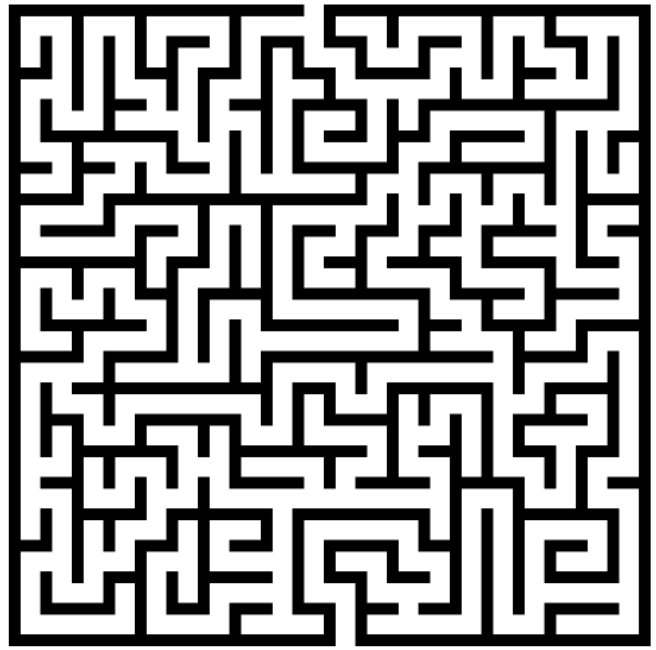
Maze Solver

20194436 Nguyen Duy Hung

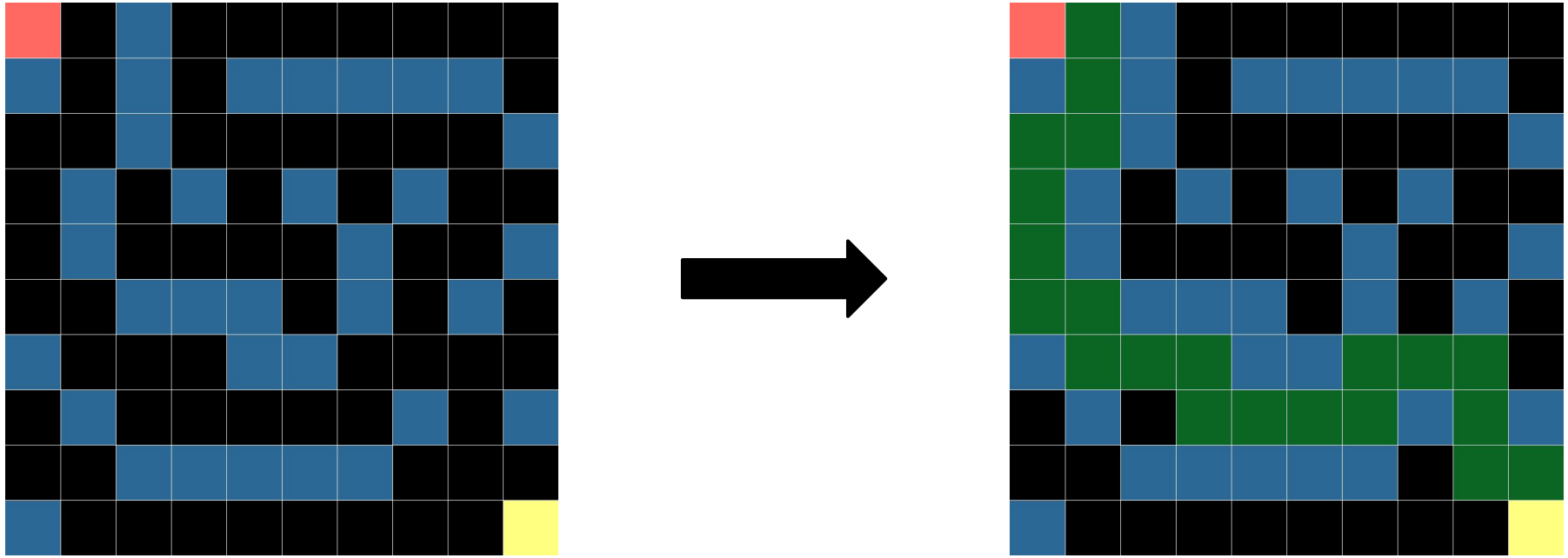
20194448 Nguyen Hoang Nhat Quang

20194449 Le Hai Son



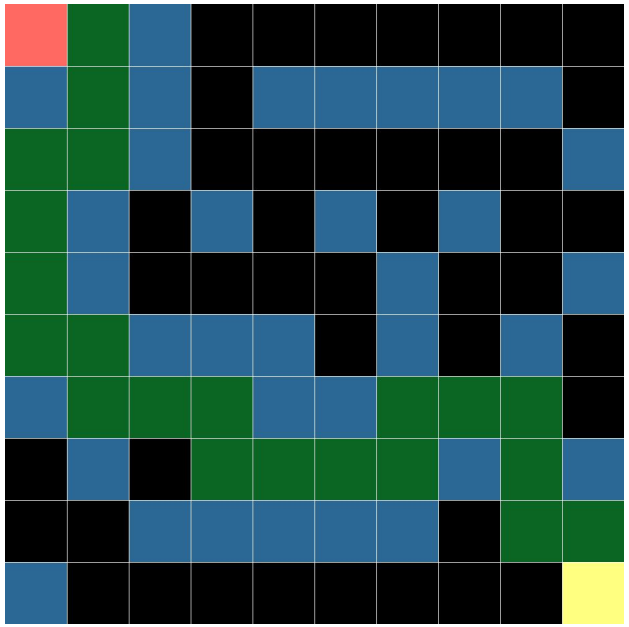
Maze Solver

a program to automatically generate, and solve a maze



Description

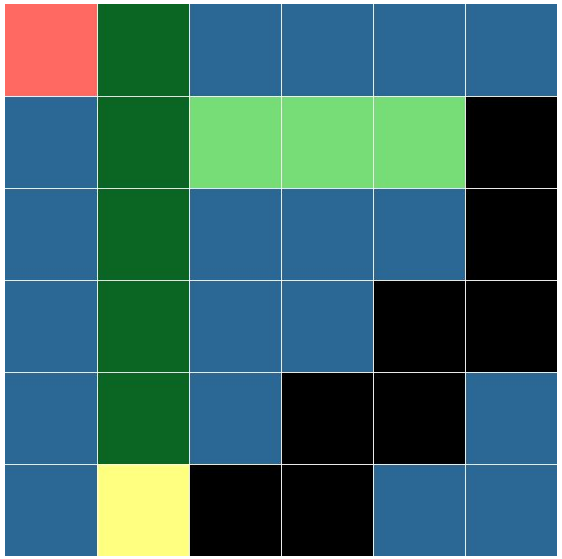
- Solve the problem by searching methods
- Formulation:
  - state: (x, y)
