Advanced Algorithms – Assignment 2

Lucas Geurtjens (s5132841)

# Q1. Tracking K-Smallest Numbers

**Introduction:**

A data structure was created to store a very large number of random numbers and a separate list of k-smallest numbers in O (N + K) space complexity. This k-smallest list was maintained using a Fibonacci heap.

**How it works:**

Fibonacci Heap

Insert

Remove

Deleting

**Input/Output:**

…

**Amortised Analysis:**

…

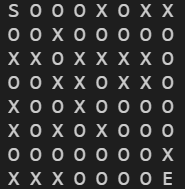
# Q2. Maze Generation

**Introduction:**

A maze generation algorithm was created using the technique of randomly knocking down walls until a path from the start and end is found.

**How it works:**

Here’s an example of an 8x8 maze:



LEGEND:

**S**: Start of maze

**E**: End of maze

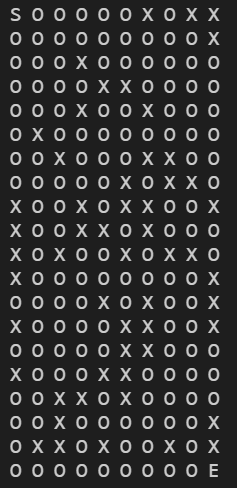
**X**: Wall (cannot pass)

**O**: Path (can pass)

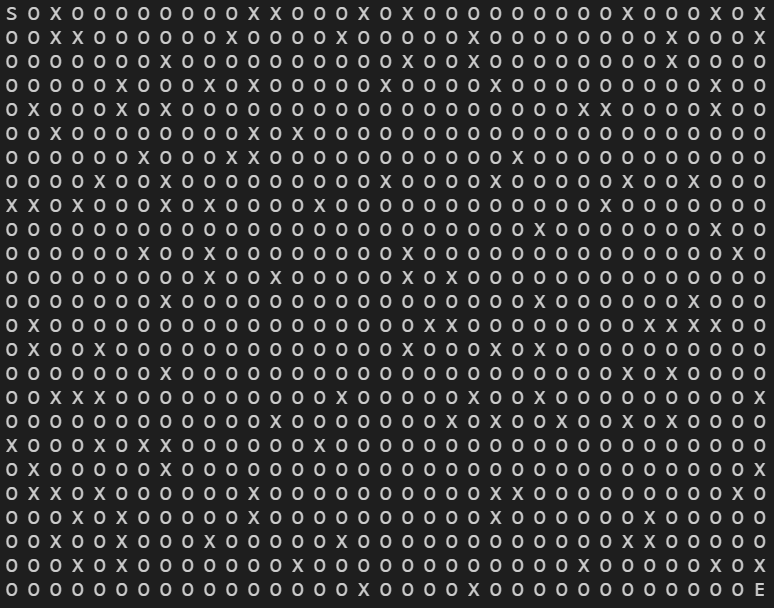
* The maze grid is initialised (filled with walls)
* While there is no complete path from the Start to the End of the maze:
  + Check if the maze is complete (if so, we can end)
    - Run a Breadth First Search from the Start -> End
    - Run a Breadth First Search from the End -> Start
      * During the BFS, calculate the furthest we can get to the goal state (done using Manhattan distance). When the BFS finishes (assuming it didn’t find the goal state), this value will help determine the minimum number of walls we must break down.
    - Calculate the number of walls to break down (the Manhattan distance from the furthest point from the start, to the furthest point from the end).
    - Breakdown the number of walls specified by randomly selecting points. If the point is a wall, break it down, and add this to the total amount of walls broken.

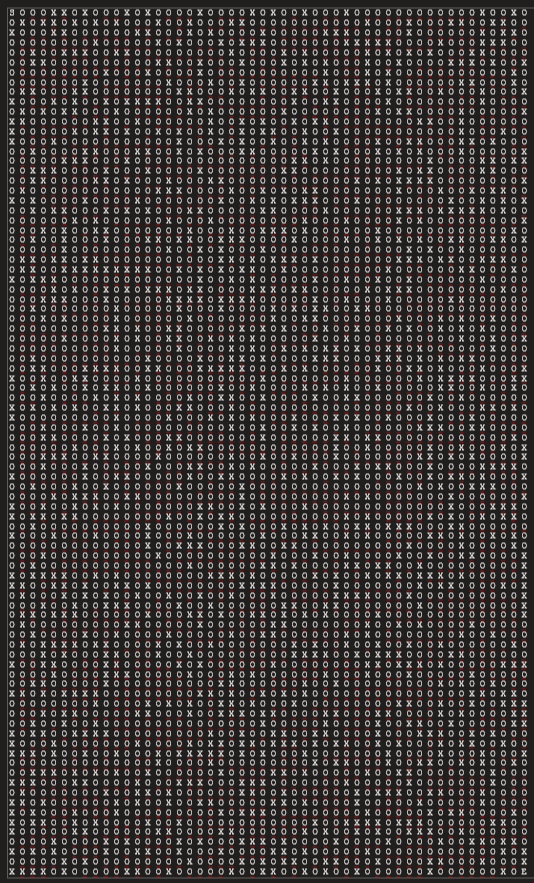
**Input/Output:**

10x20 Maze

****

35x25 Maze

****

50 x 88 Maze: 

# Q3. Red Black Trees vs Van Emde Boas Trees

**Introduction:**

…

**How it works:**

…

**Input/Output:**

…

**Performance Analysis:**

…

# Q4. The Kevin Bacon Game

**Introduction:**

The Kevin Bacon Game tries to find the minimum number of links between two actors. The program has three main methods:

1. Find Minimum Links: Finds the minimum number of links between two actors.
2. Find Bacon Number: Finds the minimum number of links between an actor and Kevin Bacon.
3. Find Highest Bacon: Finds the actor with the greatest number of links to Kevin Bacon.

**How it works:**

*Finding links between actors (all methods use this as their key function).*

* Data is input and compiled into more useful data structures (a list containing actors, and a list containing movie productions).
  + Specified actors are input.
  + A Bacon Score is kept to track how many levels down the BFS goes.
  + A Breadth First Search is performed, adding the starting actor to a queue.
    - The queue is popped and we check the popped actor’s movies.
      * If the movie hasn’t been visited, we check the actors in the specific movie.
        + If the actor is the end actor, we return the bacon score.
        + Otherwise, if the actor hasn’t been visited, we add it to the queue.

**Input/Output:**

NOTE:I only used data from bacon1.txt and some of bacon2.txt. Adding too much data caused a bug I wasn’t able to solve. Thus, since we didn’t have access to ALL movies/actors, some number of the links may be slightly inaccurate.

|  |  |
| --- | --- |
| **Method** | **Output** |
| findBaconNumber ("Steven Brill (I)") | 3 links |
| findBaconNumber ("Carrie Fisher") | 2 links |
| findBaconNumber ("Jorn Benzon ") | 4 links |
| findBaconNumber ("Billy Crystal") | 2 links |
| findMinLinks ("Denise Dabrowski", "Roy C. Johnson") | 1 link |
| findMinLinks ("Roman Bohnen", "Alan Rickman") | 4 links |
| findMinLinks ("Albert Brooks (I)", "Steve Buscemi") | 6 links |
| findMinLinks ("Gino Corrado", "Tim Condren") | 3 links |
| findHighestBacon () | 6 links (Toshiyuki Amagasa) |

# Q5. Minimum Vertex Covers for Complement Graphs

**Introduction:**

…

**How it works:**

…

**Input/Output:**

…