CAB203
Assignment 3

This assignment is intended to give you opportunities to work with the ideas developed in the lectures. To this end, you are encouraged to discuss questions, approaches, and background material with your fellow students. However, *every student must prepare and submit their own solutions. The solutions you submit must be your own.* For some questions you will benefit from reading or online research. If you use sources outside of the unit you are expected to properly cite your sources. You can use any citation style you prefer, as long as you are consistent with it.

Submit your solutions in PDF format through the Turnitin link on Blackboard. Scans of handwritten submissions are acceptable, but they must be of high quality. Scanners are available on campus.

1 Graphs

Back in Ye Olde Days *text adventure games* were a popular genre of computer games. These games had an interface that is reminiscent of command line interfaces. During a project in your computer archaeology unit, you have discovered an interesting example of a text adventure game, called *Bork*. Bork seems to take place in a place with strange multi-dimensional space-time, and your attempts to map out the locations in the game have so far been thwarted. You decide to automate the process and you soon have an interface working that allows you to move the character around programmatically. From the interface you can identify your current location (through the description of the location, which is unique to that location), get a list of exits from the current location, or move to a new location by choosing an exit. Now you need to decide how to map out the game. The interface is a bit slow, so it takes some time to move between adjacent locations.

a. Describe how you can use a graph to model the locations in the game and which other locations you can visit directly from each location. You must specify what the vertices and edges are and how they correspond to elements in the game.

(1 mark)

b. Discuss a graph traversal strategy (breadth first or depth first) that would work best in this scenario to map out the locations in the game using the graph model that you suggested above. Be sure to mention the advantages of your chosen strategy for this scenario, and the disadvantages of the other strategy.

(1 mark)

c. Suppose you have built your graph mapping out the game. Discuss one graph related algorithm (eg. finding a spanning tree) discussed in class and how it could be usefully applied to your graph to help a person play Bork.

(1 mark)

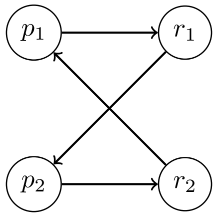

Figure 1: Minimal example of a resource allocation graph with deadlock.

2 Directed graphs

Directed graphs are sometimes used in operating systems when trying to avoid *deadlock*, which is a condition when several processes are waiting for a resource to become available, but this will never happen because other processes are holding on to those resources and are themselves waiting for other resources. In the simplest example, process p_1 is holding resource r_1 and is waiting for resource r_2 while process p_2 is holding resource r_2 and waiting for resource r_1. This is depicted in figure 1.

Let R be a set of resources and P be a set of processes (not necessarily the one depiced in figure 1). Define the *resource allocation graph* to be the directed graph G on $R \cup P$ with edges (p, r) where process p is holding resource r, and (r, p) where process p is waiting for resource r. To avoid trivial cases, we will assume that $P, R \neq \emptyset$. Note that G will always be anti-symmetric since a process will not wait for a resource that it already has. Also, G is bipartite with P and R forming the partition on the edges.

For our purposes, there are a few things that can happen. The operating system's actions are:

- if there is a process p that is not waiting for any resources — that is, there are no edges (r, p) for any r — the operating system can choose to *run* p. Only one process can be running at a time. The process runs until it performs an action on the graph (discussed below).
- if (r, p) is an edge and there are no edges (p', r) for any p', the operating system can assign resource r to process p, adding edge (p, r) to the graph and removing (r, p)

If at least one of the above is possible, then the system can *make progress*.

Process have the following possible actions:

- if p is running and (p, r) is an edge, then p can release a resource r, deleting edge (p, r) from the graph
- a running process p can request a resource r, adding edge (r, p) to the graph

a. A *minimal vertex* or *minimal element* in a directed graph is a vertex v such that there are no edges (u, v) in the graph for any u. Argue that if the resource allocation graph $G = (P \cup R, E)$ has a minimal vertex $p \in P$ then the system can make progress.

(1 mark)

b. Suppose that a resource graph $G = (P \cup R, E)$ contains a minimal element $r \in R$ and also contains an edge (r, p) for some p. Argue that the system can make progress.

(1 mark)

c. Suppose that G is a directed graph with a finite and non-empty set of vertices. Argue that if G is acyclic then G has a minimal element. (Hint: we covered directed acyclic graphs in the lecture, which might be useful. Alternatively, suggest a simple method that is guaranteed to find a minimal element and argue why.)

(1 mark)

d. For a directed graph G, we consider connectedness and connected components by ignoring the direction of any edges and taking the usual definitions for undirected graphs. Argue that if a resource graph is connected and acyclic then the system can progress. (Hint, there are two cases!)

(1 mark)

e. Argue that if G is an acyclic graph with a finite and non-empty set of vertices then *every* connected component of G contains a minimal element. (Hint: what happens when we consider each component? What happens when we put them back together?)

(1 mark)

f. Suppose that G is an acycle resource graph. Argue that the system can progress. (Hint: you need to consider the case that connected components may be isolated vertices. What happens if all minimal elements in R are isolated?)

(1 mark)

g. The above line of thought shows that being acyclic is sufficient for the system to make progress. Is being acyclic a *necessary* condition for the system being able to make progress? If yes, give a proof. If no, give a counterexample.

(1 mark)

3 Finite state automata

a. A modern computer is in fact a finite state machine, since there are a finite number of states that it can have. Suppose that a computer has a total of 4 gigabytes of memory (i.e. 2^{35} bits). How many possible states does the computer have? Give you answer in the form 2^n for some n.

