General

Read through the “Background” section below, then copy and paste the questions out of the “Assignment” section into your word processor and answer the questions. Turn in the questions using the instructions posted on the class web site.

At the top of the every document that you create (word processing or source files) include:

# **// Your name**

**// CS-160, Lab #X** (replace the X with the Lab #)

**// xxxx Term, 20xx** (i.e. Fall Term, 2008)

For ALL Word processing documents, you must submit your documents in one of the following formats: MS-Word (NOT Works), RTF (most word processors can save in this format), or Open Document (used by the freely available Open Office suite). They will be returned ungraded if submitted in any other format.

**Concepts**

This lab will introduce you to the concepts of ***data structures*** and ***search***. The focus is on abstract data types and continues your study of algorithms. We will also use the “Alice” programming environment tutorials for part of this lab for more experience with programming.

**Background**

*Note: Original source of the background contained below is from the “CS160 Worksheets” by Daniel Balls of the CS department at Oregon State University; updated and revised by Mitch Fry (CS, Chemeketa Community College).*

**Search Methods**

Some situations present problems that have many solutions, but only one that is optimal. In such a situation a searching strategy may be helpful in finding the optimal solution. To illustrate problem solving by searching, we’ll consider the 8-puzzle.

The 8-puzzle is tile game consisting of nine squares—eight numbered tiles and one ‘empty’ square (see square below, left which was adapted from Zhu, S. C., <http://www.stat.ucla.edu/~sczhu/Courses/UCLA/Stat_232B/Handouts/Introduction.pdf>.). The goal of the puzzle is to use the free space to move the tiles so that the numbers are in order starting at the top left and proceeding around the square until the 8, which is right below the 1 (see square below, right). When the numbers are in this order, we have found a solution.

|  |  |  |
| --- | --- | --- |
| **1** | **2** | **3** |
| **8** |  | **4** |
| **7** | **6** | **5** |

|  |  |  |
| --- | --- | --- |
| **2** | **8** | **3** |
| **1** | **6** | **4** |
| **7** |  | **5** |

*The puzzle on the left represents an 8-puzzle in a non-solution state, while the puzzle*

*on the right represents the solution, or objective, of an 8-puzzle.*

An optimal solution consists of one that consists of the fewest slides. To track the progress toward the optimal solution, we’ll implement a penalty rule system. Each time we slide a tile we’ll be assessed one point. Further, for each number that is out of its solution position, we will assess one penalty point. So if the 8 – puzzle we want to solve is the one on the left above, the initial position has a score of 4 points—we’ve made no slides, and there are four numbers (2, 8, 6, and 1) out of place.

First, we consider the three possible first moves and the number of points for each of the resulting states, as shown below. We’ll continue to search from the puzzle with th

e lowest point total, in this case the puzzle with 4 points.

|  |  |  |
| --- | --- | --- |
| **1** | **2** | **3** |
| **8** |  | **4** |
| **7** | **6** | **5** |

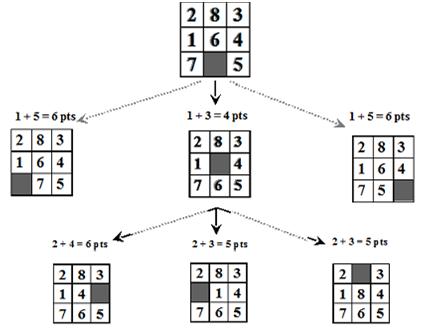


|  |  |  |
| --- | --- | --- |
| **2** | **8** | **3** |
| **1** | **6** | **4** |
| **7** |  | **5** |

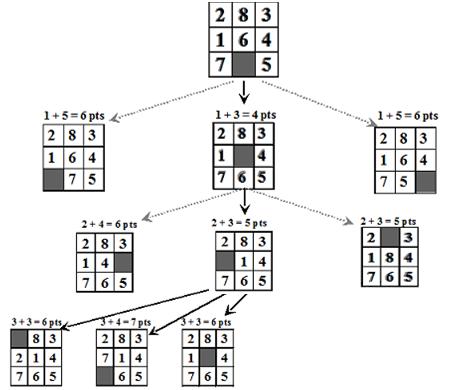
**1 + 5 = 6 pts**

**1 + 3 = 4 pts**

**1 + 5 = 6 pts**

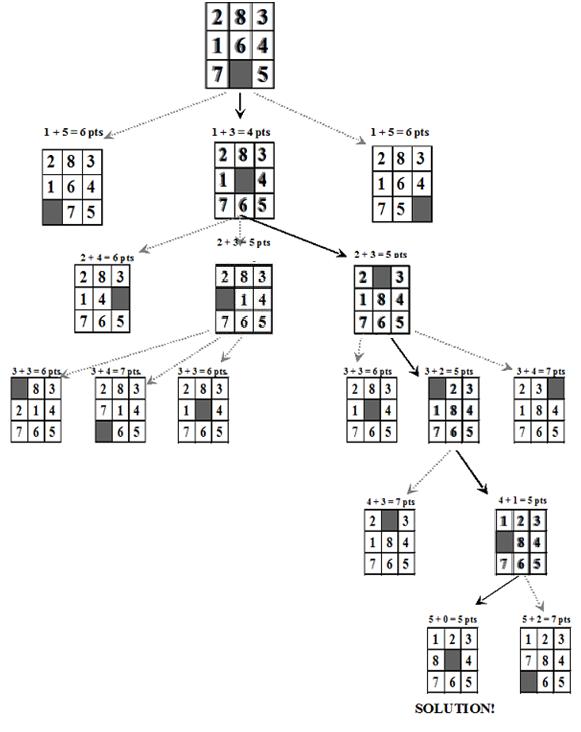
***Diagram note:*** *a shaded puzzle represents one whose ‘next steps’ have been expanded. Solid arrows represent the current ‘puzzle path’ we are currently exploring. Dotted arrows represent paths to puzzles we have not currently chosen, but may choose in the future if they become more desirable than the current path.*

Since two puzzles tie for the lowest number of points (i.e. two of them have only five points) we’ll choose one of the two arbitrarily, the one with the empty square in the middle row.



