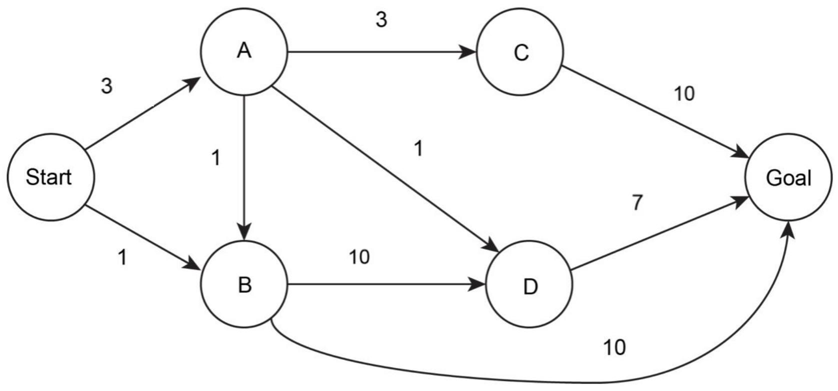
Artificial Intelligence Homework 1

Part A: Hand Writting Part

1. Uniform-Cost Graph Search

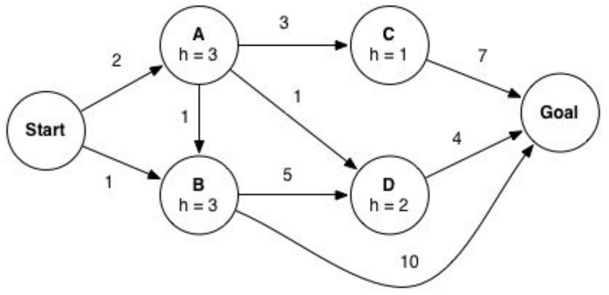
Consider the graph below. Arcs are labeled with their weights. Assume that ties are broken alphabetically (so a partial plan S->X->A would be expanded before S->X->B and S->A->Z would be expanded before S->B->A.



What path does uniform cost search return?

2. A star

Consider A* *graph* search on the graph below. Arcs are labeled with action costs and states are labeled with heuristic values. Assume that ties are broken alphabetically (so a partial plan S->X->A would be expanded before S->X->B and S->A->Z would be expanded before S->B->A.



1. In what order are states expanded by A* graph search? You may find it helpful to execute the search on scratch paper.

2. What path does A* graph search return?

3. Constraint Satisfaction Problems (Bonus!)

Use the AC-3 algorithm to show that arc consistency can detect the inconsistency of the partial assignment WA = red, V = blue for the problem shown in Figure 5.1.

You are supposed to detect inconsistency with a 3-color constraint
D = {red, green, blue}

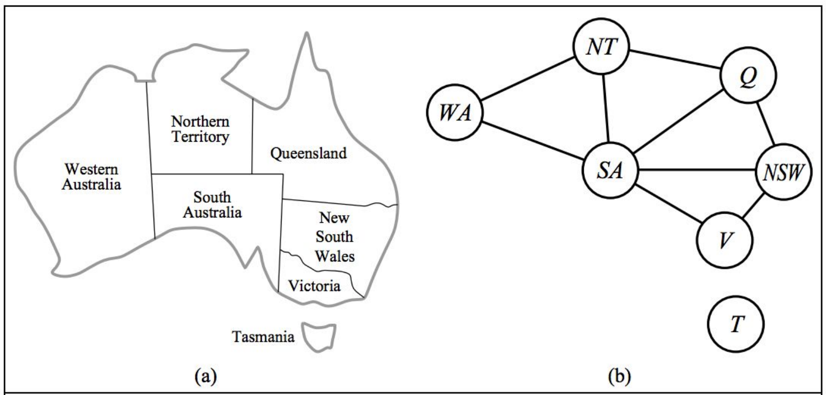


Figure 5.1 (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem. The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

What is the worst-case complexity of running the tree version of AC-3 on a tree-structured CSP? Why?

Part B: Programming Part

Search in Pacman

General Idea

Welcome to the world of pacman. This pacman program is based on the classic Berkeley CS188 course. We will use pacman as our HW1 and HW2. In HW1, you are required to implement a Depth First Search pacman to find a fixed food dot.

Step by Step:

1. Download pacman.zip from the course website.
2. Please use python 2.7
3. Run a pacman game by typing python pacman.py to terminal. (remember to change directory to pacman folder). You will see two ghosts and one pacman. You can use arrow key to control the pacman. But this fancy game is just for fun. Not related to our HW.

4. Run a search-based pacman game by typing
   python pacman.py -l tinyMaze -p SearchAgent -a fn=tinyMazeSearch
   You will see pacman moving forward to the food dot. The function tinyMazeSearch is defined in search.py. However, there is no search algorithm inside tinyMazeSearch. All the action is simply written by us. That is, this function is only suitable for tinyMaze, not for mediumMaze or bigMaze.

5. Now your work is to finish the depthFirstSearch function inside the search.py. By calling the following three commands:
   python pacman.py -l tinyMaze -p SearchAgent -a fn=dfs
   python pacman.py -l mediumMaze -p SearchAgent -a fn=dfs
   python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=dfs
   The pacman will eat the food dot in tinyMaze, mediumMaze and bigMaze.
6. Congrat! You finish HW1.

*** -l means which layout (maze) to use. -p means which pacman agent to run. -a means which algorithm to apply.*

Detail Info:

Files you'll edit:	
search.py	Where all of your search algorithms will reside. Please finish the Depth First Search part!
Files you must look at:	
util.py	Useful data structures for implementing search algorithms. Please use the data structures here only! Do not import any data structures from other python tools or package.
Files you might want to look at: (if you want to understand how pacman works)	
pacman.py	The main file that runs Pacman games. This file describes a Pacman GameState type, which you use in this project.
game.py	The logic behind how the Pacman world works. This file describes several supporting types like AgentState, Agent, Direction, and Grid.

Important note:
Your search functions need to return a list of actions:

['North','West','East','South']

That is, a list of strings. That will lead the agent from the start to the goal. These actions all have to be legal moves (valid directions, no moving through walls, we don't want to see a super pacman).

Important note:
Make sure to use the Stack, Queue and PriorityQueue data structures provided to you in util.py only!

Submission Deadline and Method:

Deadline:
Part A: Oct 2, 2017 Hand in your assignment during the lecture.
Part B: Oct 1, 2017 23:55 (UTC+8) Upload to courseweb.

Submission Method
For part A, write your answer on an A4 paper either in Chinese or English. Please use regular writing (do not use cursive writing). Remember to write your departement, student ID, and your name on top of the paper.

For partB, upload a zip file to the course website.

```
hw1_studentID.zip
   hw1_studentID/
        - search.py
```

Delay policy:
25 points off each day