Data Structure and Algorithm II

Homework #4

Due: 1pm, Monday, December 12, 2011

=== Homework submission instructions ===

- Submit the answers for writing problems (including your programming report) through the CEIBA system (electronic copy) or to the TA in R432 (hard copy). Please write down your name and school ID in the header of your documents. You also need to submit your programming assignment (problem 1) to the Judgegirl System(`http://katrina.csie.ntu.edu.tw/judgegirl/`).

- Each student may only choose to submit the homework in one way; either all as hard copies or all through CEIBA except the programming assignment. If you submit your homework partially in one way and partially in the other way, you might only get the score of the part submitted as hard copies or the part submitted through CEIBA (the part that the TA chooses).

- If you choose to submit the answers of the writing problems through CEIBA, please combine the answers of all writing problems into only one file in the doc/docx or pdf format, with the file name in the format of "hw4_[student ID].{pdf,docx,doc}" (e.g. "hw4_b99902010.pdf"); otherwise, you might only get the score of one of the files (the one that the TA chooses).

- For each problem, please list your references (they can be the names of the classmates you discussed the problem with, the URL of the information you found on the Internet, or the names of the books you read). The TA can deduct up to 100% of the score assigned to the problems where you don't list your references.

Problem 1. (30%) Gene Regulatory Network (GRN) is of significant interest in bioinformatics since it describes cellular systems. A typical gene regulatory network can be described as follows.The vertices of the network represent genes, and the edges between the vertices represent gene-gene interaction through which the products of one gene affect those of the other.

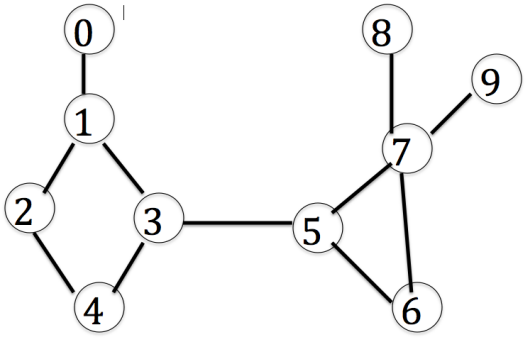


Figure 1: Gene Regulatory Network (GRN)

One important application of GRN is to find critical genes, which are articulation points in the network. After finding the critical genes, we can develop medicine to regulate the genes and cure diseases such as cancer and HIV.

In this problem, we ask you to find all critical genes for a given GRN. Figure 1 is an example of GRN.

Input:

The first line contains an integer n ($1 \leq n \leq 1000$), which gives the total number of the genes in the network. In each of the following lines, there are two numbers i_A and i_B representing the gene-gene interaction of gene A with index i_A and gene B with index i_B. (As an example, the test case corresponds to Figure 1) For each gene-gene interaction, both indexes of the genes are between 0 to 999.

Output:

For each Gene Regulatory Network, print the line "Gene i_A is a critical gene.", where i_A is the index of the critical gene. And sort the gene according to their indices non-decreasingly. If there is no critical gene in this network, print the line "No critical genes are found."

Sample Input 1:

```
10
0 1
1 2
1 3
```

```
4 2
4 3
3 5
5 6
5 7
7 6
7 8
7 9
```

Sample Output 1:

```
Gene 1 is a critical gene.
Gene 3 is a critical gene.
Gene 5 is a critical gene.
Gene 7 is a critical gene.
```

Sample Input 2:

```
6
0 1
0 2
3 4
3 5
```

Sample Output 2:

```
Gene 0 is a critical gene.
Gene 3 is a critical gene.
```

Please write a program to solve this problem. Please also submit a report in which you give a clear description of your algorithm. (20 points for 10 test cases and 10 points for the report.)

Sol: To be announced later.

Problem 2. (10%) Prove that Prim's algorithm finds a minimum cost spanning tree for every undirected connected graph. (Hint: You can try to prove this by replacing some edges in a minimum spanning tree with edges which belong to the output of Prim's algorithm.)

