KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)
Deemed to be University U/S 3 of UGC Act, 1956

TITLE :
SERVE_BUTTER_AI

Submitted to : N biraja
Submitted by : Shambhavi Kashyap
(2128121)
Date of submission:3/11/2023

ABSTRACT:

1.Problem: This is an AI project which is developed based on AI Search algorithms , here the problem is serving butter to customers on a table, for this purpose an AI agent named robot has been developed that works based on previously mentioned algorithms to serve customers ASAP.

2.Findings: Our working shows in the end the path cost and steps that agent had to take to serve/find solution to the same above mentioned problem for three different search algorithms,namely: A* Algorithm , Bidirectional BFS , Iterative Deepening Search.

Table of Contents:

1.Problem statement:

The game is about a robot which tries to serve customers ASAP. First, There is a bunch of foods on a table and all foods are given to the customers previously except for butter. This robot's duty is to give butter to the customer by putting it in a specific position on the table.

2.Algorithms and techniques used:

This game is a practice of three AI search algorithms:

- **A* Algorithm:**

  A* Search algorithm is one of the best and popular technique used in path-finding and graph traversals.

- **Bidirectional BFS**

  Bidirectional search is a graph search algorithm which find smallest path from source to goal vertex. It runs two simultaneous search –

Forward search from source/initial vertex toward goal vertex
Backward search from goal/target vertex toward source vertex
Bidirectional search replaces single search graph(which is likely to grow exponentially) with two smaller sub graphs – one starting from initial vertex and other starting from goal vertex. The search terminates when two graphs intersect.

- **Iterative Deepening Search:**
  IDDFS combines depth-first search's space-efficiency and breadth-first search's fast search (for nodes closer to root).

  How does IDDFS work?
  IDDFS calls DFS for different depths starting from an initial value. In every call, DFS is restricted from going beyond given depth. So basically we do DFS in a BFS fashion.

1.A* algorithm:

// A* Search Algorithm
1. Initialize the open list
2. Initialize the closed list
   put the starting node on the open list (you can leave its f at zero)

3. while the open list is not empty
   a) find the node with the least f on the open list, call it "q"

   b) pop q off the open list

   c) generate q's 8 successors and set their parents to q

d) for each successor
i) if successor is the goal, stop search

ii) else, compute both g and h for successor
successor.g = q.g + distance between
successor and q
successor.h = distance from goal to
successor (This can be done using many
ways, we will discuss three heuristics-
Manhattan, Diagonal and Euclidean
Heuristics)

successor.f = successor.g + successor.h

iii) if a node with the same position as
successor is in the OPEN list which has a
lower f than successor, skip this successor

iV) if a node with the same position as
successor is in the CLOSED list which has
a lower f than successor, skip this successor
otherwise, add the node to the open list
end (for loop)

e) push q on the closed list
end (while loop)

2.Bidirectional BFS:

Performance measures

1.Completeness : Bidirectional search is complete if BFS is used in both searches.
2.Optimality : It is optimal if BFS is used for search and paths have uniform cost.

3.Time and Space Complexity : Time and space complexity is O(bd/2).

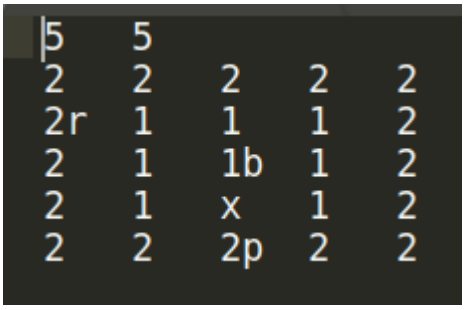
3.Iterative Deepening Search:

-ALGORITHM:

```
/ Returns true if target is reachable from
// src within max_depth
bool IDDFS(src, target, max_depth)
    for limit from 0 to max_depth
       if DLS(src, target, limit) == true
           return true
    return false

bool DLS(src, target, limit)
    if (src == target)
        return true;

    // If reached the maximum depth,
    // stop recursing.
    if (limit <= 0)
        return false;

    foreach adjacent i of src
        if DLS(i, target, limit?1)
            return true

    return false
```

4.About the robot’s(AI agent’s) working:

The robot's sensors get a first perception of items on the table at first. The sensor driver puts that perception as a text file with the following format into a specific directory:

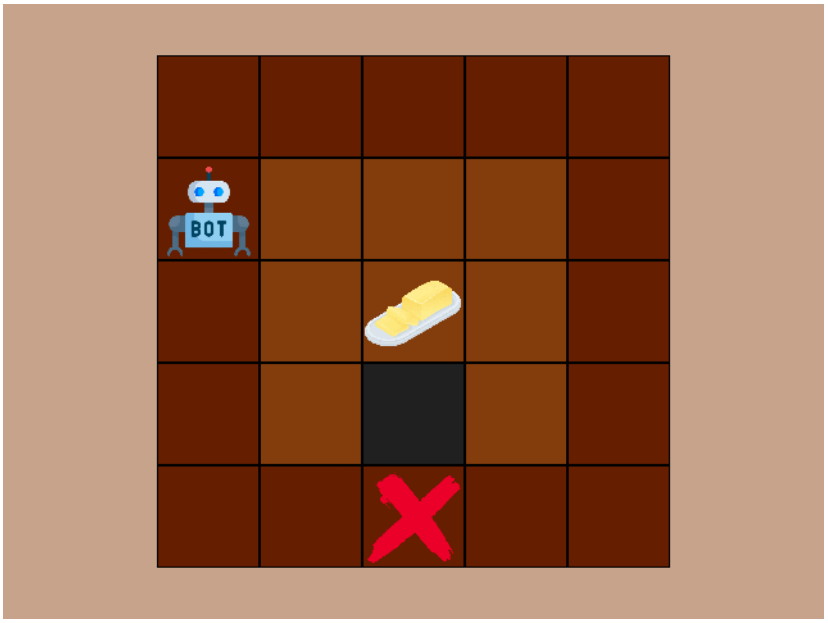


First line is the width and height of the table. Other lines describe the table items. Items are these:

-
- Numbers are cost of moving over that part of table.
- 'r' is the position of the robot itself.
- 'b' is the first position of a butter.
- 'p' is a point to put a butter on it.
- 'x' is block with other foods

3.Result

After finding shortest path, the output will be something like this:



4.Findings:

1.A* Algorithm :

PATH: RDRURDDRDL
Total moves: 10
Total cost: 14

2.Bidirectional BFS:

PATH: Found 2 possible ways.
Found a way with 10 moves.
DRRURDDRDL

Total moves: 10

Total cost: 15

3.Iterative Deepening Search:

PATH: Starting with depth 1
Starting with depth 2
Starting with depth 3
Starting with depth 4
Starting with depth 5
Starting with depth 6
Starting with depth 7
Starting with depth 8
Starting with depth 9
Starting with depth 10
RDRURDDRDL

Total moves: 10
Total cost: 14

—----------THANK YOU—----------
(2128121)