(1 mark)

b. Construct a finite state automaton that recognises the language given by the regular expression `a+baa*b`. Give your answer as a state change diagram.

(2 marks)

c. State labels do not need to be like S_0 and can in fact be anything you like. This can be quite useful for keeping track of the meaning of states if there are a lot of them. In some sense, in modern computers the state labels are given by the bit string which is the contents of the memory, and the operations that the computer does are operations on the state labels.

Construct a finite state automaton that recognises all strings over the alphabet $\{a, b, c\}$ that have at least one a, at least one b, and at least one c. Give your answer as a state change diagram. Give your states meaningful labels.

(2 marks)

d. Write a regular expression that matches exactly the strings over the alphabet $\{a, b, c\}$ that have at least one a, at least one one b and end with a c. (Hint: you may wish to consider two cases: a comes before b and b comes before a.)

(2 marks)

e. Suppose you are given a finite state automaton in the form of a state change diagram. Explain, using graph theoretic terminology, how to find the minimum length input that the automaton accepts.

(1 mark)

4 Homogeneous coordinates

Recall that an affine function is of the form $f(\vec{x}) = \mathbf{M}\vec{x} + \vec{t}$ for a matrix $\mathbf{M}$ and vector $\vec{t}$. Homogeneous coordinates are frequently used to represent affine functions in robotics and 3D graphics. We define the function H by

$$H\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

and if $f(\vec{x}) = \mathbf{M}\vec{x} + \vec{t}$ where

$$\mathbf{M} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \vec{t} = \begin{pmatrix} e \\ f \end{pmatrix}$$

then

$$H(f) := \begin{pmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{pmatrix}.$$

a. Some vectors are valid homogeneous representations of vectors, and some are not. Explain how to tell if some vector $\vec{y'}$ is the homogeneous representation of some other vector $\vec{y}$.

(1 mark)

b. Some matrices are valid homogeneous representations of affine transformations, and some are not. Explain how to tell if some matrix $\vec{M'}$ is the homogeneous representation of some affine transformation $f(\vec{x}) = M\vec{x} + \vec{t}$.

(1 mark)

c. Verify that the homogeneous representation faithfully represents vectors and affine transformations. To do this, suppose you are given f and $\vec{x}$ where $f(\vec{x}) = \mathbf{M}\vec{x} + \vec{t}$ and

$$\mathbf{M} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \vec{t} = \begin{pmatrix} e \\ f \end{pmatrix}, \vec{x} = \begin{pmatrix} x \\ y \end{pmatrix}.$$

i. Calculate $\vec{y} = f(\vec{x})$ in terms of x, y, a, b, c, d, e, f as a 2D vector

(1 mark)

ii. Calculate $\vec{y'} = H(f)H(\vec{x})$ using the definition of H above.

(1 mark)

iii. Verify that $H(\vec{y}) = \vec{y'}$ by calculating the left hand side.

(1 mark)

5 Matrix multiplication

a. Recall that if the determinant of a matrix $\mathbf{M}$ (denoted $|\mathbf{M}|$) is not zero, then $\mathbf{M}$ is invertable. Use the fact that $|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}|$ to show that if $\mathbf{A}$ and $\mathbf{B}$ are both invertable matrices, then so is $\mathbf{AB}$.

(1 mark)

b. Recall that the identity matrix $\mathbf{I}$ for dimension d is the $d \times d$ matrix that has $\mathbf{I}_{i,j} = 1$ if $i = j$ and $\mathbf{I}_{i,j} = 0$ for $i \neq j$. For $d = 2$ show that $\mathbf{IM} = \mathbf{MI} = \mathbf{M}$ for any matrix $\mathbf{M}$ by setting

$$\mathbf{M} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

and calculating $\mathbf{MI}$ and $\mathbf{IM}$. You must show your work.

(1 mark)

c. Verify that $\mathbf{I}\vec{x} = \vec{x}$ for dimension 2 by setting

$$\vec{x} = \begin{pmatrix} a \\ b \end{pmatrix}$$

and calculating $\mathbf{I}\vec{x}$. You must show your work.

(1 mark)

6 Applications

a. Let us invent a new game, called *Six degrees of Tim Berners-Lee*, which works as follows. You are given a Twitter handle — say `@cab203` — which we will call h_1. You must find Twitter handles $h_2, \ldots, h_6$ such that

- For each $j = 1 \ldots 5$ twitter handles h_j and h_{j+1} both appear on some accessible webpage
- $h_6 =$ `@timberners_lee`

Note that this has nothing to do with following each other on Twitter. The handles just have to appear on some accessible web page.

The questions below are concerned with building a program that plays the game. (To be clear, you will not have to play this game or build any programs!)

i. Discuss a reasonable use of regular expressions in building such a program.

(1 mark)

ii. Discuss a reasonable use of graphs in building such a program.

(1 mark)

iii. Discuss a graph theoretic problem that relates to building such a program.

(1 mark)

iv. Would you use breadth-first or depth-first traversal in solving the above problem? What are the reasons for your choice?

(1 mark)

Bonus point for anyone who actually writes such a program (working)! To be clear, you can achieve full marks without doing so, but if you want to program something, you can get an extra 1% on your final mark by doing so. Email `cab203@qut.edu.au` with some reasonable form of proof to claim your bonus.

Total marks: 30