Sol:

Let $T^* = (V, E^*)$ be a minimum cost spanning tree and $T = (V, E)$ be the output of Prim's algorithm. If $T = T^*$, then T is a minimum spanning tree. Otherwise, let e' be the first edge added during the construction of T that is not in T^*. Assume that e' is the edge that connects V_1 and V_2, where V_1 is the set of vertices which connected by the edges added before e, and $V_2 = V - V_1$. There exists an edge $e \in E^*$ connecting V_1 and V_2 that is not in E. Let $E' = E^* - \{e\} + \{e'\}$ and $T' = (V, E')$. It is clear that T' is connected and since $|E'| = |E^*|$, T' is a tree. Since $w(e') \leq w(e)$, $w(E') = w(E^*)$ and thus T' is also a minimum spanning tree. Repeating the above procedure until $E' = E$, we can conclude from $T = T'$ that T is also a minimum cost spanning tree.

Problem 3. (10%) Solve the following problems:

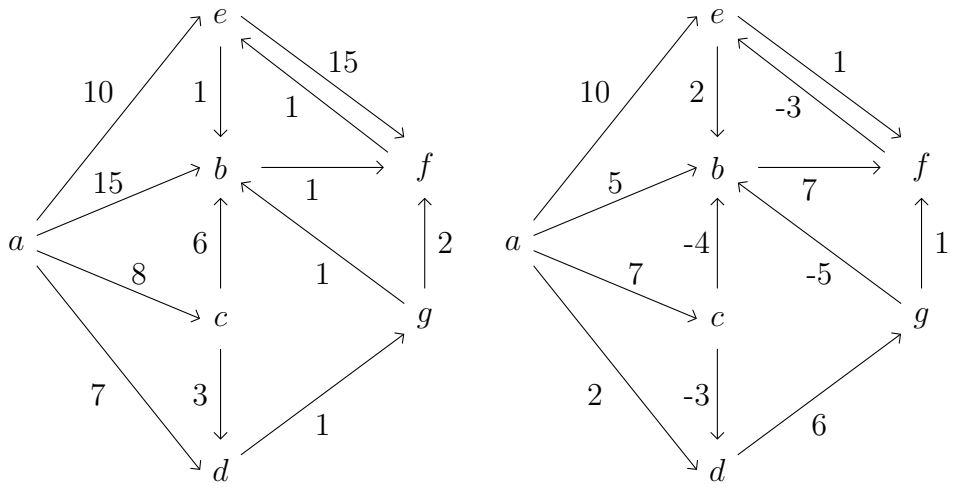
1. (5%) Consider digraph G_1 shown in Figure 2(a). Use the Dijkstra algorithm described in section 24.3 in the textbook to determine the shortest path from vertex a to all other vertices in G_1. Please fill in the table in Figure 3(a) in order to explain your answer in a step-by-step manner.

2. (5%) Consider digraph G_2 shown in Figure 2(b). Use the Bellman-Ford algorithm described in section 24.1 in the textbook to determine the shortest path from vertex a to all other vertices in G_2. Please fill in the table in Figure 3(b) in order to explain your answer in a step-by-step manner.

Sol:

1. See Figure 4(a) for the resulting table. Figure 4(b) shows the resulting shortest path tree returned by Dijkstra's algorithm.

2. See Figure 5 for the resulting table. After $k = 6$ runs, the algorithm then check whether there exists a negative weight cycle. Since there exists a (u, v) such that



(a) Directed graph G_1 (b) Directed graph G_2

Figure 2: Graphs

$v.d > u.d + w(u, v)$, as when $(u, v) = (e, f)$, there is no solution for the shortest path problem. Therefore, BELLMAN-FORD(G,w,s) will return FALSE.

Problem 4. (10%) Read Section 21.1, 21.2 and 21.3 on p.561-572 of the textbook, then solve the following problems:

1. (5%) 21.2-2 on p.567
2. (5%) 21.3-1 on p.572

Sol:

1. See Figure 6.
2. See Figure 7.

Problem 5. The outcome of the software company game will contribute to 40% of your homework 4 score.

Sol: Skipped.

Iteration	Vertex selected	Distance						
		a	b	c	d	e	f	g
Initial	- - - -	0	15	8	7	10	∞	∞
1								
2								
⋮								

(a) Table for the Dijkstra algorithm

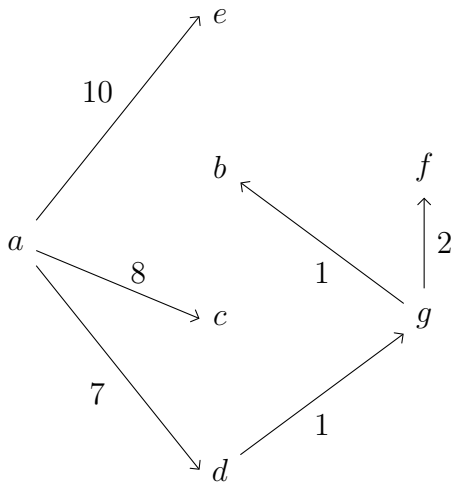
	Distance						
k	a	b	c	d	e	f	g
1	0	5	7	2	10	∞	∞
2	0						
3	0						
4	0						
5	0						
6	0						

