CSCE 476/876 Spring 2004
Recitation Exercises #2

Yaling Zheng
Constraint Systems Laboratory
University of Nebraska-Lincoln
yzheng@cse.unl.edu

February 26, 2004

Some exercises were taken from AIMA or borrowed from colleagues.

1 Problem formulation

Give the initial state, goal test, successor function, and cost function for each of the following. Choose a formulation that is precise enough to be implemented.

1. You have to color a planar map using only 4 colors, with no two adjacent regions having the same color.
2. A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. He would like to get the bananas. The room contains two stackable, movable, climable 3-foot high crates.
3. You have three jugs measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet.. You need to measure out exactly one gallon.

2 Problem formulation: n-Queen problem

Give three different incremental-state formulations of n-queen problem.

3 Heuristic search: AIMA, Exercise 4.2, Page 134

4 Heuristic search: searching a specified graph

Assume you have the following search space:

State	Next-state	Cost
A	B	4
A	C	1
B	D	3
B	E	8
C	D	2
C	F	6
D	C	2
D	E	4
E	G	2
F	G	8

1. Represent the state space of this problem as a directed graph in which the states are represented as nodes and actions/operators as directed edges labeled by their cost.

2. Assume that the initial state is A and the goal state is G. Use each of the following search strategies *with loop control*, to find a path from the initial state to the goal state.

    - Breadth-first search
    - Depth-first search
    - Uniform cost search
    - Iterative deepening search

5 Lisp: Basic search strategies in random graphs

- Write a function `(graph-generator number-of-nodes number-of-edges)` that returns a connected graph `G` with `number-of-nodes` nodes and `number-of-edges` vertices. Also every edge in the graph has a positive integer weight.

- Write a function `(breadth-first-traverse u G)` that gives the vertices visiting sequence using breadth first traverse strategy(with loop control) to a random generated connected graph `G`.
- Write a function `(uniform-search u G)` that gives the nodes visiting sequence from vertex u using uniform search strategy(with loop control) to a random generated connected graph `G`.

6 Lisp: Basic search strategies in random trees

- Write a function `(tree-generator N)` that returns a tree with `N` vertices. Note that a tree has a specified root; each node except root has its only parent; each node except leaves has one or more children.
- Write a function `(depth-first-traverse T)` that gives the nodes visiting sequence using depth first traverse strategy to a random generated tree `T`.

7 Lisp: A special set-difference

Amy Davis, a form graduate student in CSE, need to implement the following function for her experiments.

The function should remove some specified elements of a set from a partition of the set. It takes two lists, a list of atoms `lost` and a partition of the set `partition` (with `lost` $\subseteq \bigcup_{S \in partition} S$). It returns a new list obtained by removing, in the elements of `partition`, the elements of `lost`.

Example: Given

```
        (setf lost '(a b c d))
        (setf partition '((a b f) (c) (h d j k))
        (amys-set-difference partition lost)
```

the function returns `((f) (h j k))`.

You are asked to implement the function `amys-set-difference` in two manners:

1. iteratively, and
2. recursively.

Hint: check the definition of `remove-if`, which may be useful at some point.

8 Lisp: Point, Line, Polygon

The goal here is to write the basic functions that would allow us to solve the path-planning problem of Exercise 3.15, Figure 3.22 in AIMA2e.

- Write `defstructures` for points and line segments in two dimensions.
- Write a function `(distance p1 p2)` that returns the distance between two points.
- Write a function `(midpoint l)` that returns the midpoint of a line segment.
- Write a function `(intersectp l1 l2)` that decides if two line segments intersect.

  Hint: the location of an arbitrary point on AB can be written as $vA + (1 - v)B$; a point on CD is $wC + (1 - w)D$. (Consider any point M in line AB. If we denote $A(x_A, y_A)$, $B(x_B, y_B)$, $M(x_M, y_M)$, and $\frac{\overrightarrow{BM}}{\overrightarrow{BA}} = v$, then we know $x_M = x_B + v(x_A - x_B) = vx_A + (1 - v)x_B$. similarly, we get $y_M = vy_A + (1 - v)y_B$. So the location of an arbitrary point on AB can be written as $vA + (1 - v)B$).

  This gives two equations (equating both x and y parts) for v and w. Solve these to find the intersection point, if any. Then you need to check that the intersection is actually on both segments - this can be determined by looking at the values of v and w.

- A scene is a list of polygons, and a polygon is a list of points. Write a method `(visiblep scene p1 p2)` that checks if one point is visible from another in a scene (that is, there is a straight line from one to the other not intersecting any polygon). Note that one vertex on a polygon should be visible from its immediate neighbors on the same polygon.
- Create a scene full of polygons by instantiating your data types, using the following coordinate data:

```
;;; Geometric scene specified as a list of polygons,
;;; each specified as a list of (x y) coordinate pairs
(((220 616) (220 666) (251 670) (272 647))
  ((341 655) (359 667) (374 651) (366 577))
  ((311 530) (311 559) (339 578) (361 560) (361 528) (336 516))
```

```
    ((105 628) (151 670) (180 629) (156 577) (113 587))
    ((118 517) (245 517) (245 557) (118 557))
    ((280 583) (333 583) (333 665) (280 665))
    ((252 594) (290 562) (264 538))
    ((198 635) (217 574) (182 574)))
```

Check to make sure that your `visiblep` function works.

Now, solve Exercise 3.15, page 91, in AIMA2e.