Grundlagen der Wissensverarbeitung
Computer Science
Universität Hamburg

Übung 1: Search Space Design

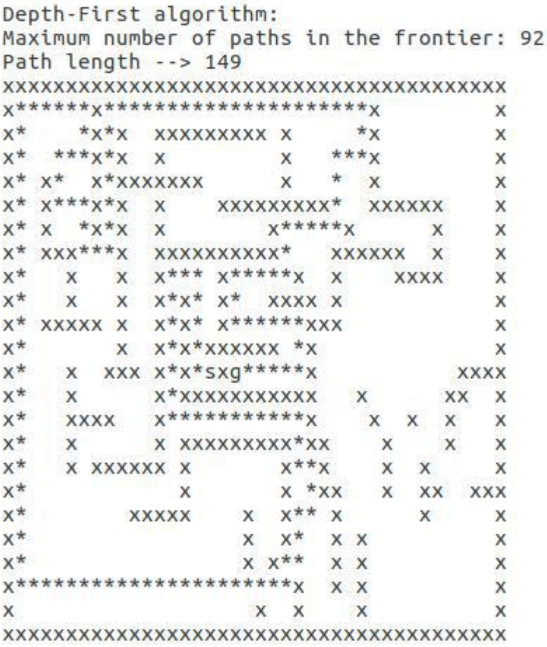
Rafael Ruz Gómez
Miguel Robles Urquiza

5 November 2017

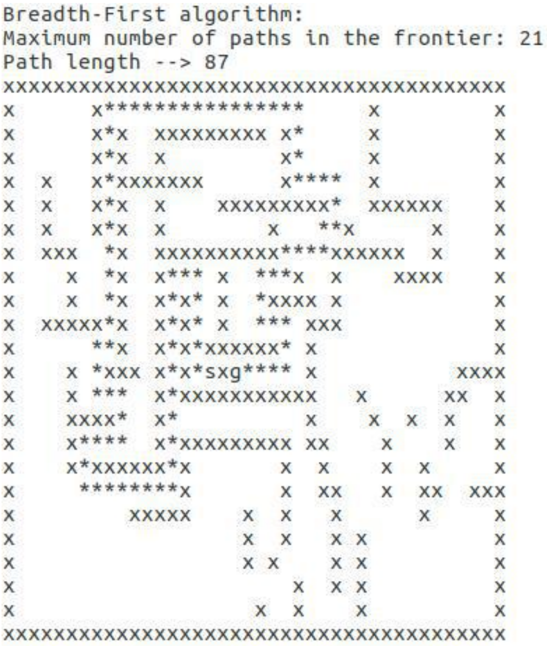
Exercise 4.3

Provide example mazes that show the differences between and properties of the search strategies. Describe these properties

Breadth first is useful when you want a solution containing the fewest arcs. It's obvious that bread first search have few arcs than depth first search.



```
Depth-First algorithm:
Maximum number of paths in the frontier: 92
Path length --> 149
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x******x*********************x         x
x*    *x*x  xxxxxxxxx x     *x         x
x*  ***x*x  x         x   ***x         x
x* x*  x*xxxxxxx      x   *  x         x
x* x***x*x  x    xxxxxxxxx*  xxxxxx    x
x* x  *x*x  x        x*****x      x    x
x* xxx***x  xxxxxxxxxx*   xxxxxx  x    x
x*   x   x  x*** x*****x  x    xxxx    x
x*   x   x  x*x* x*  xxxx x            x
x* xxxxx x  x*x* x******xxx            x
x*       x  x*x*xxxxxx *x              x
x*   x  xxx x*x*sxg*****x           xxxx
x*   x      x*xxxxxxxxxxx   x      xx  x
x*   xxxx   x***********x    x  x  x   x
x*   x      x xxxxxxxxx*xx    x    x   x
x*   x xxxxxx x       x**x    x  x     x
x*            x       x *xx   x  xx  xxx
x*        xxxxx    x  x** x      x     x
x*                 x  x*  x x          x
x*                 x x**  x x          x
x**********************x  x x          x
x                   x  x    x          x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

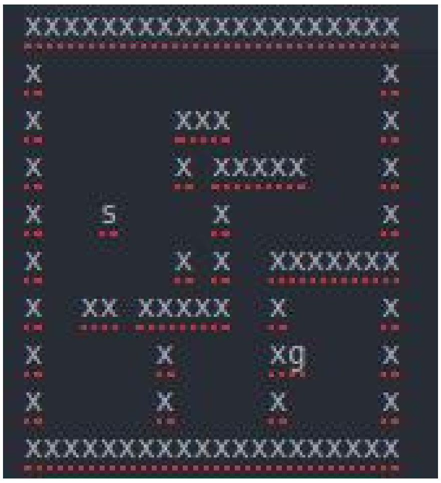


```
Breadth-First algorithm:
Maximum number of paths in the frontier: 21
Path length --> 87
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x      x****************     x         x
x      x*x  xxxxxxxxx x*     x         x
x      x*x  x         x*     x         x
x  x   x*xxxxxxx      x****  x         x
x  x   x*x  x    xxxxxxxxx*  xxxxxx    x
x  x   x*x  x        x   **x      x    x
x  xxx  *x  xxxxxxxxxx****xxxxxx  x    x
x    x  *x  x*** x  ***x  x    xxxx    x
x    x  *x  x*x* x  *xxxx x            x
x  xxxxx*x  x*x* x  *** xxx            x
x      **x  x*x*xxxxxx* x              x
x    x *xxx x*x*sxg**** x           xxxx
x    x ***  x*xxxxxxxxxxx   x      xx  x
x    xxxx*  x*          x    x  x  x   x
x    x****  x*xxxxxxxxx xx    x    x   x
x    x*xxxxxx*x       x  x    x  x     x
x     ********x       x  xx   x  xx  xxx
x         xxxxx    x  x   x      x     x
x                  x  x   x x          x
x                  x x    x x          x
x                      x  x x          x
x                   x  x    x          x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Exercise 4.4

Are there cases in which your program is unable to find a solution? Provide example mazes.

We have found just one case in wich our program is unable to find a solution, when there is no real solution. We meant, when we can not arrive to the goal because the goal is totally surrounded by 'x'.



```
xxxxxxxxxxxxxxxxxxxx
x                  x
x       xxx        x
x       x xxxxx    x
x   s     x        x
x       x x  xxxxxxx
x  xx xxxxx  x     x
x      x     xg    x
x      x     x     x
xxxxxxxxxxxxxxxxxxxx
```

Exercise 5.2

Write a second heuristic function that works correctly with portals. What do you have to change?

We check which distance is the minimum distance between our actual position and each portal. If taking this portal we arrive to the goal with minimum cost, we choose this path.

Exercise 5.3

The maze above is a slightly modified version of the environment (Not possible path between start and goal). How does your search react in this case? Can you ensure termination?

We send a message to the user telling that the program has been unable to find a path between the start node the goal node.
Yes, we can ensure termination