## CNG 462 Artificial Intelligence
### Assignment 1

| **Date Handed Out** | **Date Submission Due** |
|---|---|
| November 20, 2020 | Till the end of the day of Nov 30, 2020 |

## Introduction

This assignment has two parts: two short written questions and a practical programming question. You need to hand in a report including your answers for part 1 and 2, and also you need to submit your code for part 2.

## Part 1: Questions

1. Consider the task of solving the traveling salesman problem (TSP) [1, 2, 3]. Define this problem space in terms of a search problem. You need to discuss the **initial state**, **goal test**, **state space** (i.e., the possible states), **successor function** and **cost function**, additionally you need to also discuss at least **two possible heuristics** that can be used in informed search algorithms.

2. The TSP can be solved with the minimum-spanning tree (MST) heuristic [4, 5], which estimates the cost of completing a tour, given that a particular tour has already been constructed. The MST cost of a set of cities is the smallest sum of the link costs of any tree that connects all the cities.

   a. Show how this heuristic can be derived from a relaxed version of the TSP.
   b. Show that the MST heuristic dominates straight-line distance.

## Part 2: Programming Task

The second part of this assignment is adopted from here [6]. The overall goal of this assignment is to enable you practice how to use search algorithms we learnt in the class with a real-world problem. In this programming assignment, you will solve a variant of the traveling salesman problem (TSP) using A* search algorithm with Manhattan distance heuristic.

Our variant of the TSP problem is defined as follows: Given a robot on a map, (1) construct a shortest path graph by calculating the pair-wise distance for all checkpoints on the map, and (2) then find the shortest way for the robot to visit all the checkpoints exactly once and return to its starting checkpoint.

To solve the robot problem as explained above, there are two main sub-parts of this assignment. (1) Construct the shortest path graph, (2) solve the TSP problem on the shortest path graph with BFS and UCS.

## 1. Shortest Path Graph

In the first part of the programming task, you need to compute the shortest path graph. Given the map, to construct the shortest path graph, you need to find a shortest path between each pair of the checkpoints. <u>For example,</u>
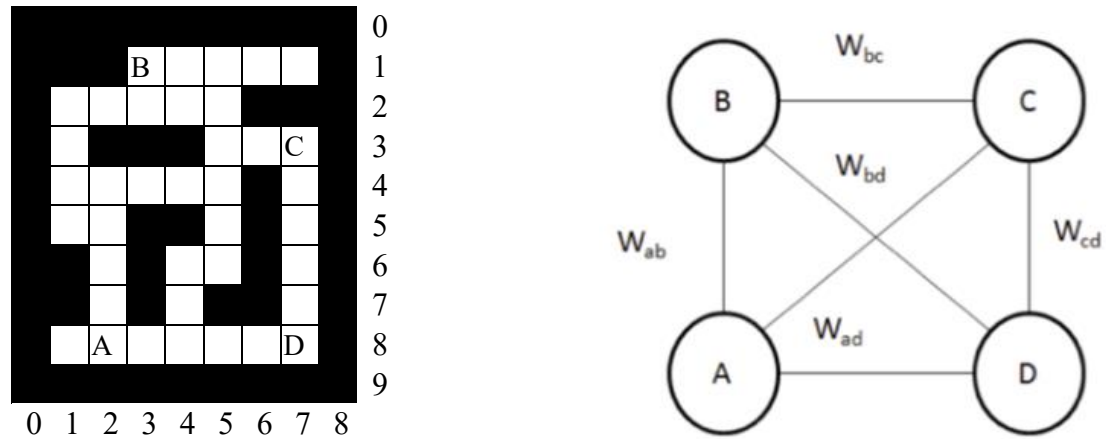


**Fig. 1.** Shortest path between each pair of points.

The value of Wab, Wac, Wad, Wbc, Wbc, Wcd can be determined by finding the shortest path between two checkpoints. <u>For example,</u>