  - actions: ‘up’, ‘down’, ‘left’, ‘right’
  - transition model: e.g. *result*(‘up’, (x, y)) = (x - 1, y)
  - goal: reach the final state
  - cost: 1 per action executed



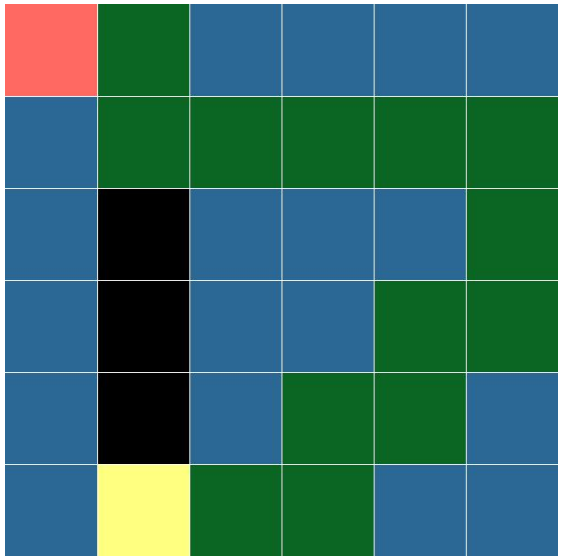
Uninformed search

search strategy that uses no problem specific knowledge

breadth-first search: expands shallowest node in the frontier
depth-first search: expands deepest node in the frontier



