещава и свободните клетки по връх. Дали това е достатъчно, за да блокираме всичко извън областта? Все пак е възможен следния случай:

```
1111.10
1..1.1.
1111.1.
11...1.
C11111.
```

С червено са отбелязани първите клетки от пътеката, а със синьо проходимите зони, които са извън областта. По-формално, случаят възниква, когато маршрутът, който сме намерили, **може да се скъси** (поразпишете малко примери и вижте, че е така). Тук и подобреното DFS няма да

открие затворените две зони, но се оказва, че подобен случай е невъзможен! Това не е никак очевидно и е следствие на специфичния начин на подбиране на пътека, отколкото на аргумент от вида, че изобщо не е възможно да намерим такава пътека.

Можем да покажем, че случая е невъзможен по индукция.

База: Първата намерена пътека ще е най-краткият път от С до О, следователно няма да може да се скъси.

Индукционна стъпка: Да приемем, че последно намерената пътека Т1 не е можела да се скъси и сега сме я повдигнали чрез BFS 0-1/Дейкстра до пътека Т2. Разглеждайки клетките от Т2, които не са част от Т1 отново можем да твърдим, че те не могат да се скъсят – следствие от BFS 0-1/Дейкстра. Прибавяйки общите клетки между Т1 и Т2 отново не можем да имаме скъсяване – защото при сменянето от Т1 на Т2 и обратно сме пропуснали поне една клетка от Т1.

С това имаме работещо решение, което в зависимост от имплементацията може да вземе само четвърта подзадача или и пета.

Сложност – **$O(K \times N^2 \times M^2)$**

Шеста подзадача. Трябва да оптимизираме по някакъв начин намирането на горна пътека. Една оптимистична идея е да напишем DFS, който просто върви по нея без да е нужно в последствие да „повдигаме“ пътеки и да усложняваме значително кода. Оказва се, че такова DFS наистина съществува.

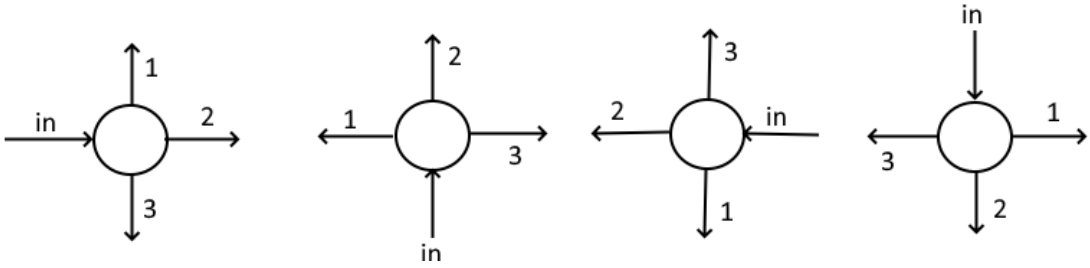
Да помислим кога едно нормално DFS може да сбърка. Една възможност е да продължи нагоре вместо да завие наляво, когато е имал възможност, или да продължи надясно, когато е можел да завие нагоре. Ето и примери за тези два случая:

```
..110      11110
..1..      1....
..1XX      1..XX
..1.X      1...X
..1.X      111.X
XX1.X      XX1.X
111.X      111.X
C....      C....
```

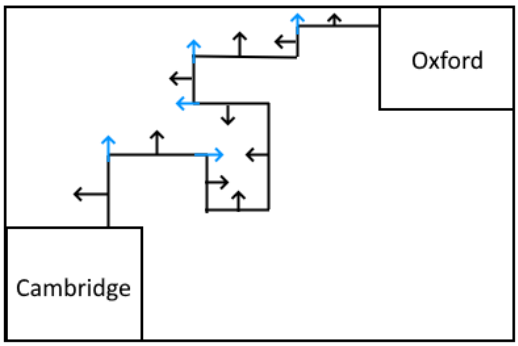
```
......0    ..11110
XX.XXX1    XX1XXX.
1111X.1    111.X..
C..1111    C......
```

Генерално, искаме DFS да завива към „горната-лява част“, когато е възможно. С малко разписване на случаи за приоритета на посоки ще получим следните стойности:

НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
НАЦИОНАЛЕН КРЪГ 13-15 МАРТ 2026
Група С 7-8 клас



Тук с in бележим посоката, от която DFS е достигнало текущата клетка, а с 1 2 3 реда, в който трябва да посетим съседите. Така 1 винаги сочи към областта, 2 продължава в същата посока а 3 е към противоположното на областта. Посоките винаги са по часовниковата стрелка.



На примера вдясно с черни стрелки е показана посока 1 по клетките на пътека, със сини, посока 2. Лесно може да видим и защо решението е коректно – ако съществува нейно подвигане, то ще включва отбиване в областта й или завиване по някоя черна стрелка. Тъй като DFS-то е посетило тях преди всичко останало и не е намерило пътека, знаем че подвигане не съществува.

Това доказателство е почти пълно като изключим началото на пътеката, защото теоретично и то може да бъде преместено в някои случаи. Това, което ще покажем в последната подзадача е, че е достатъчно да пускаме модифицираното DFS от всички клетки по границата, подредени отляво-надясно до ъгъла на квадрата и после отгоре-надолу.

Имплементирането на тази идея е доста по-лесно от предната подзадача – маркираме всяка следваща пътека като блокирани клетки. Така обхождаме цялата таблица веднъж с модифицираното DFS.

Сложност – **O(N × M)**.

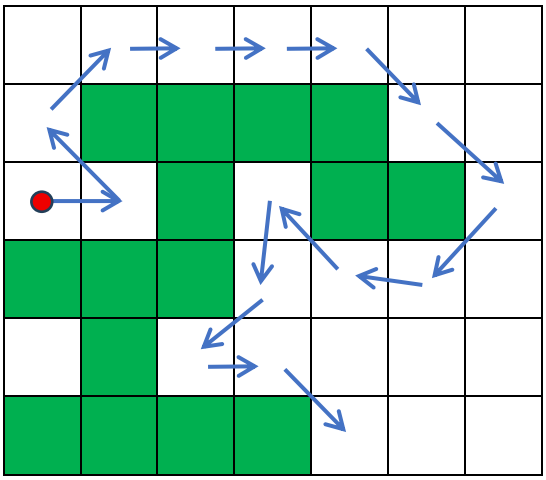
Седма подзадача. Остава да видим какво правим с региони, които нямат квадратна форма, защото при тях реда на стартови позиции за пътеки не е очевиден. Понеже регионите са долепени до съответните си ъгли (t[n][1]='C' и t[1][m]='O'), можем да подредим клетките по границата от единия до другия край последователно. По-интересното е как да ги намерим. По дефиниция те трябва да са свободни клетки, които имат клетка тип С съседна по стена. Ще използваме едно допълнително DFS, което отново гледа съседите си (този път и по връх) по часовниковата стрелка, започващо от първата свободна клетка в колона 1, която има клетка С отдолу.

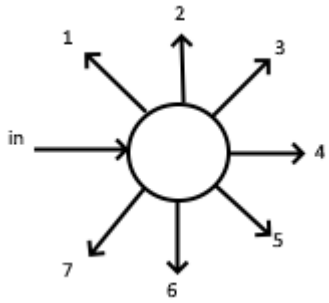
Сложност – **O(N×M)**.

НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА

НАЦИОНАЛЕН КРЪГ 13-15 МАРТ 2026

Група С 7-8 клас





Автор: Александър Гатев