
Midterm Solutions

October 31st, 2005

<table>
<thead>
<tr>
<th>Name</th>
<th>E-mail</th>
</tr>
</thead>
</table>

Note: Budget your time wisely. Some parts of this quiz could take you much longer than others. Move on if you are stuck and return to the problem later.

<table>
<thead>
<tr>
<th>Problem Number</th>
<th>Max</th>
<th>Score</th>
<th>Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>120</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 Search (40 points)

Part 1.A Uninformed Search (15 points)

Consider the problem of navigating a mouse through a pentagonal-star shaped maze, shown below. The mouse is located at node S and the cheese is located at node G.

Part 1.A.1 Mouse Strategy for Maze Search (7 points)

The poor mouse unfortunately has a bad memory and cannot remember where he has been to already. From any one node, he can look in all directions and see the adjacent nodes. As any intelligent mouse, he orders them alphabetically. After looking around, he always takes the path to the neighboring node that appears earliest in the alphabet (e.g., from node S, he would go to A instead of B). Since there are no dead ends, he never backtracks.

List the first five steps of the path taken by the mouse. At each step list the node the mouse is at and the adjacent nodes seen.

<table>
<thead>
<tr>
<th>Step</th>
<th>At Node</th>
<th>Sees Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>S</td>
<td>ABCD</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>BEHS</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>ACES</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>BEHS</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>ACES</td>
</tr>
</tbody>
</table>

Does the mouse get to the cheese using this strategy? Why or why not?

No. He will loop infinitely between B and A.
What search method or methods is the mouse’s search strategy most similar to?

This is essentially a DFS without a visited list or backtracking.
It is also similar to greedy search whose cost is alphabetical position of the neighbor

Part 1.A.2 Breadth-first Maze Search (8 points)

Using a different approach, the mouse decides to leave a little excrement at each node he goes to, so that he can keep track of where he has gone. He now decides to use these marks to implement a breadth first approach, when going through the maze.

Model his behavior using breadth-first search, with a visited list and a queue. Keep track of the visited list and elements taken off and put on the queue at each step (it is not necessary to rewrite the whole queue at each step unless you find it helpful). Stop when a path to G is found.

<table>
<thead>
<tr>
<th>Step</th>
<th>Dequeued</th>
<th>Enqueued</th>
<th>Visited List</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(S)</td>
<td>(AS)(BS)(CS)(DS)</td>
<td>(S A B C D)</td>
</tr>
<tr>
<td>1</td>
<td>(AS)</td>
<td>+ (EAS)(HAS)</td>
<td>+ E H</td>
</tr>
<tr>
<td>2</td>
<td>(BS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(CS)</td>
<td>+ (FCS)</td>
<td>+ F</td>
</tr>
<tr>
<td>4</td>
<td>(DS)</td>
<td>+ (IDS)</td>
<td>+ I</td>
</tr>
<tr>
<td>5</td>
<td>(EAS)</td>
<td>(GEAS)</td>
<td>+ G</td>
</tr>
<tr>
<td>6</td>
<td>(HAS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>(FCS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(IDS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>(GEAS)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part 1.B Informed Search (25 points)

The mouse finds and eats the cheese at G. Now we consider a new piece of cheese at node J, as shown below. Note that J is closer to B and C than it is to E and F. The mouse becomes smarter and uses his sense of smell to guide him. He uses the fact that the cheese smells stronger when he is physically closer to it, to estimate his straight-line distance to the goal. He tries out three search methods: greedy, uniform cost, and A*.

Demonstrate each search, by writing out the full queue for the first three steps of each search. Lengths of edges in the graph are given above, to the right; these symbolic values follow from simple geometry. Remember that each queue element should include an estimated cost. Give the full queue at each step, not just the newly enqueued elements.

Part 1.B.1 Greedy Maze Search (6 points)

<table>
<thead>
<tr>
<th>Step</th>
<th>Dequeue</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(σ S)</td>
<td>(β BS)(β CS)(α AS)(α DS)</td>
</tr>
<tr>
<td>1</td>
<td>(β BS)</td>
<td>(β CS)(β CBS)(ε EBS)(α AS)(α ABS)(α DS)</td>
</tr>
<tr>
<td>2</td>
<td>(β CS)</td>
<td>(β CBS)(ε EBS)(ε FCS)(α AS)(α ABS)(α DS)(α DCS)</td>
</tr>
</tbody>
</table>

Part 1.B.2 Uniform Cost Maze Search (6 points)

<table>
<thead>
<tr>
<th>Step</th>
<th>Dequeue</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0 S)</td>
<td>(b BS)(b CS)(a AS)(a DS)</td>
</tr>
<tr>
<td>1</td>
<td>(b BS)</td>
<td>(b CS)(a AS)(a CBS)(a DS)(a EBS)(2b ABS)</td>
</tr>
<tr>
<td>2</td>
<td>(b CS)</td>
<td>(a AS)(a CBS)(a DS)(a EBS)(a FCS)(2b ABS)(2b DCS)</td>
</tr>
</tbody>
</table>
Part 1.B.3 A* Maze Search (6 points)