breadth-first search



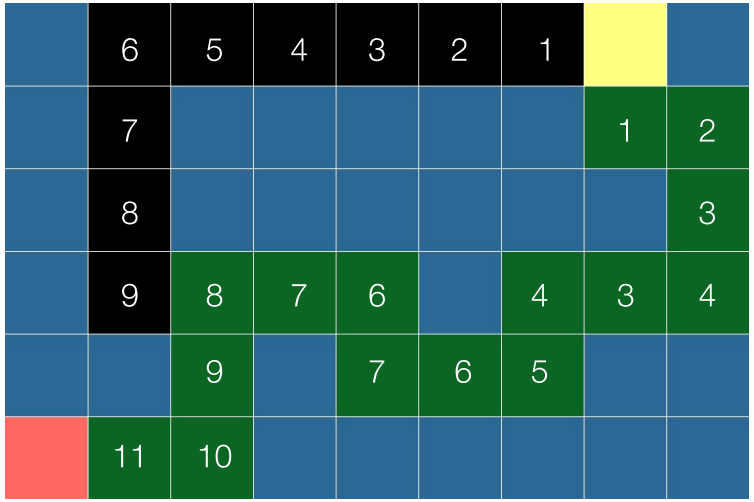
depth-first search

Informed search

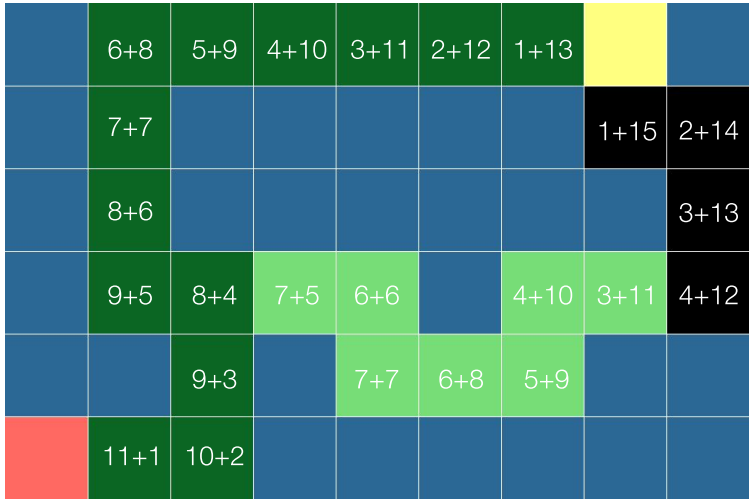
search strategy that uses problem-specific knowledge
to find solutions more efficiently

- h(n) = estimated cost to goal
- c(n) = cost to reach node

greedy best-first search: expands the node that is closest to the goal using h(n)
A* search: expands the node with lowest value of evaluation function = h(n) + c(n)



