National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 2: Uninformed Search

Issued: January 23, 2020 Due: Week 4 in the tutorial class

Important Instructions:

- *Your solutions for this tutorial must be TYPE-WRITTEN.*
- *Make TWO copies of your solutions: one for you and one to be SUBMITTED TO THE TUTOR IN CLASS. Your submission in your respective tutorial class will be used to indicate your CLASS ATTENDANCE. Late submission will NOT be entertained.*
- *YOUR SOLUTION TO QUESTION* 3 *will be GRADED for this tutorial.*
- *You may discuss the content of the questions with your classmates, but you should work out and write up ALL the solutions by yourself.*

1. Sudoku is a popular number puzzle that works as follows: we are given a 9×9 square grid; some squares have numbers, while some are blank. Our objective is to fill in the blanks with numbers from $1 - 9$ such that each row, column and the highlighted 3×3 squares contain no duplicate entries (see Figure 1). Solving Sudoku puzzles is often modeled as a CSP (which we will cover later in the course). Consider, however, the problem of *generating* Sudoku puzzles. Consider the following procedure: start with a completely full number grid (see Figure 2), and iteratively make some squares blank. We continue blanking out squares as long as the resulting puzzle can be completed in only one way. Questions:
   - Give the representation of a state in this problem;
   - Using the state representation defined above, specify the initial state and goal state.
   - Define its actions; and
   - Using the state representation and actions defined above, specify the transition model/function T (i.e., when each of the actions defined above is applied to a current state, what is the resulting next state?).
2. Consider a class managed by an instructor. The instructor allocates one day to meet with students to discuss a final project; the day has t time slots in it denoted $S = \{s_1, \dots, s_t\}$. Every student i indicates their availability denoted by a subset $A_i \subseteq S$. Our objective is

2	5			3		9		1
	1				4			
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

Figure 1: A simple Sudoku Puzzle

2	5	8	7	3	6	9	4	1
6	1	9	8	2	4	3	5	7
4	3	7	9	1	5	2	6	8
3	9	5	2	7	1	4	8	6
7	6	2	4	9	8	1	3	5
8	4	1	6	5	3	7	2	9
1	8	4	3	6	9	5	7	2
5	7	6	1	4	2	8	9	3
9	2	3	5	8	7	6	1	4

Figure 2: Solution to the Sudoku puzzle in Figure 1.

to schedule all students for a meeting, while respecting their availability constraints[1]. Your goal is to model this as a search problem, with the state space defined as a partial (valid) allocation of some of the students into the time slots.

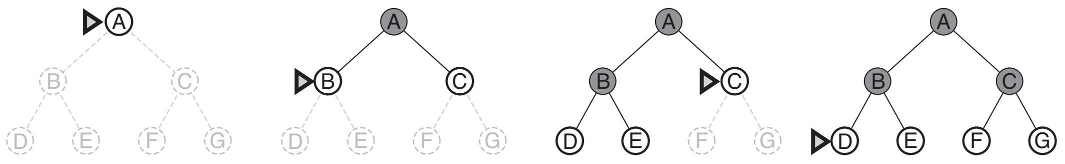

Figure 3: BFS on a simple binary tree, At each stage, the node to be expanded next is indicated by a marker.

Suppose there are three time-slots, and three students whose availability is given below. Assume that whenever you need to choose a student to add to a schedule, you do so in the order (A)lice (B)ob and then (C)laire. In addition, when choosing a time slot, ties are always broken in order of $s_1 \succ s_2 \succ s_3$.

	s_1	s_2	s_3
Alice	✓	✓	✓
Bob	✓		
Claire		✓	

(a) Give a trace of the BFS algorithm in the style of Figure 3.
That is, show the search tree at each stage (all repeated states are eliminated).

(b) Give a similar trace of the DFS search algorithm.

(c) Which of these two search algorithms is better for this problem? Why? Is one search strategy always better than the other in general?

3. We have seen various search strategies in class, and analyzed their worst-case running time. Prove that *any deterministic search algorithm* will, in the worst case, search the entire state space. More formally, prove the following theorem

Theorem 1. *Let $\mathcal{A}$ be some complete, deterministic search algorithm. Then for any search problem defined by a finite connected graph $G = \langle V, E \rangle$ (where V is the set of possible states and E are the transition edges between them), there exists a choice of start node s_0 and goal node g so that $\mathcal{A}$ searches through the entire graph G.*

4. Consider the graph shown in Figure 4. Let S be the initial state and G be the goal state. The cost of each action is as indicated.

[1]This problem can also be modeled as a CSP, which we will cover in later classes.

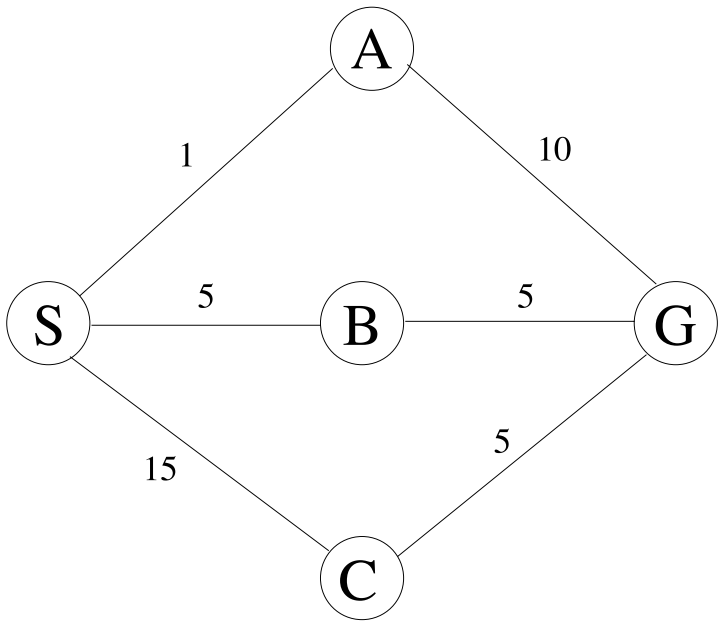

Figure 4: Graph of routes between S and G.

(a) Give a trace of uniform-cost search, assuming we run a graph search.

(b) When UCS generates G which is the goal with a path cost of 11, why doesn't the algorithm halt and return the search result since the goal has been found? Using this observation, prove the following claim (also shown in class)

Theorem 2. *For any search problem, UCS returns an optimal goal state.*