<table>
<thead>
<tr>
<th>Step</th>
<th>Dequeue</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(σ S)</td>
<td>(b+β BS)(b+β CS)(a+α AS)(a+α DS)</td>
</tr>
<tr>
<td>1</td>
<td>(b+β BS)</td>
<td>(b+β CS)(a+β CBS)(a+ε EBS)(a+α AS)(a+α DS)(2b+α ABS)</td>
</tr>
<tr>
<td>2</td>
<td>(b+β CS)</td>
<td>(a+β CBS)(a+ε EBS)(a+ε FCS)(a+α AS)(a+α DS)(2b+α ABS)(2b+α DCS)</td>
</tr>
</tbody>
</table>
Part 1.B.4 Heuristics for Maze Search (7 points)

Now the mouse decides to use a different heuristic: “Count the smallest number of line segments on a path to the goal.” Under what conditions is this heuristic admissible?

Heuristic is admissible if all line segment lengths are greater than one. That is, \( b, g \geq 1 \).

What method, of those learned in class, can you use to compute this estimate?

Branch and Bound or Breadth first Search, (or A* or single source shortest path).

Is this a time efficient way of estimating cost? Justify your answer.

No. The worst case cost of, for example, breadth-first search is:
\[
\frac{b^{(d+1)} - 1}{b-1}
\]
where \( d \) is the shortest number of links, and \( b \) is the branching factor. This is exponential in the number of links.
Problem 2 Constraints (40 points)

Part A Visual Interpretation (10 points)

Recall that the goal of edge detection is to abstract away from a highly detailed image a more compact, abstract representation. The motivation for this is that edge contours in an image correspond to important scene contours. A line drawing is one such abstract representation of an image. Recall that our line drawings assume a world wherein objects have no surface marks, shadow edges, or specularities. As a pre-processing step for object recognition, line interpretation classifies each edge as convex, concave, or occluding. For occluding edges, line interpretation identifies which of the two surfaces bordering the curve in the line drawing is closer in the scene. These classifications can be represented by giving each line one of four possible line labels.

Briefly define each line label in the following table:

<table>
<thead>
<tr>
<th>Line Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Represents a convex edge. This is associated with a surface normal discontinuity in which both surfaces that meet along the edge are visible, and the two surfaces create a visible angle &gt; 180 degrees.</td>
</tr>
<tr>
<td>-</td>
<td>Represents a concave edge. This is associated with a surface normal discontinuity in which both surfaces that meet along the edge are visible, and the two surfaces create a visible angle &lt; 180 degrees.</td>
</tr>
<tr>
<td>← or →</td>
<td>Represents an occluding convex edge. When viewed from the camera, both surface patches that meet along the edge lie on the same side, one occluding the other. As one moves in the direction of the arrow, the surfaces are to the right.</td>
</tr>
</tbody>
</table>

Illustrate these labels by correctly labeling every edge of the object below with one of these four labels. We have labeled one edge for you:
Part B Solving A Crossword as a Constraint Program (20 points)

Consider the following CSP representation of a crossword puzzle problem:

<table>
<thead>
<tr>
<th>Variables:</th>
<th>Domains:</th>
<th>Constraints:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 across</td>
<td>{ laser, still, super }</td>
<td>C1.) 1 across,letter3 = 2 down,letter1</td>
</tr>
<tr>
<td>4 across</td>
<td>{ calm, corn, fort }</td>
<td>C2.) 1 across,letter5 = 3 down,letter1</td>
</tr>
<tr>
<td>2 down</td>
<td>{ bean, iron, same }</td>
<td>C3.) 2 down,letter3 = 4 across,letter2</td>
</tr>
<tr>
<td>3 down</td>
<td>{ lab, let, run }</td>
<td>C4.) 3 down,letter3 = 4 across,letter4</td>
</tr>
</tbody>
</table>

Each variable corresponds to a word in the puzzle, \{ 1 across, 4 across, 2 down, 3 down \}. The domain of each variable is a list of candidate words, and the constraints are imposed by shared squares between the variables.