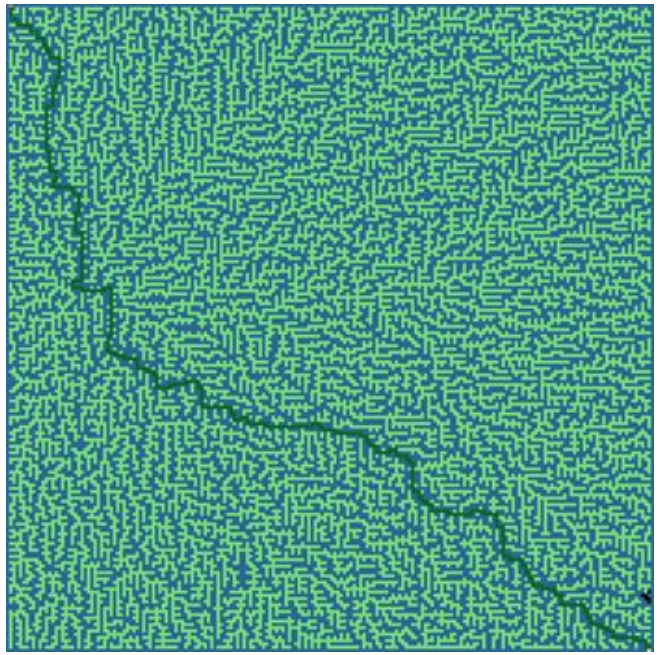
greedy best-first search



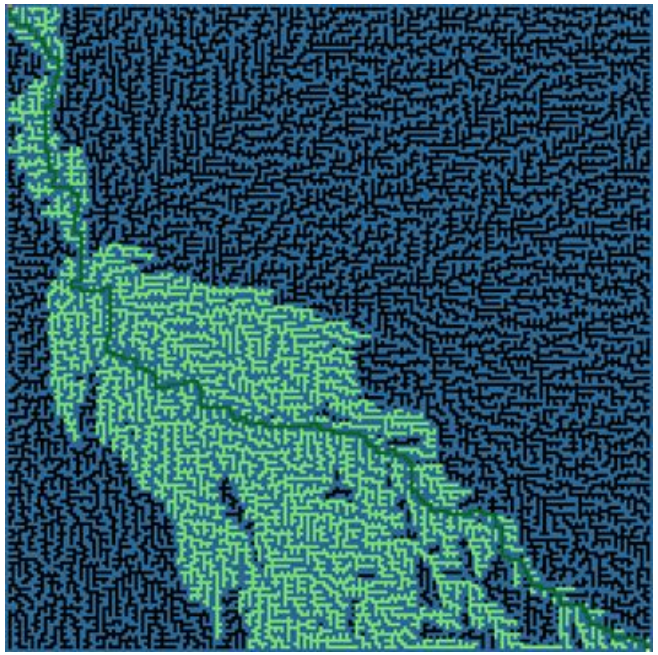
A* search

Contours of search

- breadth-first search expands in all directions
- A* expands mainly toward the goal, but enlarges its contours to ensure optimality



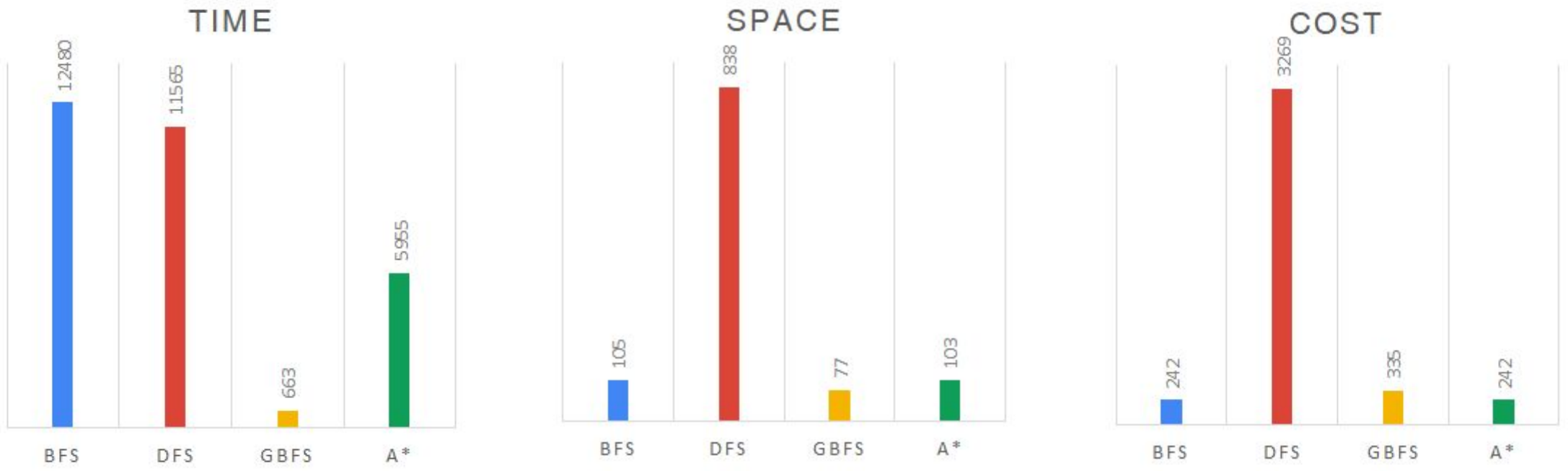
bfs



A*

Algorithm’s performance

Time complexity: number of nodes explored to find the solution
Space complexity: maximum number of nodes stored in frontier
Optimal accuracy: path solution is optimal or not



*average time, space complexity, solution observed in practical and collected on 200x200 maze dataset

Data analysis

- Most time-consuming: breadth-first search and depth-first search
- Most space-consuming: depth-first search
- Optimal: breadth-first search and A*
- Greedy best-first search doesn't guarantee the optimality but the deviation between its solution and the optimal solution is quite small
- Depth-first search mostly results in the longest path

MAZE	BFS			DFS			GBFS			A*		
INPUT	Time	Space	Path	Time	Space	Path	Time	Space	Path	Time	Space	Path
10x10	50	4	25	35	6	28	30	6	26	34	5	25
30x30	218	13	34	335	35	105	68	12	42	114	12	34
60x60	1039	27	86	1054	101	404	245	25	101	565	26	86
100x100	2926	48	132	3554	246	741	381	41	168	1468	43	132
200x200	12864	105	242	11565	838	3269	663	77	335	5955	103	242

Conclusion

- Greedy best-first search is the most efficient and admissible algorithm to solve maze problems among these algorithms with the best time-saving and space-saving result
- A* search is a more acceptable method compared to breadth-first search in term of optimality
- In general, space complexity among all methods in practical are different from theory and space complexity of depth-first search is often the largest

Thanks for listening!