

University of Birmingham

*MSc Computer Science*

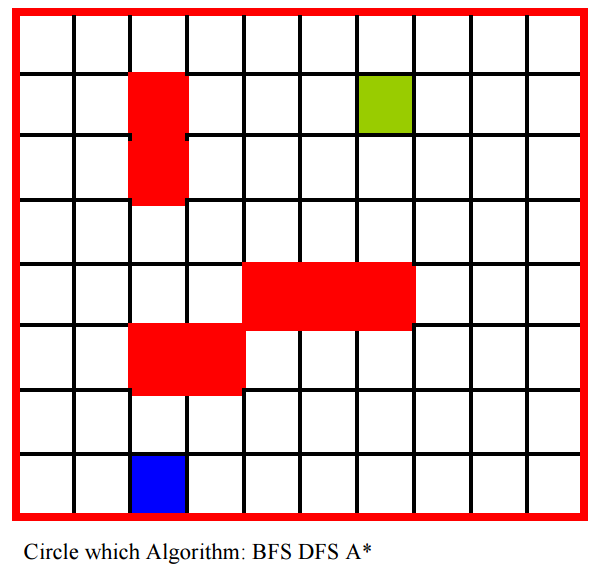
**Artificial Intelligence Mid-Semester Assignment**

Assignment prepared by:

**AXT114**

**Abdikhaliq Timer**

# Question 1



68

67

66

65

64

63

62

61

60

59

58

57

56

55

54

53

52

51

50

49

48

47

46

45

44

43

42

41

40

39

38

37

36

35

34

33

32

31

30

29

28

27

26

25

24

23

22

21

20

19

18

17

16

15

14

132

12

11

10

9

8

7

6

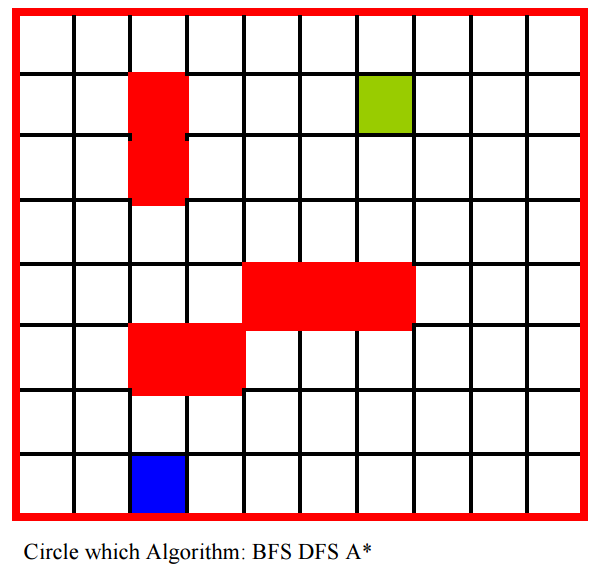
5

4

3

0

BFS uses a Queue data structure to process the nodes.



26

25

24

23

22

21

20

19

18

17

16

14

13

11

15

10

12

9

7

6

5

8

4

3

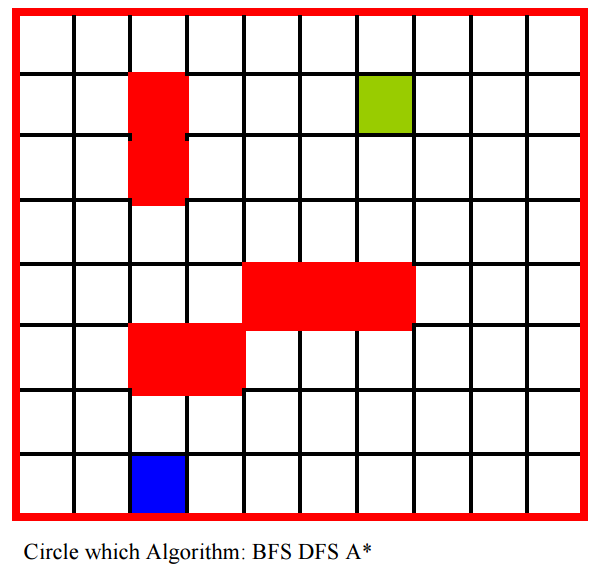
2

1

0

DFS uses a Stack data structure to process the nodes.

More efficient than BFS as it stores number of nodes at the max height of the DFS tree within a stack, whilst the BFS stores every adjacent node it processes within a queue.



66

**76**

62

24

**88**

64

62

**96**

34

90

**138**

48

48

**96**

48

38

**90**

52

58

**110**

52

72

**110**

38

38

**90**

52

34

**96**

62

86

**110**

24

14

**104**

90

86

**96**

10

86

**120**

34

86

**100**

14

90

**132**

42

38

**124**

86

86

**110**

24

48

**124**

76

72

**106**

34

76

**96**

28

76

**96**

20

24

**96**

72

54

**116**

62

62

**96**

34

58

**102**

44

84

**112**

28

24

**104**

80

76

**104**

28

66

**90**

24

48

**98**

50

48

**88**

40

54

**96**

42

38

**82**

44

28

**96**

68

24

**82**

58

28

**76**

48

80

**90**

10

70

**84**

14

72

**82**

10

62

**76**

14

66

**76**

10

0

0

0

A\* search, for which I used the f(p) = PathCost(p) + h(p). Where the

# Question 4 (3 points)

Name an **advantage** of each algorithm used in questions 1, 2, 3:

**Breadth First Search**: Complete and Optimal

It will always find the solution, and if there is more than one solution (end goal), it will find the solution with the minimal steps (optimal solution).

**Depth First Search**: Time and Space complexity

memory requirement is linear with relation to the nodes, less time and space complexity in comparison with BFS and solution can be found without searching most of the tree.

**A\* Search**: Complete and Optimal

No other optimal algorithm is also guaranteed to expand few nodes than the A\* algorithm. Furthermore, it considers both the cost and the distance already travelled, g(n). As our search is Admissible, it will never overestimate the actual cost of the cheapest path from a node to the goal.

Which algorithm would you use to **solve this problem**? Briefly explain why.

-wont go down blind allet BFS vs DFS

Applications:

BFS is used for: finding the shortest path from source node to other nodes in an unweighted graph like the one from our example, checking a graph that has bipartiteness (hence determining if the graph being searched is Bipartite – meaning the vertices within the graph can be divided into two disjoint sets, where vertices within their own set are independent of one another).

DFS is

# Question 5

You are playing **tic-tac-toe (noughts and crosses)**. We define **Xn** as the number of rows, columns, or diagonals with exactly n X’s and no O’s. Similarly, **On** is the number of rows, column, or diagonals with just n O’s. The utility function assigns +1 to any position with X3=1 (there is at least one row, column, or diagonal with 3 X’s and no O’s, therefore player X wins) and -1 to any position with O3 = 1 (player O wins). All other terminal positions (draws) have utility 0. For non terminal positions, denote here as s, we use a linear evaluation function defined as *Eval(s)=3X2(s)+X1(s)-(3O2(s)+O1(s)).*

1. Show the whole game tree starting from an empty board down to depth 2 (i.e. one X and one O on the board), taking symmetry into account.
2. Mark on your tree the evaluations of all the positions at depth 2. Start always from the most left position, then centre and right.
3. Using the minimax algorithm, mark on your tree the backed-up values for the positions at depth 1 and 0, and use those values to choose the best starting move.
4. Circle the nodes at depth 2 that would not be evaluated if alpha-beta pruning were applied.