Part 2.B.i Constraint Propagation (6 points)

The constraint graph is provided below. On the graph, cross out each domain element that is eliminated by constraint propagation (using the AC-3 algorithm). For example, to start you off, we have eliminated *super* from the variable 1 across, which is eliminated using constraint C1.
Part 2.B.ii Backtrack Search with Forward Checking (14 points)

In this part you will use *backtrack search with forward checking* to search for the *first consistent solution* to the crossword puzzle above. Your task is to draw the search tree and to indicate which domain values are eliminated by forward checking at each search step. In your search, variables are to be assigned in *numerical order*, and values are to be assigned in *alphabetical order*. At each search step, make *only one variable assignment*, and then perform *forward checking*, by crossing off any domain values that forward checking would prune by that assignment. Recall that, for each step moving down the tree, you copy into the next constraint graph only domain values that haven’t been pruned by its parent. To get you started, we have completed steps 1 and 2 for you. Finding the first consistent solution should take 10 steps or less.

The crossword puzzle and CSP representation are repeated on each page for convenience:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Variables: Domains: Constraints:

1 across { laser , still , super } C1.) 1 across,letter3 = 2 down,letter1
4 across { calm , corn , fort } C2.) 1 across,letter5 = 3 down,letter1
2 down { bean , iron , same } C3.) 2 down,letter3 = 4 across,letter2
3 down { lab , let , run } C4.) 3 down,letter3 = 4 across,letter4

**Step 1:**

1A

2D

3D

4A

---

**Step 2:**

1A = laser

2D

3D

4A

---
Variables: 1 across \{ laser, still, super \}  
4 across \{ calm, corn, fort \}  
2 down \{ bean, iron, same \}  
3 down \{ lab, let, run \}

Domains:  
1A = \{ laser \}  
2D = \{ same \}  
3D = \{ run \}  
4A = \{ fort, corn, calm \}

Constraints:  
C1.) 1 across, letter3 = 2 down, letter1  
C2.) 1 across, letter5 = 3 down, letter1  
C3.) 2 down, letter3 = 4 across, letter2  
C4.) 3 down, letter3 = 4 across, letter4

Step 3:

1A  
2D same  
3D  
4A

Step 4:

1A  
2D same  
3D  
4A

Step 5:

1A laser still  
2D same iron  
3D  
4A

Step 6:

1A laser still  
2D same iron lab  
3D  
4A
Variables: Domains: Constraints:
1 across { laser, still, super } C1.) 1 across, letter3 = 2 down, letter1
4 across { calm, corn, fort } C2.) 1 across, letter5 = 3 down, letter1
2 down { bean, iron, same } C3.) 2 down, letter3 = 4 across, letter2
3 down { lab, let, run } C4.) 3 down, letter3 = 4 across, letter4

Step 7:
1A 1A
2D 2D
3D 3D
4A 4A

Step 8:
1A 1A
2D 2D
3D 3D
4A 4A

Step 9:
1A 1A
2D 2D
3D 3D
4A 4A

Step 10:
1A 1A
2D 2D
3D 3D
4A 4A
Part 2.C Complexity of Backtrack with Forward Checking (10 points)

Derive the worst case run-time of the Backtrack Search with Forward Checking algorithm, which you employed in the previous part. Assume that there are \( v \) variables, each with a domain size of \( d \), and there are \( e \) binary constraints. Show and justify each step in your derivation. Remember that you need to consider both the number of search nodes expanded, and the combined cost of performing forward checking on each of these nodes.

Worst Case Run Time: \((v*d)^*(d)^v\)

Derivation:

In the worst case, we assume that no solution is found and the entire search tree is searched. Additionally, we assume that forward checking prunes NO domain values.

To find the worst case complexity of forward checking, we assume that at each step, the current variable assignment is connected to every other variable by a constraint edge. Thus, the number of forward checks is approximately \( v*d \) at each step. (\( d \) checks for checking the entire domain of each variable that the new variable assignment is connected to.)

In the worst-case we assume that the entire tree is searched, so the number of steps is equivalent to the complexity for backtrack search.
\[
1 + d + d^2 + \ldots + d^v = (d^v*(v+1) - 1)/(d-1) \approx d^v
\]

Therefore, the worst case of BT-FC = (worst case BT) \* (worst case FC)
Problem 3 Linear Programs (40 points)

Part 3.A Framing a Linear Program (20 points)

Kurt Procrastinator has a problem set due in the morning and he needs some help to keep awake long enough to complete the work. Kurt knows that a little espresso greatly enhances his ability to stay awake and work faster. In fact he has discovered that every milliliter increases his productivity by 1%; however, if he has more than 100 milliliters, he gets so loopy that he can’t work at all, hence 100 milliliters is a hard limit for him. Sugar also helps to boost his productivity and each gram increases his productivity by 2%. Unfortunately, adding more than 2 grams of sugar per milliliter of espresso makes the drink so sweet as to be undrinkable - another limit. Finally, adding less than 1 gram of sugar per 10 milliliters of espresso makes the drink too bitter for him to drink.

Part 3.A.i Framing a Linear Program (10 points)

In the box below, specify a linear program that finds Kurt’s ideal coffee concoction, that is, it should determine the optimal amount of espresso and sugar in Kurt’s coffee.

