Анализ на Задача 1 от Конкурс по програмиране на Musala Soft и PC Magazine

Автор: Антон Димитров

Едно лесно решение на поставената задача би било да се разгледат всички възможни пътища, които започват от началната позиция и стигат до край на дъската. От тях можем да изберем това, което е най-оптимално. За съжаление, такова решение работи за поставеното времево ограничение само при малки тестове.

В моето решение съм избрал да генерирам някакъв брой пътеки като е избран критерий за това кои са „интересни" и трябва да се разгледат. Важен е и начинът, по който се генерират тези пътеки. В случая се използва алгоритъм, подобен на обхождане в ширина. На първа стъпка се генерират пътеки с дължина само един символ, след това от тях се генерират тези с дължина два символа и т.н. Разбира се в един момент пътеките стават прекалено много. Ако приемем, че всяка пътека може да породи други 4 с дължина с един символ повече, то следва че пътеките с дължина N са 4^{N-1}. За достатъчно големи N това е прекалено много, за да се побере в паметта, а и да бъде пресметнато бързо. Ясно е че за някои тестове дори най-оптималния отговор ще е прекалено голям, за да работи добре такова решение.

Поради тези причини е нужно на всяка стъпка по генериране на нови пътеки да се избират само най-обещаващите, с които да се продължи нататък. Ако критерият е достатъчно добър, това ще позволи да се избират малко на брой пътеки и цялото изпълнение на програмата да се побере в зададените ограничения по памет и време. В моето решение създадените във всеки един момент пътеки всъщност се разглеждат като образци, чиято дължина всъщност търсим да оптимизираме. Това означава, че ако на дадена стъпка са генерирани пътеки с дължина N, те всъщност са потенциални образци, с които започват пътеки и тези образци трябва да се повтарят до излизане от дъската. Критерият, който е използван, се базира на това, че „обещаващи" са тези образци, които имат най-много свои копия из дъската. Ако например един образец, които започва от началната клетка е „abcdef" и някъде из дъската има още 50 такива образеца на различни позиции, това дава оценка 50 на този образец. Причината да смятам това за полезна оценка е че колкото повече е наситена дъската с такива образци, толкова по-голям е шансът те да се навържат в една непрекъсната пътека, която стига до край на дъската.

Използвайки този критерий, ако на всяка стъпка се генерират пътеки и в последствие се ограничават до първите X с най-висока оценка се гарантира, че програмата ще може да достигне до достатъчно дълги образци без да загуби прекалено много време или памет за целта. Разбира се съществува проблемът с това, че най-добрите решения може да са изгредени от образци, които получават ниска оценка и те да не бъдат разгледани. За да се намали този негативен ефект, само част от избраните на всяка стъпка образци са тези с най-висока оценка. Останалата част са образци, които са избрани произволно, за да се повиши разнообразието.

Важна стойност в този алгоритъм е колко пътеки се оставят след всяка нова генерация на пътеки. Когато тази стойност е малка, алгоритъмът ще е бърз и ще достига до големи дължини на образците, които разглежда. Ако стойността е по-

голяма, алгоритъмът ще се бави повече, но ще разглежда повече възможни образци. Ето защо, решението се стартира два пъти, веднъж с по-малка константа и веднъж с по-голяма, като се взема най-доброто намерено решение.

Както става ясно, това решение не гарантира намиране на решение. Ето защо в самото начало се използва обхождане в ширина, за да се намерят решения, които достигат до край на дъска. От тях се избира това, което има най-къс образец.

Също така, когато се генерират образци се разглеждат тези, които имат поне още едно срещане в дъската, което достига до край на дъската. Това се прави, защото е възможно ако се свържат по произволен начин двата образеца да образуват пътека, която в началото си започва с някакъв образец, след това съдържа произволна последователност от символи и накрая завършва с някаква част от образеца. За да стане по-ясна идеята ще използвам един пример: нека е намерен образец „abcdef", който има начало в стартовата клетка на дъската. Също така е намерен образец „acbd", който завършва в край на дъската. Следователно ако края на първия образец – буквата 'f' - се свърже чрез произволен път до началото на другия образец – буквата 'a' – ще се получи път, който има дължина 6 + M + 4, като M е дължината на свързващия път. Този път ще има образец с дължина 6 + M. В някои произволно генерирани тестове, това може да доведе до малко по-добри решения от намерените от едно обхождане с ширина.