$W_{ab} = 10$      $W_{ac} = 10$      $W_{ad} = 5$

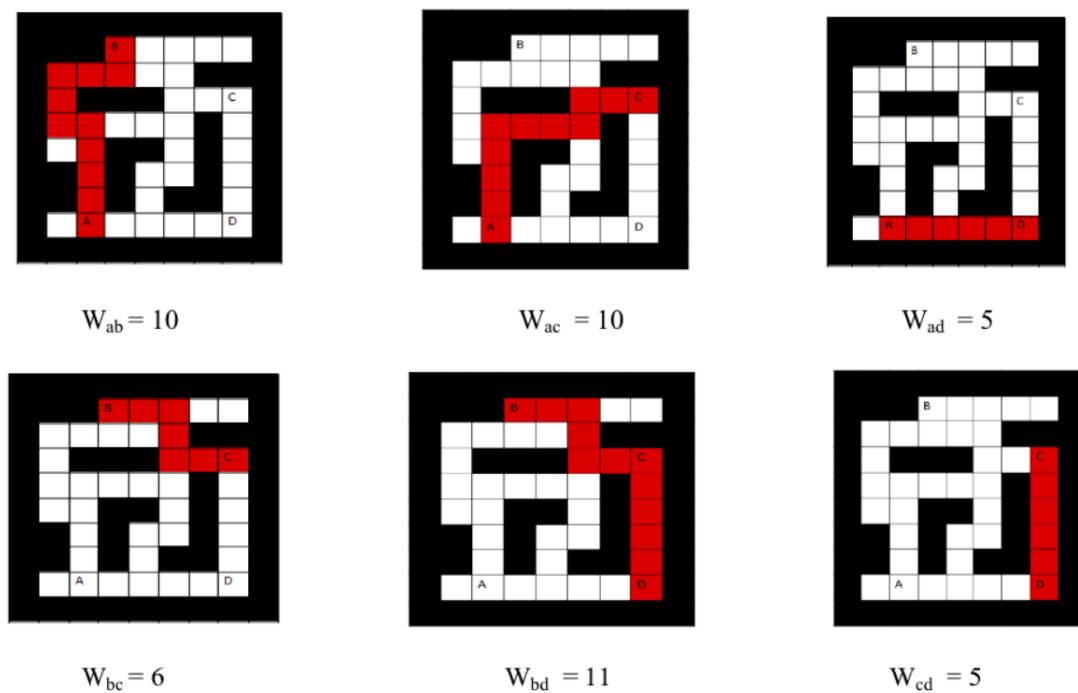$W_{bc} = 6$      $W_{bd} = 11$      $W_{cd} = 5$

**Fig. 2.** Shortest path calculations.

To find the shortest path between two check points, you need to use A* search algorithm with

Manhattan distance between two points as a heuristic. Please note that: the robot can move only in four directions: Up, Right, Down, Left. You can assume that the graph is fully connected – meaning that all checkpoints are reachable from all other checkpoints. Given the left map (Fig. 3), the shortest path graph should be
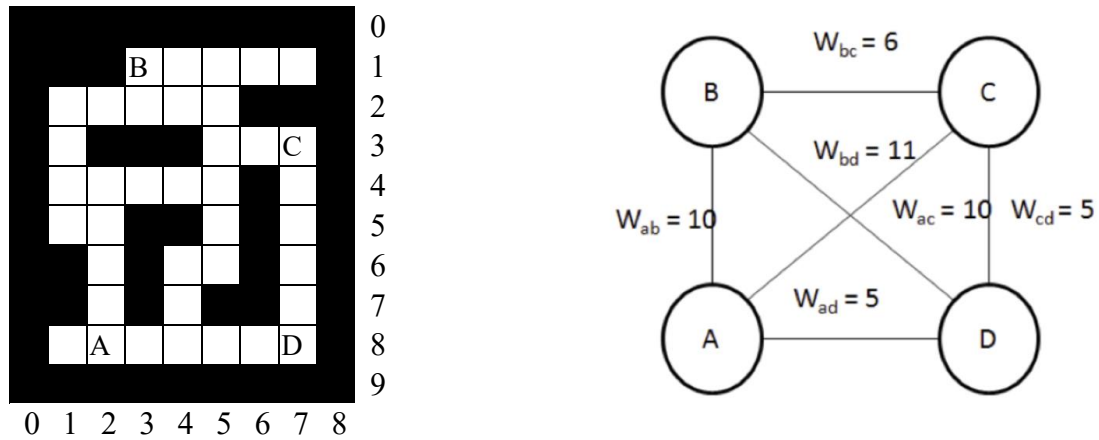