Searching this path leads to three puzzles with point totals of six (two have this many points) and seven. At this point if we were using the greedy method, we would follow the path of one of the puzzles having six points. However, according to the search method we will take a step back and survey all possible remaining paths and then choose from them the one that is most desirable. Currently there are seven puzzles that have not yet been expanded. Each of these puzzles has five, six or seven points. There is only one puzzle with five points and we’ll expand that puzzle next, as it is currently the most desirable path.

As shown on the next page, by following this path and implementing the search strategy, the optimal solution to the puzzle can be obtained in five slides, indicated in the diagram by the solid black arrows.

******

The key principle of this searching method is always following the path that appears to be leading to the lowest-cost solution. If the algorithm is following a promising search path, but at some point the path seems to be inferior to another path through the search tree, the algorithm should switch focus and pursue the path that now seems to be better.

**Assignment Instructions**

*Answer the following questions. Please copy and paste the questions below into your editor (use the I-bar tool in Acrobat Reader to do this) and TYPE your answers below each question.*

**Part 1: Search**

***S****olve each of the following problems using a best-first search method as described in the background section. Diagram out your search trees; note that you will need to design a representation of each state in the tree that is appropriate to the problem that you are solving (Hint: on 1 & 3 use a text description for each state).* ***IMPORTANT NOTE: 2/3rds of the grade on these questions will be for showing that you understand the problem solving technique, only 1/3 of the grade is for correctness of you answer. So you MUST show your work in order to get most of the points on these questions; just listing the correct answer is only worth 1/3 of the points.***

***Q1:*** What is the minimum number of steps required to transform the eight-puzzle below into its solution state? Solve this using the method and evaluation functions shown in the background section. Show your state search tree!

|  |  |  |
| --- | --- | --- |
| **1** | **2** | **3** |
| **8** |  | **4** |
| **7** | **6** | **5** |

|  |  |  |
| --- | --- | --- |
|  | **1** | **3** |
| **6** | **2** | **4** |
| **8** | **7** | **5** |

***Q2:*** On a dark, cold and stormy night four friends find themselves on one side of a bridge. There is only one lamp between them and because the bridge is so old, at most two of them can cross the bridge at once. Whenever anyone crosses the bridge—either alone or with a partner—the lamp must go with him.

Hint on Q2 & Q3:

1) Design your state representation for the problem

2) Define the set of possible transformations that change the state

3) Define your state evaluation function

4) Construct the state search tree

One of the friends can cross the bridge in 15 minutes, the second in 10 minutes, the third in 2 minutes and the last in 1 minute. Find the minimum amount of time needed before they each reach the other side of the bridge.

Hint on a representation: Use a text description for each state; something like:

Starting side: Friend1, Friend2, Friend3, Friend4

Goal side: no one

Time so for: none

***Q3*:** Two boys with a small boat agree to help three soldiers cross a river without a bridge. The boat is so small it can support only one soldier or two boys. A soldier and a boy can't be in the boat at the same time for fear of sinking it. What is the minimum number of trips it will take to ferry all the soldiers across? (Adapted from *Science News for Kids*, http://www.sciencenewsforkids.org/pages/puzzlezone/muse/muse0904.asp).

**(continued on next page)**

**Part 2: Data Structures**

**(**See the online tutorials for the background on data structures)

***Q4:*** Show how a stack would appear after **each** of the following 16 operations:

push(t), push(t), push(y), pop(), push(g), push(u), push(o), pop(), pop(), pop(), push(o), pop(), pop(), push(i), pop(), pop()

***Q5:*** Show how a queue would appear after **each** of the following operations:

add(a), add(p), add(p), remove(), add(l), remove(), remove(), add(e), remove(), remove()

**Part 3: Introduction to Programming (Scratch)**

***Q6:*** This project will be split over the last 2 labs (Lab #8 & #9). The project is for you to create an animation, simulation, or game in Scratch that somehow demonstrates or teaches any concept that we have learned in CS160 this term. You project does not need to be elaborate, as long as some concept is demonstrated. At minimum, your project should have some graphics, some animation or motion and some sound (you could even record your own voice!). As a model of what you might do, review any of the 50 flash animations and java applets that have been used to teach concepts in this class. You are welcome to simply re-implement any one of these if you wish.

The requirement for Lab #9 is to complete the project that you started in Lab #8 (the prototype of your project). TAKE A SCREEN SHOT (alt + print screen) of your final Scratch project and paste it into your assignment document as your answer for Q6.

TURN IN your document that answers the assignment questions, and also TURN in your scratch project file (the file that ends in “.sb”).