**2013 Spring Computer Science II Final Exam**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date: 4/30/2013**

1) (10 pts) Determine the maximum value of a knapsack with weight 12 or less chosen from the following items listed below. In order to get full credit, please complete the chart below indicating what would be stored in the array used in executing the algorithm in class after each item has been processed. Please process items in the order indicated by the chart.

|  |  |  |
| --- | --- | --- |
| Item | Weight | Value |
| A | 5 | 12 |
| B | 3 | 7 |
| C | 4 | 10 |
| D | 6 | 14 |
| E | 4 | 11 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Item | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| A |  |  |  |  |  |  |  |  |  |  |  |  |
| B |  |  |  |  |  |  |  |  |  |  |  |  |
| C |  |  |  |  |  |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |  |  |  |  |  |
| E |  |  |  |  |  |  |  |  |  |  |  |  |

Maximum value of knapsack = \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Items in knapsack = { \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ }

2) (10 pts) Determine the number of ways to make change for 11 cents using 2 cent, 3 cent, 4 cent and 7 cent coins. In order to get full credit, please complete the chart below using the algorithm shown in class.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Num Cents | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Max Coin |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |

3) (20 pts) Consider the following problem:

Teams A and B are going to play a series of matches. Team A’s goal is to win a matches while Team B’s goal is to win b matches. (Both a and b are positive integers.) Thus, at most a + b – 1 matches will be played. The probability that team A wins an individual match is p and there are no ties. Determine the probability that A wins the series.

A recursive solution to the problem is as follows:

public static double recProb(int a, int b, double pA) {

 // Probabilities when one team has already won.

 if (a == 0) return 1;

 if (b == 0) return 0;

 // Break our counting into two groups:

 // (a) Situations where a wins the first game.

 // (b) Situations where b wins the first game.

 return pA\*recProb(a-1,b,pA) + (1-pA)\*recProb(a,b-1,pA);

}

(a) (10 pts) Edit this solution above to use memoization. Change the method prototype as necessary and write a segment of code separate to the method showing how an initial call to your method would look using winA, winB and pAWin as the first three parameters to the method.

Edited method

Code Segment setting up initial call to the edited method

(b) (10 pts) Now, rewrite the solution in part a to utilize dynamic programming. The method should be self-contained, not requiring any extra arrays outside of the method. Use the method prototype outlined below:

public static double getProbDP(int a, int b, int p) {

}

4) (5 pts) The array below stores a disjoint set. Draw the corresponding tree representation of that disjoint set.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| value | 4 | 0 | 5 | 6 | 4 | 5 | 5 | 6 | 0 | 2 |

5) (15 pts) Consider solving the following recurrence relation: T(n) = 2T(n/2) + n2, T(1) = 1.

(a) (12 pts) Use the iteration technique to determine a Big-O bound for T(n), in terms of n.

(b) (3 pts) Use the Master Theorem to verify your work from part (a).

6) (10 pts) Execute Prim’s Algorithm to find the Minimum Spanning Tree of the graph whose edges are given below, ***starting at vertex C.*** Please list each edge considered. If an edge is not placed in the MST, list the cycle that would be formed by adding that edge. (Note: assume the edges listed below are undirected, so an edge from A to B is also an edge from B to A.)

AB = 3 AF = 17 AG = 8 AH = 12

BD = 7 BE = 4 BG = 5

CD = 9 CF = 13 CH = 11

DE = 14 DF = 8 DG = 9 DH = 6

EG = 10 EH = 5

FH = 15

|  |  |  |
| --- | --- | --- |
| Edge Considered | Added? (Yes/No) | If no, cycle formed |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

5) (10 pts) Show the result of inserting the following items in a 2-4 tree in the order indicated: 18, 13, 77, 19, 2, 6, 8, 45, 20, and 33. Show the state of the tree right before each structural change, as well as the final tree.

8) (10 pts) Consider writing your own ArrayList class which dynamically sizes the internally stored array as follows:

(a) Initial size is 100, with each element pointing to null.

(b) When an element needs to be added and the array is full with n elements, a reallocation is done which adds √n elements. In all cases, the element is added to the end of the list.

(c) Whenever an element is deleted, no elements are moved in the list. Rather, that particular spot is left to point to null.

(d) After every √n delete operations, where n represents the current length of the ArrayList, “coalesce” operation is run which resizes the array so no null elements exist.

Note: There are many undesirable properties of the data structure above, but I’ve created it to simplify the analysis posed.

Consider an ArrayList implemented as described above that has filled to contain exactly n elements, where none of them are null. Consider the next n operations (inserts and deletes). What is the amortized worst-case run time of each of these operations? (Remember, amortized worst cost simply means taking the worst case time of all the operations and dividing it by the number of operations.)

9) (5 pts) What is the fewest number of comparisons necessary to sort an array of size 9? You may use the fact that 9! = 362880 and that 216 = 65536.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10) (5 pts) In which city is Seattle’s Best Coffee based? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Scratch Page – Please clearly mark any work on this page you would like graded.**