**Fig. 3.** Shortest path graph example.

**Tiebreaking for Task 1:** if there are multiple equivalent choices about which node goes next, selecting the node that is first in positional order manner on the image below.
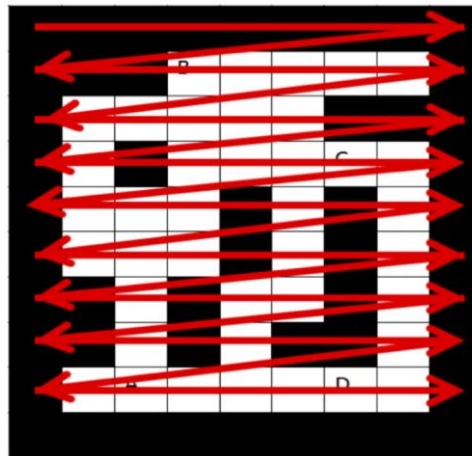


**Fig. 4.** Tiebreaking for Task 1.

## 2. Traveling Salesman Problem (TSP)

Your second programming task is to compute the shortest tour starting at checkpoint A. Once you construct the shortest path graph, the problem is transformed to the standard TSP. TSP is a well-known NP-Complete problem. Given the fully connected graph from previous section, and suppose the robot is always at `checkpoint A` in the first place, you need to find a TSP tour that the robot can visit all the checkpoints exactly once and returning to the starting `checkpoint A` with minimal distance.
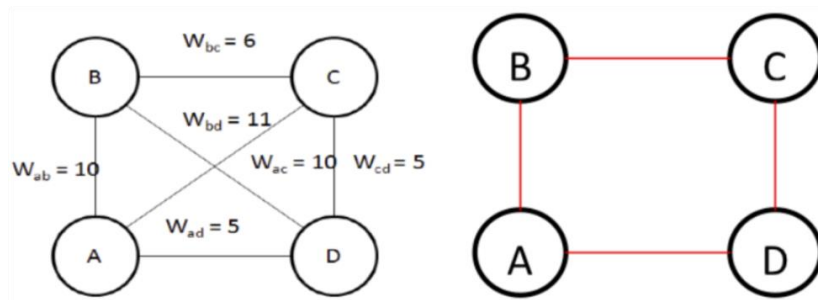
**Fig. 5.** TSP example.

**Starting checkpoint for task 2:** The starting checkpoint for the robot for Task 2 is always at 'A'. For Task 2, you will use Breadth-First Search (BFS) and Uniform Cost Search (UCS).

## 3. Tasks

In this assignment, you will write a program to implement the following tasks.
1. Construct a shortest path graph with A* using Manhattan distance as a heuristic.
2. Using a graph from the first task, find the optimal tour with UCS and BFS. Your program MUST compute the shortest path graph first before proceeding to solve TSP. TSP cannot be solved in your program without the shortest path graph.

## 4. Input

There are two inputs for your programs:
1. Which task to perform: there are two possible values: (1) Construct a shortest path graph and (2) TSP Solution with BSF and UCS.
2. The input map file is given in Fig. 6. Name the input file as **map.txt**. You can download map.txt from ODTUClass.
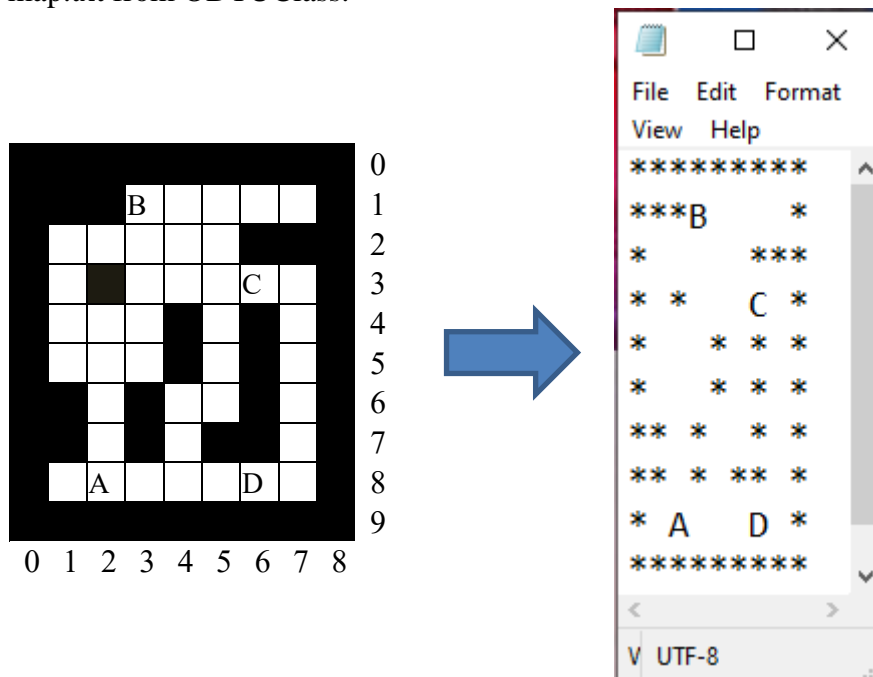


**Fig. 6.** Input map file.

## 5. Output

In this assignment, we are interested in both how your search algorithms traverse the graph and the path. There are two output files that your program needs to produce.

