Задачи:

1. Витез(Knight Tour problem)

Backtracking може наједноставно да го објасниме преку примерот Knight Tour problem, односно станува збор за пример од играта Шах, но на таблата се наоѓа само една фигура, а тоа е витезот кај кој се дефинирани истите движења како кај фигурата Крал. Дозволени се 8 движења во зависност од позицијата, движење во насока на едно поле. Главна цел на играта е да самиот витез ги посети сите полиња само еднаш.

Псевдокод за самата игра:

while there are untried tours

{

generate the next tour

if this tour covers all squares

{

print this path;

}

}

Псевдокод кој се користи кај Backtracking алгоритамот:

If all squares are visited

print the solution

Else

a) Add one of the next moves to solution vector and recursively

check if this move leads to a solution. (A Knight can make maximum

eight moves. We choose one of the 8 moves in this step).

b) If the move chosen in the above step doesn't lead to a solution

then remove this move from the solution vector and try other

alternative moves.

c) If none of the alternatives work then return false (Returning false

will remove the previously added item in recursion and if false is

returned by the initial call of recursion then "no solution exists" )

Приказ на решението во програмскиот јазик Java:

1. Стаорец во лавиринт(Rat in a Maze)

Лавиринтот е составен од бинарна матрица М\*М, каде почетното поле е полето [0][0], а целта или крајното поле е последното или полето [М-1][М-1]. Стаорецот започнува од првото поле и мора да стигни до последното или дестинацијата. Тој може да се движи по полињата само право(десно) и долу. Дефинирано е правилото: ако полето е обележано со 1 тоа е дел од дозволеното движење, но ако полето е обележано со 0 тоа поле е недостапно. Следниот пример е еден од поедноставните примери од оваа загатка бидејќи постајат и многу посложени загатки од истиот тип.

Во нашиот случај разгледуваме матрица 4х4:



Исто со:

{1, 0, 0, 0}

{1, 1, 0, 1}

{0, 1, 0, 0}

{1, 1, 1, 1}

Решение:



Пристап за решение: Креираме една рекурзивна функција која го следи одбраниот пат и проверува дали тој пат е соодветен за да стигнеме до самиот крај. Ако одбраниот пат не не води до крајот или дестинацијата тогаш повторно се повикува функцијата.

Алгоритам:

* Креираме матрица исполнета со нули.
* Креираме рекурзивна функција која ги зема вредностите на почетната матрица, матрицата со додадени единици и позицијата на стаорецот.
* Ако позицијата не е легитимна тогаш печати грешка.
* Креирај индикатор за крајното поле, дека станува збор за крај на лавиринтот.
* Рекурзивно повикувај да позицијата на стаорецот се менува -> (i+1,j) и (i,j+1).

Преглед на решението во програмскиот јазик Java:

1. Судоку (SUDOKU)

На сите ние позната играта Судоку. Но за решавање на една таква загатка наједноставен начин, односно со помош на компјутерската технологија е користење на Backtracking алгоритам. Дадена е дводимензионална матрица 9х9, целта е да се пополни подматрицата (3х3) со броевите од 1 до 9 така што ни еден број не се повторува хоризонтално и вертикално и во самата подматрица. Како би изгледал псевдокодот:

Find row, col of an unassigned cell

If there is none, return true

For digits from 1 to 9

a) If there is no conflict for digit at row, col

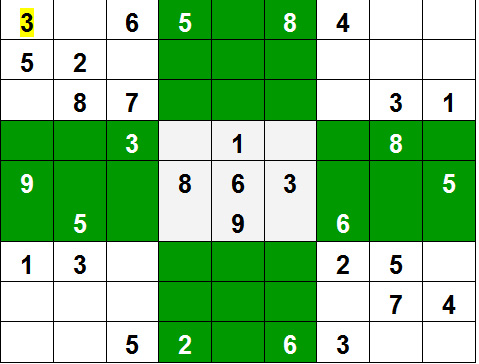
assign digit to row, col and recursively try fill in rest of grid

b) If recursion successful, return true

c) Else, remove digit and try another

If all digits have been tried and nothing worked, return false

Пример за Судоку матрица:



Преглед на решението во програмскиот кој Java: