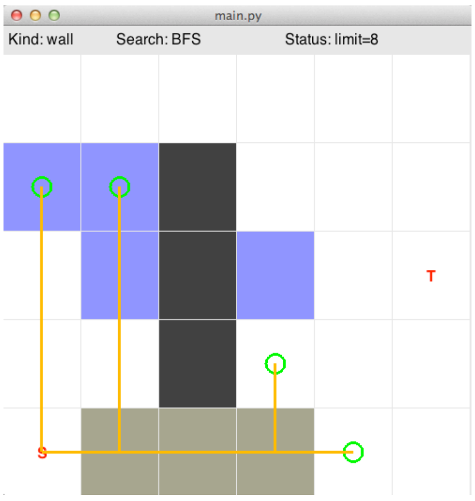
Lab – Graph, Paths and Searching

Objective: To understand and utilise Depth First Search (DFS), Breadth First Search (BFS), Dijkstra’s and A* Search algorithms applied to a simple “box” based world.

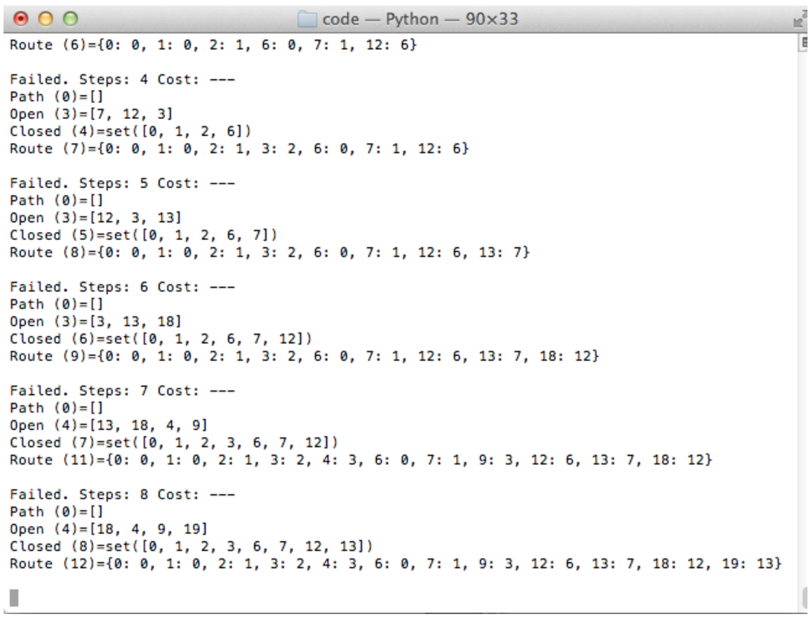
Quick Start:

Download code for this lab from blackboard. Extract to your favourite work location. The file to run is main.py. You will need to specify a map file to load. Something like:

```
C:>python main.py map1.txt
```



(a)



```
Route (6)={0: 0, 1: 0, 2: 1, 6: 0, 7: 1, 12: 6}

Failed. Steps: 4 Cost: ---
Path (0)=[]
Open (3)=[7, 12, 3]
Closed (4)=set([0, 1, 2, 6])
Route (7)={0: 0, 1: 0, 2: 1, 3: 2, 6: 0, 7: 1, 12: 6}

Failed. Steps: 5 Cost: ---
Path (0)=[]
Open (3)=[12, 3, 13]
Closed (5)=set([0, 1, 2, 6, 7])
Route (8)={0: 0, 1: 0, 2: 1, 3: 2, 6: 0, 7: 1, 12: 6, 13: 7}

Failed. Steps: 6 Cost: ---
Path (0)=[]
Open (3)=[3, 13, 18]
Closed (6)=set([0, 1, 2, 6, 7, 12])
Route (9)={0: 0, 1: 0, 2: 1, 3: 2, 6: 0, 7: 1, 12: 6, 13: 7, 18: 12}

Failed. Steps: 7 Cost: ---
Path (0)=[]
Open (4)=[13, 18, 4, 9]
Closed (7)=set([0, 1, 2, 3, 6, 7, 12])
Route (11)={0: 0, 1: 0, 2: 1, 3: 2, 4: 3, 6: 0, 7: 1, 9: 3, 12: 6, 13: 7, 18: 12}

Failed. Steps: 8 Cost: ---
Path (0)=[]
Open (4)=[18, 4, 9, 19]
Closed (8)=set([0, 1, 2, 3, 6, 7, 12, 13])
Route (12)={0: 0, 1: 0, 2: 1, 3: 2, 4: 3, 6: 0, 7: 1, 9: 3, 12: 6, 13: 7, 18: 12, 19: 13}
```

(b)

Figure 1. In (a) the GUI view of the program for a BFS search limited to 8 steps is shown. In (b) details of the last five step-limited results are shown, with the current open queue yet-to-be looked at, then closed set of nodes already considered, and the route list of the current search results.