(b) Table for the Bellman-Ford algorithm

Figure 3: Tables

Iteration	Vertex selected	Distance						
		a	b	c	d	e	f	g
Initial	- - - -	0	15	8	7	10	∞	∞
1	d	0	15	8	7	10	∞	8
2	c	0	14	8	7	10	∞	8
3	g	0	9	8	7	10	10	8
4	b	0	9	8	7	10	10	8
5	e	0	9	8	7	10	10	8
6	f	0	9	8	7	10	10	8

(a) Table for the Dijkstra algorithm



(b) A shortest path tree of G_1

Figure 4: Solution to the Dijkstra's Algorithm in Problem 3

	Distance						
k	a	b	c	d	e	f	g
1	0	5	7	2	10	∞	∞
2	0	3	7	2	10	11	8
3	0	3	7	2	8	9	8
4	0	3	7	2	6	9	8
5	0	3	7	2	6	7	8
6	0	3	7	2	4	7	8

Figure 5: Solution to the Bellman-Ford Algorithm in Problem 3

for $i = 1$ **to** 16
    MAKE-SET(x_i)

for $i = 1$ **to** 15 **by** 2
    UNION(x_i,x_{i+1})

$x_1 \leftarrow x_2 \quad x_3 \leftarrow x_4 \quad x_5 \leftarrow x_6 \quad x_7 \leftarrow x_8 \quad x_9 \leftarrow x_{10} \quad x_{11} \leftarrow x_{12} \quad x_{13} \leftarrow x_{14} \quad x_{15} \leftarrow x_{16}$

for $i = 1$ **to** 13 **by** 4
    UNION(x_i,x_{i+2})

$x_1 \leftarrow x_2 \leftarrow x_3 \leftarrow x_4 \quad x_5 \leftarrow x_6 \leftarrow x_7 \leftarrow x_8 \quad x_9 \leftarrow x_{10} \leftarrow x_{11} \leftarrow x_{12} \quad x_{13} \leftarrow x_{14} \leftarrow x_{15} \leftarrow x_{16}$

UNION(x_1,x_5)
UNION(x_{11},x_{13})

$x_1 \leftarrow x_2 \leftarrow x_3 \leftarrow x_4 \leftarrow x_5 \leftarrow x_6 \leftarrow x_7 \leftarrow x_8 \quad x_9 \leftarrow x_{10} \leftarrow x_{11} \leftarrow x_{12} \leftarrow x_{13} \leftarrow x_{14} \leftarrow x_{15} \leftarrow x_{16}$

UNION(x_1,x_{10})

$x_1 \leftarrow x_2 \leftarrow x_3 \leftarrow x_4 \leftarrow x_5 \leftarrow x_6 \leftarrow x_7 \leftarrow x_8 \leftarrow x_9 \leftarrow x_{10} \leftarrow x_{11} \leftarrow x_{12} \leftarrow x_{13} \leftarrow x_{14} \leftarrow x_{15} \leftarrow x_{16}$

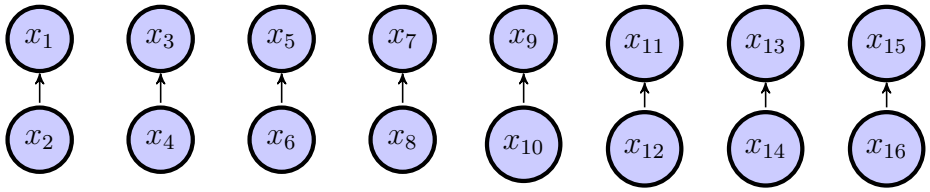
FIND-SET(x_2): **return** x_1

FIND-SET(x_9): **return** x_1

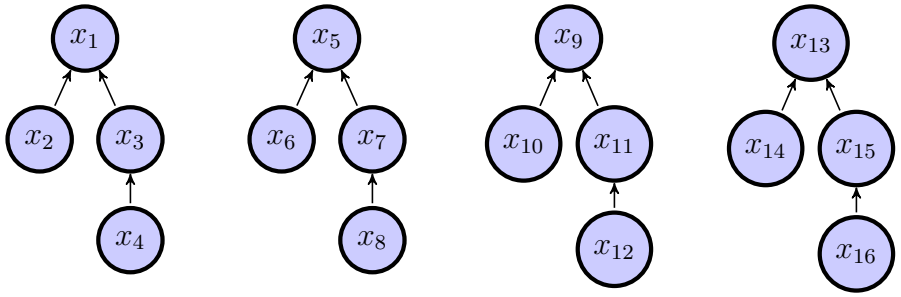
Figure 6: The data structure for Problem 21.2-2

