Backtracking

CS 3AC3

Ryszard Janicki

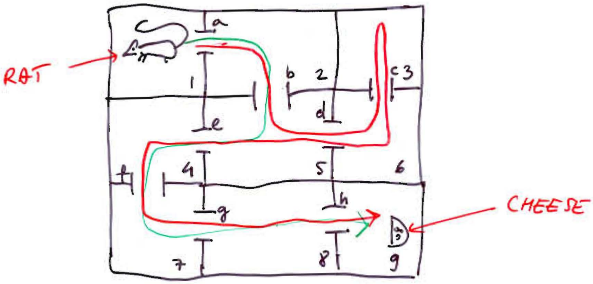
Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada

This material is not covered by the textbook

Backtracking

- Sometimes we are faced with the task of finding an optimal solution to a problem, yet there appears to be no applicable theory to help us find the optimum, except by resorting to **exhaustive** search.
- But we want to check each case no more than once!

Rat in Maze



Rooms: 1, 2, 3, 4, 5, 6, 7, 8, 9

Doors: *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*

Doors: 1-2, 2-5, 5-6, 3-6, 4-7, 7-8, 8-9

Rat Procedures and Data Structures

WhereIsDoor:

1. Look North, if there is unused door, use it, otherwise goto 2.
2. Look East, if there is unused door, use it, otherwise goto 3.
3. Look South, if there is unused door, use it, otherwise goto 4.
4. Look West, if there is unused door, use it, otherwise goto 5.
5. Unused door does not exist, go back through the door you entered.

IHaveBeenThere: Mark the room you have entered.

IHaveUsedThisDoor: Mark the door you have used.

- Since doors are between rooms, it suffices to mark:

DoorBetweenRooms

Rat Procedures and Data Structures

MyImportantPath:

$$sequence: \ room_1, door_1, room_2, \ldots, room_k, door_k, room_{k+1}$$

and the rat has been in each room *exactly once* except the $room_{k+1}$, where it might be for the second time, and he used each door *exactly once*, except $door_k$.

- MyImportantPath is a **stack**.

WasIThere?: Returns YES if the room is entered for the second time.

Rat Algorithm

RatAlgorithm:

1. WhereIsDoor;
2. If WasIThere? = YES, go back through the door you entered and modify MyImportantPath by popping stack **twice**, and goto 1.
3. Otherwise, modify IHaveBeenThere, IhaveUsedThisDoor and MyImportantPath (by pushing $door_{used}$ and $room_{current}$), and goto 1.

Proposition

Rat algorithm has time complexity O(size of maze) and space complexity O(size of maze) too.