Maximize \[ z = e + 2s \]

Subject to
\[ 2e - s \geq 0, \]
\[ e - 10s \leq 0, \]
\[ e \leq 100, \]
\[ e \geq 0, s \geq 0 \]

Part 3.A.ii Graphical Depiction of your Linear Program (10 points)

In the box below, graph the LP that you specify in Part 3.A.i. Label the corners of the polytope by their co-ordinates, and identify which corner represents the optimal potion.
Part 3.B Solving a Linear Program using Simplex (20 points)

Solve the following linear programming problem using Simplex, while showing all Simplex steps. You may use either algebraic or Tableau form in your solution. Explain your steps in solving the problem including any transformations that you make to the problem.

Maximize  \( z = x_1 + 2x_2 + 3x_3 \)

Subject to
\[
\begin{align*}
& x_1 + x_2 + x_3 \leq 100 \\
& x_1 + x_2 \leq 70 \\
& x_2 + x_3 \leq 60 \\
& x_1 + x_3 \geq 30 \\
& x_1 \geq -5, \ x_2 \geq 0, \ x_3 \geq 0
\end{align*}
\]

Replace \( x_1 \) with \( x_1' \) where \( x_1' = x_1 + 5 \) to deal with \( x_1 \) being allowed to be \(-5\).
Replace \( z \) with \( z' \) \( z' = z + 5 \)

Maximize  \( z' = x_1' + 2x_2 + 3x_3 \)

Subject to
\[
\begin{align*}
& x_1' + x_2 + x_3 \leq 105 \\
& x_1' + x_2 \leq 75 \\
& x_2 + x_3 \leq 60 \\
& x_1' + x_3 \geq 35 \\
& x_1' \geq 0, \ x_2 \geq 0 \ x_3 \geq 0
\end{align*}
\]

Step 1: Add slack/artificial variables
\[
\begin{align*}
1: & \quad z' - x_1' - 2x_2 - 3x_3 + Mz_1 = 0 \\
2: & \quad x_1' + x_2 + x_3 + y_1 = 105 \\
3: & \quad x_1' + x_2 + y_2 = 75 \\
4: & \quad x_2 + x_3 + y_3 = 60 \\
5: & \quad x_1' + x_3 - y_4 + z_1 = 35
\end{align*}
\]

Step 3: Row 1 – M*Row5
\[
\begin{align*}
1: & \quad z' - (M+1)x_1' - 2x_2 - (M+3)x_3 + My_4 = -35M \\
2: & \quad x_1' + x_2 + x_3 + y_1 = 105 \\
3: & \quad x_1' + x_2 + y_2 = 75 \\
4: & \quad x_2 + x_3 + y_3 = 60 \\
5: & \quad x_1' + x_3 - y_4 + z_1 = 35
\end{align*}
\]

Step 3: Obtain initial basic feasible solution
\[
<0 \ 0 \ 0 \ 105 \ 75 \ 60 \ 0 \ 35> \quad z' = -35M
\]

Select \( x_3 \) as the basic entering variable.
2: 105, 4: 60, 5: 35. So select 5
Remove x3 from all constraints

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>$z’ + 2x1’ - 2x2 + (M + 4)y4 = 105$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td>$x2 + y1 + y4 - z1 = 70$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:</td>
<td>$x1’ + x2 + y2 = 75$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:</td>
<td>$-x1’ + x2 + y3 + y4 - z1 = 25$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:</td>
<td>$x1’ + x3 + x3 - y4 + z1 = 35$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Select x2 as basic entering variable.
2: 70, 3: 75, 4: 25 So select 4

Remove x2 from all constraints

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>$z’ + 2y3 + (M + 6)y4 - 2z1 = 155$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td>$x1’ + y1 - y3 = 45$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:</td>
<td>$2x1’ + y2 - y3 - y4 + z1 = 50$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:</td>
<td>$-x1’ + x2 + y3 + y4 - z1 = 25$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:</td>
<td>$x1’ + x3 + x3 - y4 + z1 = 35$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Select z1 as the basic entering variable
3: 50, 4: -25, 5: 35 Select 5

Remove z1 from all constraints

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>$z’ + 2x1’ + 2x3 + 2y3 + (M + 4)y4 = 225$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td>$x1’ + y1 - y3 = 45$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:</td>
<td>$x1’ + y2 - y3 = 15$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:</td>
<td>$x2 + x3 + y3 = 60$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:</td>
<td>$x1’ + x3 - y4 + z1 = 35$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$z’ = 225, x1’ = 45, x2 = 0, x3 = 60,$

So $x1 = x1’ - 5 = 40$
and $z = z’ - 5 = 220$

**Hence $z = 220, x1 = 40, x2 = 0, x3 = 60$**

*Obviously the tableau version of the above would also be acceptable as would the two phase method.*