for $i = 1$ **to** 16
    MAKE-SET(x_i)

for $i = 1$ **to** 15 **by** 2
    UNION(x_i,x_{i+1})



for $i = 1$ **to** 13 **by** 4
    UNION(x_i,x_{i+2})



UNION(x_1,x_5)
UNION(x_{11},x_{13})

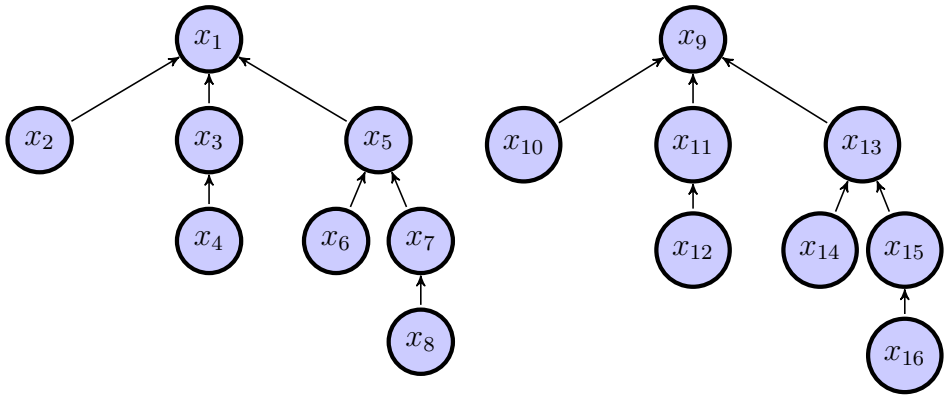
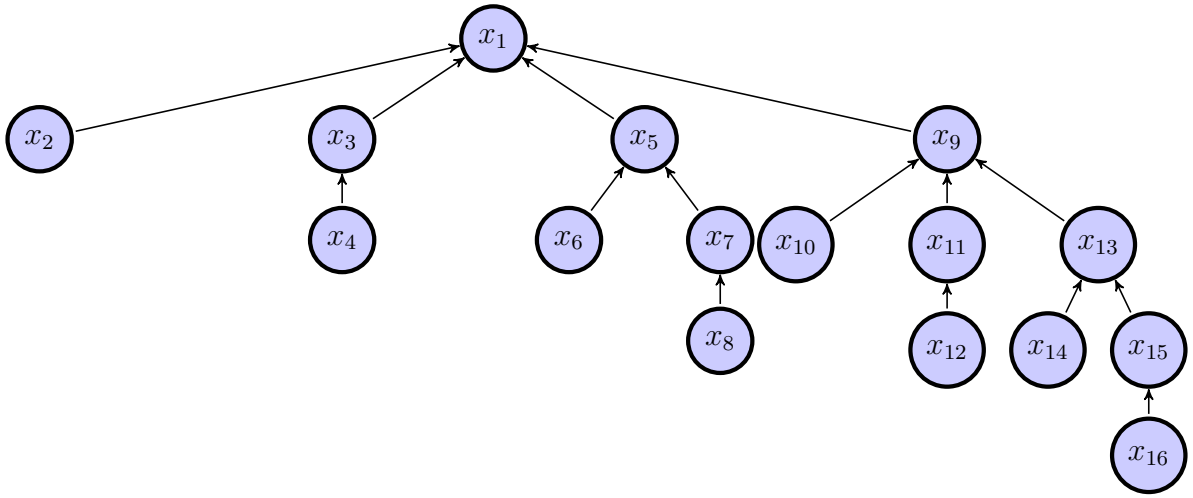


Figure 7: Solution to Problem 21.3-1

UNION(x_1,x_{10})



FIND-SET(x_2): **return** x_1

FIND-SET(x_9): **return** x_1

Figure 7: Solution to Problem 21.3-1 (cont)