# ЛЕТЕН ТУРНИР ПО ИНФОРМАТИКА
## Пловдив, 6 - 8 юни 2025 г.
## Група D - 6 клас

### Задача D?. РЕГИОНИ (Анализ на решението)

⏳0,5 сек. 💾256 MB

Задачата беше създадена с целта да има две различни решения, които да минават за 100 точки. И в двете решения ще използваме следните дефиниции:

- Ще казваме, че *манхатановото разстояние* (в анализа ще използваме само разстояние) между две клетки $(i_1, j_1)$ и $(i_2, j_2)$ е $|i_1 - i_2| + |j_1 - j_2|$;
- Ще казваме, че $k$-тият слой на подходящ регион е множеството от всички клетки, които са на разстояние точно $k$ от центъра.
- Ще казваме, че подходящ регион е с дължина $k$, ако максималното разстояние на клетка от региона до центъра му е $k$.

Така всъщност дефиницията за подходящ регион от условието се услеснява значително. Тя е еквивалентна на "всички клетки с разстояние не повече от $k$ от някоя фиксирана клетка (център) стига всичките такива клетки да са свободни.

### *Първа подзадача*

Решението тук е стандартен bruteforce. Фиксираме център на регион и слой по слой проверяваме дали има заета клетка. Важно е да се отбележи, че при този подход разчитаме на факта, че ако в даден момент проверяваме $k$-тия слой, то вече сме проверили, че до $k-1$-вия всички клетки са свободни, а не ги проверяваме наново.

Постигната сложност: $O(NM \times \min(N, M)^2)$

Имплементация: `regions_kiril_9p.cpp`

### *Втора подзадача*

Достатъчно е да правим същото като в първа подзадача, но без проверката за свободни клетки.

Постигната сложност: $O(NM \times \min(N, M))$

Имплементация: `regions_kiril_7p.cpp`

### *Трета подзадача*

Достатъчно е да правим същото като в първа подзадача, но проверката за свободни префикси да става с префикси. По-точно, понеже $k$-тият слой на подходящ регион може да се раздели на 4 части, всяка от която представлява последователност от клетки, образуваща "диагонал трябва да пазим диагонални префикси.

Постигната сложност: $O(NM \times \min(N, M))$

Имплементация: `regions_kiril_32p.cpp`

### *Четвърта подзадача*

Всъщност цикълът по дължината на подходящия регион е напълно ненужен, защото можем директно да изчислим колко са. По-точно, ако сме фиксирали като център клетката $(i, j)$, то броят подходящи региони, които могат да се изберат с център тази клетка е $\min(i, j, N+1-i, M+1-j)$.

Постигната сложност: $O(NM)$

Имплементация: `regions_kiril_20p.cpp`

## *Пета подзадача*

Ще направим твърдението ни от четвърта подзадача по-силно. Да си припомним, че за да съществува подходящ регион с център дадена клетка и дължина $k$, трябва:

- регионът да не излиза от таблицата;
- да не съществува заета клетка, която е на разстояние по-малко от $k$ от центъра.

Според условието има само една заета клетка - нека тя да е $(x, y)$. Нека фиксираме център на региона $(i, j)$, както е дължина $k$. Както видяхме в четвърта подзадача, първото условие е еквивалентно на $k \leq \min(i-1, j-1, N-i, M-j)$. Според дефиницията на разстояние второто условие е еквивалентно на $k < |i-x| + |j-y|$. Чрез тези две условия можем да съобразим, че броят на възможните дължини е равен на $\min(i, j, N+1-i, M+1-j, |i-x| + |j-y|)$.

Постигната сложност: $O(NM)$

Имплементация: `regions_kiril_37p.cpp`

## *Междинни разсъждения*

Ако условието гарантира, че съществуват $S$ заети клетки, то може да се разшири тази формула, като формулата ще приеме вида $\min(i, j, N+1-i, M+1-j, T)$, като

$$T = \min(|i-i_1| + |j-j_1|, |i-i_2| + |j-j_2|, \ldots, |i-i_S| + |j-j_S|) = \min_{p=1}^{S} |i-i_p| + |j-j_p|,$$

където $(i_p, j_p)$ е $p$-тата свободна клетка. Това е сравнително лесно за писане и осигурява 46 точки.

Постигната сложност: $O(NM \times S)$

Имплементация: `regions_kiril_46p.cpp`

## *Шеста и седма подзадача*

Единствената разлика между шеста и седма задача е качеството на написания код. Предвидено е кодът да не минава за седма задача, ако не е написан сравнително оптимално.

Ще представим два варианта за довършване на задачата.

*Първо решение:* Задача, при която се преброяват региони, като смяната между $k$-тото ниво и $k+1$-вото е на практика безплатно (ако е направено с префиксни суми), плаче за показалки. Нека сме намерили максималната дължина $k$ на подходящ регион с център $(i, j)$. Тогава е гарантирано, че максималната дължина на подходящ регион с център $(i, j+1)$ е поне $k-1$. Това се дължи на факта, че този регион се съдържа изцяло в намерения подходящ. Така при всяка стъпка извършваме най-много една проверка, за да се върнем до миналата стойност на $k$. Ако пренебрегнем изваждането на единица поради причината, която току що обяснихме, забелязваме, че редицата е растяща и с горна граница $\min(N, M)$. Така общо за един ред не можем да направим повече от $2M$ операции.

Постигната сложност: $O(NM)$

Имплементация: `author.cpp`, `regions_kiril_100p_2pointers.cpp`

*Второ решение:* В междинните разсъждения сведохме задачата до оптималното намиране на $T$. То е дефинирано като минималното разстояние от клетката до някоя заета клетка. Познаваме ли алгоритъм, който може да постигне това за "линейна"сложност? Към края на D група

се очаква, че състезателите знаят (или поне имат представа) какво е BFS. Той обаче обикновено намира разстоянието от една клетка до всички останали. Можем да се справим с това, като го пуснем от всички заети клетки едновременно. Това работи, защото типичното BFS пази всички клетки от един слой едновременно, така че той всъщност ни позволява да кажем, че всичките заети клетки са на едно и също ниво (за удобство и целите на задачата това ниво е 0). Ако е трудно за разбиране, може да си представите как пускате BFS от дадена невидима клетка, след което след време достига до заетите клетки (като са на едно и също ниво) и чак тогава започва да маркира всички останали клетки в таблицата. За повече детайли вижте имплементацията.

Постигната сложност: $O(NM)$

Имплементация: `regions_kiril_100p_bfs.cpp`

*Автор: Кирил Зулямски*