CSC 464/564 Homework #6 Instructor: Jeff Ward

Assigned: Thursday, October 11, 2018 Due: 11:59pm, Wednesday, October 24

Covers: Greedy Algorithms Worth 25 points

Problem 1 (Prim’s and Kruskal’s algorithms – 10 points) Suppose we want to the find a minimum spanning tree of the following graph:

4

9

3

2

6

5

8

7

1

1. Run Prim’s algorithm starting from node A. Fill in the following table showing the intermediate values of the cost array.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | initially | dequeue A | dequeue | dequeue | dequeue | dequeue | dequeue | dequeue |
| A | 0,nil |  |  |  |  |  |  |  |
| B | ∞,nil |  |  |  |  |  |  |  |
| C | ∞,nil |  |  |  |  |  |  |  |
| D | ∞,nil |  |  |  |  |  |  |  |
| E | ∞,nil |  |  |  |  |  |  |  |
| F | ∞,nil |  |  |  |  |  |  |  |
| G | ∞,nil |  |  |  |  |  |  |  |

Draw the resulting minimum spanning tree.

1. Run Kruskal’s algorithm on the same graph. First, show your sorted list of the edges. As the algorithm proceeds, show how the disjoint-sets data structure looks at every intermediate stage (includes both the current rank values and the current parent pointers). Do not perform path compression. The nodes of the trees are given below.

List of edges:

Initially:

Add edge \_\_\_\_\_:

Add edge \_\_\_\_\_:

Add edge \_\_\_\_\_:

Add edge \_\_\_\_\_:

Add edge \_\_\_\_\_:

Add edge \_\_\_\_\_:

2. (Kruskal’s algorithm – 3 points) Suppose that we are running Kruskal’s algorithm on a weighted, undirected graph and that the following directed tree exists in the supporting data structure (assume F has a loop pointing back to F):

Which node is returned by find(I)?

Draw the tree structure that results from performing find(I). Use the version of find that does path compression.

3. (Horn formulas – 6 points)

1. Trace the algorithm from Section 5.3 on the following Horn formula, indicating which variables are set to true in which order:

a ∧ b → d

d ∧ e → g

→ e

e → a

→ b

¬d ∨ ¬g

Algorithm trace:

Is the formula satisfiable?

If it is satisfiable, what is the minimum satisfying assignment?

1. Repeat the above problem for the Horn formula:

a ∧ b → d

d ∧ e → g

→ e

a → e

→ b

¬d ∨ ¬g

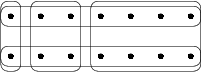
Algorithm trace:

Is the formula satisfiable?

If it is satisfiable, what is the minimum satisfying assignment?

4. (Set cover – 6 points)

Consider the following diagram taken from http://en.wikipedia.org/wiki/Set\_cover\_problem:



If we label the dots along the top row as a1,…,a7 and the dots along the bottom row as b1,…b7, then the diagram corresponds to the following set cover problem:

Cover {a1,…,a7, b1,…b7} with the following subsets:

S1 = {a1,…,a7}

S2 = {b1,…,b7}

S3 = {a1,b1}

S4 = {a2,a3,b2,b3}

S5 = {a4,a5,a6,a7,b4,b5,b6,b7}

(a) Recall the following notation: “Let nt be the number of elements still not covered after t iterations of the greedy algorithm (so n0 = n).” Trace the greedy algorithm by listing the successive values of nt and the subset Si chosen at each step. How many subsets does the greedy algorithm choose?

(b) What is the optimal set cover for this problem instance?

(c) Extend the above diagram to one with 24-1 = 15 dots along the top row and 15 dots along the bottom row. Using a similar pattern of subsets, we can have |S1| = |S2| = 15; S3,..,S5 as before; and a new subset S6 added on the far right with |S6| = |16|. The greedy algorithm yields a rather high (that is, poor) approximation factor on this instance. How large is the set cover that is selected by the greedy algorithm, and how large is the optimal set cover?

(d) Now extend this pattern to one with 2n-1 dots along the top row and 2n-1 dots along the bottom row. How large is the set cover that is selected by the greedy algorithm? How large is the optimal set cover?