4

1. **A shortest path graph (Task 1):** in an adjacency list format. Each line contains three columns: node1, node2, and weight. Each column is separated by a comma. Print the results in alphabetic order. <u>Example output</u>[1],

```
a,b,10
a,c,11
a,d,5
b,c,6
b,d,10
c,d,5
```

2. **A tour (Task 2):** List the names of the nodes in the tour (separated by new lines) in order, and you need to also print statistics. <u>Example output</u>[2],

```
Algorithm Used: BFS
a-b-c-d-e-f-a
Total Tour Cost: 50.0

Algorithm Used: UCS
…

Statistics:
        Nodes       Time        Cost
BFS     x           x           x
UCS     x           x           x
```

---

[1] The values given are for illustration only.
[2] The values given are for illustration only.

## 6. Program specifications

Your program can be written in C, C#, C++, or Python.

## 7. Submission

You need to submit a ZIP file (firstname_lastname.zip, e.g. `okan_topcu.zip`) including the following:

1. Report in PDF format for part 1.
2. Your code in part 2.
3. A sample run screenshot of your part 3.
4. Readme.txt file for your part 2 code. This should include a short description of your program and it should explain how to compile and run your code, please include your full name and id at the top.

To submit your assignment, simply select the appropriate assignment link from the ODTUCLASS page. Upload your zip file and click submit (clicking send is not enough). Please make sure ALL source files are included in your zip file when submitted. A program that does not compile as submitted will be given 0 points. Only your **final submission** will be graded. Remember there is no late submission for this assignment.

## Grading

| Grading Point | Total (100) |
|---|---|
| Part 1 Question 1 (Detailed problem specification and understanding) | /10 |
| Part 1 Question 2 (Detailed description of the problem and detailed justification of the answers in 1 and 2) | /15 |
| Part 2 Readme.txt (Your code should include a readme.txt that has a short description of your program and explains how to compile and run your code, please include your name, surname and id at the top) (you cannot get this if you do not submit a code) | /10 |
| Part 2 Task 1 (your program should clearly document how you solved the shortest path problem with the given heuristic) | /20 |
| Part 2 Task 2 (TSP with BFS, statistics on the algorithm) | /15 |
| Part 2 Task 2 (TSP with UCS, statistics on the algorithm) | /15 |
| Part 2 Output (Your output clearly shows the shortest path graph and also the TSP solution for BFS, UCS and also compares the statistics) | /15 |
| **TOTAL** | /100 |

# References

[1] Wolfram MathWorld, «Travelling salesman problem,» [Çevrimiçi]. Available: http://mathworld.wolfram.com/TravelingSalesmanProblem.html. [Erişildi: 5 November 2019].

[2] «Travelling salesman problem,» Wikipedia, [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/Travelling_salesman_problem. [Erişildi: 5 November 2019].

[3] «Traveling Salesman Problem Visualization,» Google, [Çevrimiçi]. Available: https://www.youtube.com/watch?v=SC5CX8drAtU. [Erişildi: 5 November 2019].

[4] «Minimum spanning tree,» Wikipedia, [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/Minimum_spanning_tree. [Erişildi: 5 November 2019].

[5] «The Traveling Salesman Problem 4: Spanning Tree Heuristic,» Wolfram, [Çevrimiçi]. Available: https://demonstrations.wolfram.com/TheTravelingSalesmanProblem4SpanningTreeHeur istic/. [Erişildi: 5 November 2019].

[6] [Çevrimiçi]. Available: http://xn--www-scf-806c.usc.edu/~kangwang/doc/Assignment%202.pdf.

[7] X. Geng, Z. Chen, W. Yang, D. Shi ve K. Zhao, «Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search,» *Applied Soft Computing,* cilt 11, no. 4, pp. 3680-3689, 2011.

Please note that you can refer to the AIMA online code repository of your textbook for these algorithms:

http://aima.cs.berkeley.edu/code.html