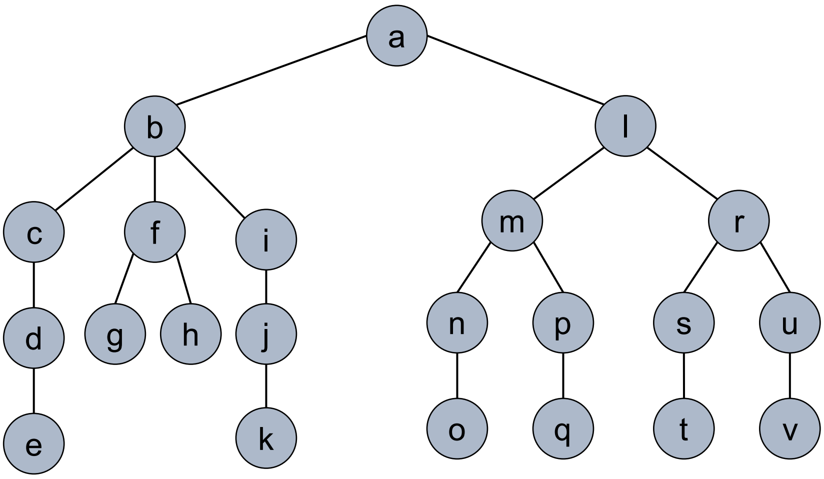
INF 4130 Exercise set 4

Exercise 1

List the order in which we extract the nodes from the Live Set queue when we do a breadth first search of the following graph (tree) with the Live Set implemented as a LIFO (“last in, first out”) queue.



Exercise 2

Solve exercise 23.6 from the text book (B&P). (See the course web page for a downloadable scan of the book.)

Exercise 3

Solve exercise 23.7 from the text book.

Exercise 4

Solve exercise 23.8 from the text book.

Exercise 5

Is your answer regarding monotonicity in 23.7 also valid if we allow moving the hole diagonally?

Exercise 6

Is it possible to use the actual cost as our heuristic (it is after all 100% exact and should be good)? Will the actual cost always be monotone? Will we expand a smaller tree? What would the problem be, if any?

Exercise 7

Show that the straight line (actually the circumference of a great circle, but let’s not get into details) between a point and the goal point is a monotone heuristic for finding the shortest path the way it is done in chapter 23.3.3 (page 728).

Exercise 8

Assign *g*-, *h*- and *f*-values to the states in figure 23.7 (page 727) and check that we actually avoid expanding the full breadth-first-tree in figure 23.3 (page 719).

Exercise 9

Adjust the DFS procedure below to instead do iterative deepening with one extra level at a time. You should only check once for each node whether it is a goal node, and you need an extra parameter to the procedure DFS. Show how the procedure should be called from a “main” program/procedure for the whole thing to work properly

```
proc DFS(v) {
   if <v is a goal node> then return “…”
   v.visited = TRUE
      for <each neighbor w of v> do
         if not w.visited then DFS(w)
      od
}
```

Exercise 10

Study the example on slide 22 from September 20 (page 723 on the textbook) to confirm that when *h*(*v*) is not monotone, then nodes sometimes will have to be taken back from tree to the priority queue, thus increasing execution time.

Exercise 11

Study the A*-algorithm described (textually) on slide 27 from the lecture.

[end]