АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ТРАФИК

За решаване на задачата използваме алгоритъма на Дейкстра за намиране на най-кратък път, като реализацията е осъществена посредством приоритетна опашка. Теглата са времената за преминаване през дадена клетка.
Дефинираме двумерен масив `int dist` в който съхраняваме времената за преминаване. От съответната буква изваждаме ‘A’. Символът ‘*’ заменяме с -1. В променливата start запазваме координатите на стартовата клетка.
Дефинираме структурата cell:

```
struct cell {
                int row, col, d;
                bool operator < (const cell &toll) const {
return d > toll.d;}
};
```

За всяка клетка от мрежата пазим координати (ред, колона) и време за преминаване през клетката. Дефинираме приоритетна опашка, елементите на която са елементи на структурата cell. В структурата дефинираме оператор „<”, тъй като на върха на пирамидата искаме да се намира най-малкия елемент (по премълчаване е най големия).

```
priority_queue<cell> pq;
```

Двумерният масив `inspected` използваме за маркиране на обходените върхове.
Използваме масивите `dr` и `dc` за движение по редове/колони на матрицата.
Основният алгоритъм:

- Намираме стартовата клетка (‘#’);
- Проверка за край;
- Проверка дали Добавяме я като елемент на структурата cell с даден приоритет в опашката;
- Цикъл до изчерпване на елементите на приоритетната опашка:
- клетката вече не е проверена;
- Намиране на следваща клетка. Пропускат се клетките в които времената за преминаване са отрицателни.
- Добавяне като нов обект с даден приоритет в опашката;
- Премахваме на текущия обект от опашката.

Автор: Пано Панов