Sample Problems: Test 1

1.Given a clearly defined grid of items (as in the first three problems of homework 0, with the number of rows and columns specified), or some variation, can you answer the following questions?

a. Give a formula for the number of elements in row i

b. Give the formula for the total number of elements.

c. Give a formula for the number of elements preceding row i

d. Noting that an element is not defined for all combinations of i and j, what relationship must hold for the element to be

defined?

e. Assuming the items are stored in a single-dimensional array B, in the sequence from top to bottom, the formula to find the position of the element in row i, column j, when the element is defined.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2. Write a recursive function that returns the value of the following recursive definition:

F(X, Y) = X - Y, if X or Y < 0

F(X, Y) = F(X-1, Y) + F(X, Y-1), otherwise.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3. Suppose W satisfies the following recurrence relation and boundary condition

(where c is a constant):

W (n) = c n + W ( n /2 )

W (1) = 1.

What is the order of W ?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4. Exercise 2.3-3 (page 39).

[You may answer by using k as the induction parameter. Thus, you answer by first verifying the base case, for

n = 2^1 = 2, and then assuming it is true for n=2^m, prove that it holds for n = 2^(m+1).]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5. Design a good algorithm for the following game, based on a divide-and-conquer strategy:

The user thinks of a positive integer number in the range 1 to n. The program (takes n as a parameter and) attempts to find the number by asking questions of the form “is the number less than (greater than) x?” The object is to ask as few questions as possible. What is the number of questions that may have to be asked in the worst case (θ)?

HINT: This is a based on an efficient (and popular) search algorithm, on which you have done some homework.

Sample Questions: Test 2

1. Compare Dynamic Programming: vs. Divide and Conquer or vs. Greedy algorithms.

2. Identify when the three cases of the Master Theorem apply and be able to solve a

recurrence relation, or two.

3. Analyze the complexity of a given algorithm, and show the results using order notation.

4. Using the idea of quicksort, give an efficient algorithm, without completely sorting the

data, to identify the kth smallest (or kth largest) of n numbers. (This is called the

“quickselect” algorithm and is based on an idea similar to the one used in homework 3,

problem 1.)

4. Consider the following problem (supplied at test time). Decide if the Principle of

Optimality (Optimal Substructure) applies. Explain why/why not.

Sample Questions: Test 3 FINAL

1. Compare and contrast dynamic programming and the greedy approach. Be able to explain

why a particular greedy choice works (to globally optimize) in a given problem.

2. Be able to apply greedy activity selection, 0-1 knapsack, fractional knapsack, Huffman

codes, Kruskal, Prim, and Dijkstra’s algorithms (as you did in homework assignments).

3. Graph representation (adjacency list and adjacency matrix); the color, distance, and

predecessors in BFS; the color, predecessor, discovery and finish times in DFS; properties of

DFS and the parenthesis theorem; edge types in a depth-first forest; topological sorting and strongly connected components; the notions of a minimum spanning tree (MST) and the shortest

path.

4. Complexity of all the algorithms we discussed (including the various graph algorithms)

5. Describe the sets P, NP, NPC, and their relationship.

6. Optimization vs. decision problems and their correspondence: Specify a decision problem that corresponds to an optimization problem.

7. What is polynomial reducibility; how is it uses in establishing polynomial time decision

algorithms, and in establishing NP-completeness of a new decision problem?

8. Give examples of NP-complete problems, and a hierarchy of them in terms of reducibility.