CS440 MP1 Report

Andong Jing ajing2 - 3 credit

Siping Meng smeng10 - 3 credit

Siyu Tao siyutao2 - 3 credit

# Basic pathfinding

## BFS

### Medium maze

Solution cost (# of steps): 104 # of expanded nodes: 634

### Big maze

Solution cost (# of steps): 156 # of expanded nodes: 1261

### Open maze

Solution cost (# of steps): 53 # of expanded nodes: 573

## DFS

### Medium maze

Solution cost (# of steps): ??? # of expanded nodes: ???

### Big maze

Solution cost (# of steps): ??? # of expanded nodes: ???

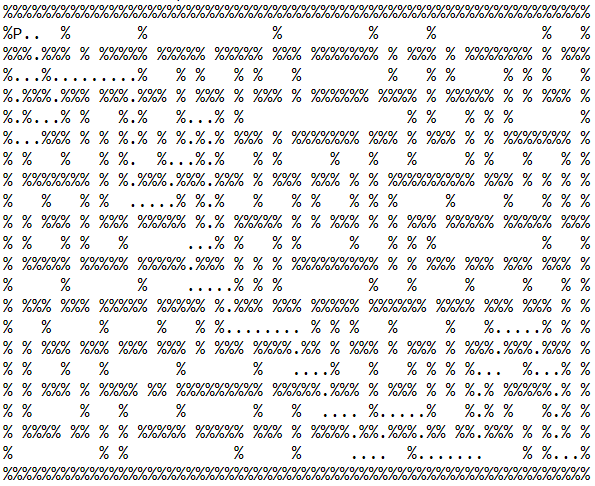
### Open maze

Solution cost (# of steps): ??? # of expanded nodes: ???

## Greedy

### Medium maze

Solution cost (# of steps): **116** # of expanded nodes: **143**



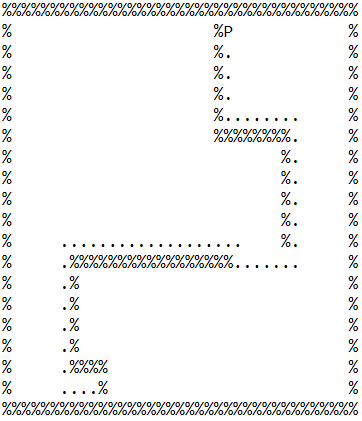
### Big maze

Solution cost (# of steps): **234** # of expanded nodes: **293**



### Open maze

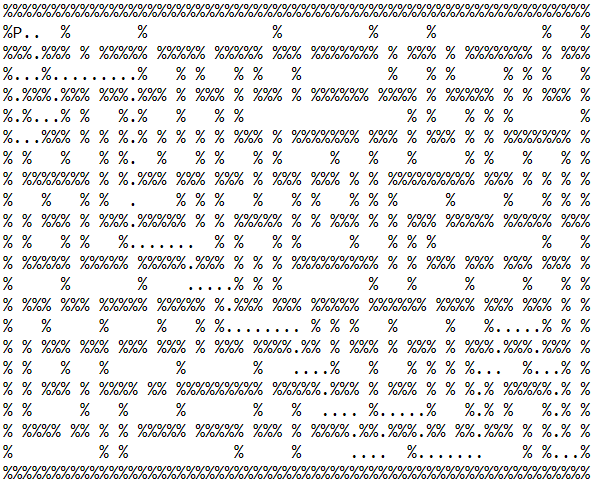
Solution cost (# of steps): **53** # of expanded nodes: **198**



## A\*

### Medium maze

Solution cost (# of steps): **104** # of expanded nodes: **513**



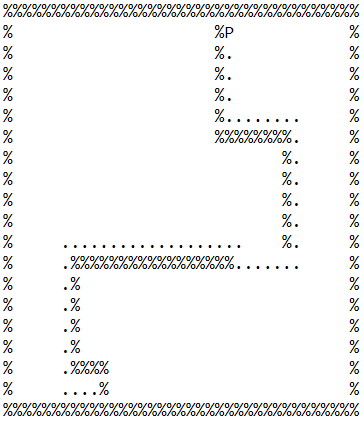
### Big maze

Solution cost (# of steps): **178** # of expanded nodes: **1116**



### Open maze

Solution cost (# of steps): **53** # of expanded nodes: **474**



# 1.2 Search with multiple dots

Discuss your heuristic, including its admissibility.

## Tiny search

Solution cost (# of steps): ??? # of expanded nodes: ???

## Small search

Solution cost (# of steps): ??? # of expanded nodes: ???

## Medium search

Solution cost (# of steps): ??? # of expanded nodes: ???

# 1.2 Extra Credit

We used a combination of greedy algorithm and A\* algorithm to guide our search. The algorithm will first find the nearest dot and mark it the current goal. Then, a\* algorithm used in 1.1 will find the optimal route to that nearest dot. The process is repeated until no dots are left in the maze. According to our output, the solution cost is **353 steps** and **883 nodes** are expanded. While choosing the nearest dot as the goal does not guarantee optimality, we compensate the algorithm by incorporating a\* search, which makes sure the optimality to each “current goal”.

# 2. Sokoban

Soko3: 410.6101870536804