**CS 540-1: Introduction to Artificial Intelligence**
**Homework Assignment # 2**

**Assigned: 9/13**
**Due: 9/20 before class**

## Hand in your homework:

This homework includes one written question and one programming question. Please type the written portion and hand in a pdf file named **hw2.pdf**. For the programming question, hand in the Java program **successor.java**. Go to UW Canvas, choose your CS540-1 course, choose Assignment, click on Homework 2: this is where you submit your files.

## Problem 1. Search [60]

Given the undirected, unweighted graph in Figure 1, find the shortest path **from A to G** using different search algorithms.

You are to "run" the algorithms by hand. Use a CLOSED data structure to avoid cycles. Specifically, for uninformed search and uniform cost search follow slides 28 and 43 (Hint: these slides allow multiple copies of the same node in OPEN); for A* search follow slide 21.

To break ties, assume nodes are expanded (taken out of the OPEN data structure) in alphabetical order with everything else being equal.

For each algorithm, write down :

- States of OPEN and CLOSED in each step of the algorithm, together with the appropriate back pointer (see example steps below)
- The solution path found

a) [10] Breadth First Search.

The first few iterations of Breadth First Search have been shown to help you start.

| **OPEN (Queue)[Top is towards right]** | **CLOSED** |
|---|---|
| A(Null) | |
| D(A), C(A), B(A) | A(Null) |
| E(B), A(B), D(A), C(A) | A(Null), B(A) |
| ... | ... |

b) [10] Depth First Search.

c) [10] Iterative Deepening. Let us use the following convention: for the first depth cutoff (the very first iteration of the outer loop) we will goal check A and also A's successors, but nothing further.

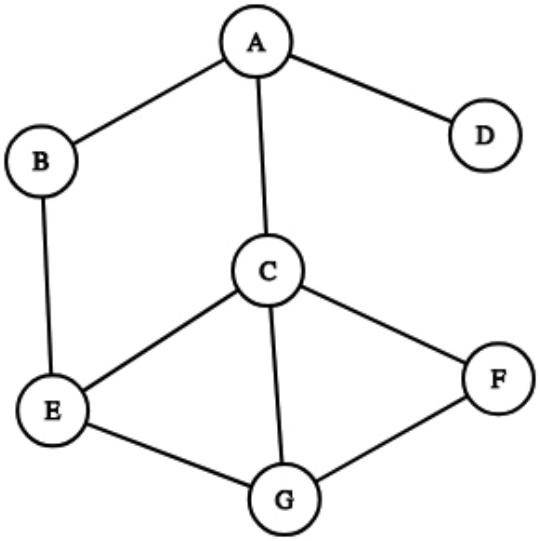


Figure 1: The unweighted, undirected graph

Now suppose we have different edge costs and heuristic function $h$ as in Figure 2.

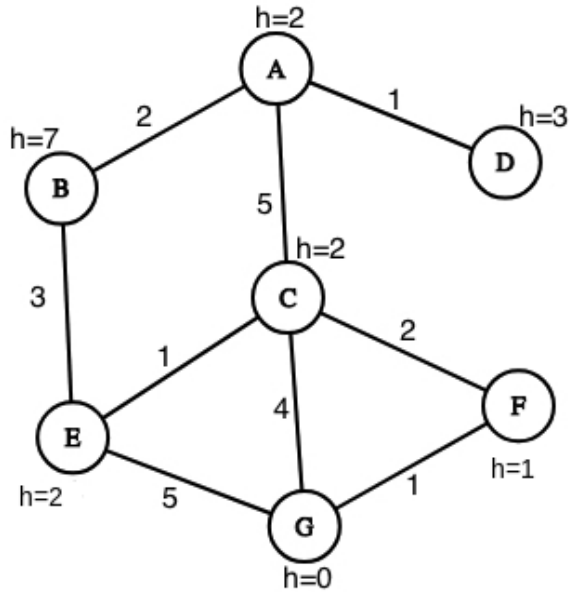


Figure 2: The weighted graph and the $h(\cdot)$ function

For parts (d) and (e) also include the node score (for A* this is $g + h$) in your answer alongside with the back pointer.

d) [10] Perform **Uniform Cost Search**. The first few steps are shown for Uniform Cost Search in the example table below.

| **OPEN (Priority Queue)[Top is towards right]** | **CLOSED** |
|---|---|
| A(0,Null) | |
| C(5,A), B(2,A), D(1,A) | A(0,Null) |
| ... | ... |

e) [20] Perform **A* search** with the given heuristic function $h(\cdot)$ in Figure 2.

# Problem 2. Successor Function [40]

This is a programming question. The solution to the programming problem should be coded in Java, and **you are required to use only built-in libraries** to complete this homework. Please submit a source code file named **successor.java** with **no package statements**, and make sure your program is runnable from command line on a department Linux machine. We provide a skeleton successor.java code that you can optionally use, or you can write your own.

The goal of this assignment is to become familiar with the state space in a real-world problem.
You are given two water jugs with capacity A and B liters respectively. A and B are positive integers. At each step, you can perform one of these actions:

- Empty a jug
- Fill a jug
- Pour water from one jug to another until either the former is empty or the latter is full.

Write a program **successor.java** to print the successor states of an input state.

- Input: 4 integers $A, B, a, b$, where $(A, B)$ are the capacity of the jugs, and $(a, b)$ are the current amount of water in each jug. You may assume that the input is valid, namely $a \leq A, b \leq B$, and $A, B$ are positive integers, and $a, b$ are non-negative integers.
- Output: Print the list of successor states reachable in one step from $(a, b)$, one on each line. The lines must be sorted as per the examples shown (using the provided sorting function in the skeleton code). The successors should not include $(a, b)$ itself or duplicates.

Here are some examples of running the program from the command line. The inputs and outputs are space-separated with no comma. Please follow the same input/output format.
Example 1:

```
$java successor 3 2 3 1
0 1
2 2
3 0
3 2
```

Example 2:

```
$java successor 11 5 6 3
0 3
4 5
6 0
6 5
9 0
11 3
```