There are two simple maps to start with named map1.txt and map2.txt. You can create your own later if you wish.

The worlds’ boxes can be changed with a combination of selecting the box “kind” (shown in the top left text) and then left-clicking on to a box. There are currently four different box kinds:

1: “clear” (white)
2: “mud” (grey-brownish)
3: “water” (blue)
4: “wall” (black)

Walls are not included (have no edges) in the navigation graph.

For each search a start (“S”) and target (“T”) box is needed. To change the location of either, first set the kind value, and click on to a box.

5: “start”
6: “target”

You can allocate start or target to wall box that has no edges, however this will stop any search from being successful!

Pressing the SPACE key will perform a search using the current map and search mode, however most changes to the world force a new search to be done immediately.

There are currently four different search modes, which can be cycled through using the N and M keys (backwards and forwards respectively). The search modes are:

"BFS" for Best First search algorithm
"DFS" for Depth First search algorithm
"Dijkstra" for Dijkstra's lowest-cost-so-far search algorithm
"A*" (written as "AStar") for the lowest cost-so-far + lowest-estimated-cost algorithm

At the end of this document is a list of keys and features. You can also read the code directly to understand what you can do, and alter it as you wish.

Search "depth" (the number of search steps taken) can be limited and changed using the UP and DOWN arrow keys. The limit can be removed using the "0" key.

Tasks

When a search is performed you can see several things both visually on the map and in the console text output. For example:

```
Success! Done! Steps: 14 Cost: 18.071
Path (4)=[18, 19, 26, 27]
Open (6)=[3, 9, 15, 16, 22, 28]
Closed (14)=set([0, 1, 2, 6, 7, 12, 13, 18, 19, 21, 24, 25, 26, 27])
Route: …
```

If a successful path has been found, it is shown. Note that a search can be successful (found the target) but not finished (still has alternatives to try). The number of steps and the cost are shown to enable comparison of algorithms. The route is also printed (but not shown) which is pairs of to:from index values which is used to represent the entire search tree and reconstruct the final search path.

Using the UP and DOWN keys, limit the search depth for each search type and observe the changes in the open and closed lists. The numbers match the box numbers, and can be displayed by pressing the "L" key.

Change the location of the Start and Target markers and observer the effect and performance of different searches.

Refer to the lecture notes on graph search and follow the search steps for a small (easy) situation. Look at the code in searches.py and see if you understand the differences for each search method. (They are quite similar.)

Making Cost-Based Searches Work Correctly.

There are a couple of things to change:
 * Initially, the world only has N-S-E-W connections. In order for

In the box_world.py module there is a variable for min_edge_cost used by the A* algorithm. It is currently set to the wrong value. Change this value and demonstrate (find a case) where the bahaviour between the old value (which can given non-optimal results) and you new value (which should give optimal results) are different. The console will show the path cost to assist in your comparisons.

Currently the navigation graph only uses N-S-E-W edges between boxes. A* is also currently using Manhattan distance to estimate cost. Find the code in the reset_navgraph method of the BoxWorld class (in the box_world.py module) and uncomment the code needed to add diagonal edges.

Is the use of Manhattan distance calculation correct now? Change this to another method by editing the reset_navgraph method of the BoxWorld class. (The correct code is currently commented out.)

By the end of this lab you should have a solid understanding of the differences between each of the search methods used, and be able to describe the behaviour of each. In particular you must understand the influence of the edge_cost_value on A* star search, and the different types of cost estimation heuristics (and how they must match the topology and the real cost in some way.

Optional Tasks

- Create your own map.
- Add additional box types, costs and colours.
- Create an agent the follows each node of the path in order.

All Keys

There are many other features that can be changed, and some keys have been assigned:

1: “clear” (white)
2: “mud” (grey-brownish)
3: “water” (blue)
4: “wall” (black)
5: “start”
6: “target”
SPACE: force a replan.
N and M: cycle (back or forward) through the search methods and replan.
UP and DOWN: Increase or decrease the search step limit by one.
0: Remove the search step limit. (A full search will be performed.)

E: toggle edges on/off for the current navigation graph (thin blue lines)
L: toggle box (node) index values on/off (useful for understanding search and path details).
C: toggle box “centre” markers on/off
T: toggle tree on/off for the current search if available
P: toggle path on/off for the current search if there is a successful route.