CS 206 Data Structures Spring 2015

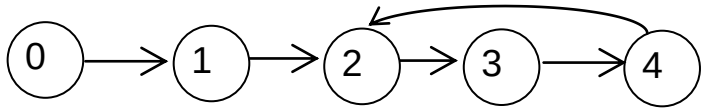
**Homework 6**

Sungwon Kang Due June 4

1. (a) Write a program that finds a depth-first spanning tree of a connected graph.
   (b) Write a program that finds a breadth-first spanning tree of a connected graph.

2. Write a program that finds all the connected components of a graph.

3. The 3 leftmost columns of the table below is a adjacency matrix representation of the following graph G.

G: 


Fill out the blank columns with T or F as the Warshall's algorithm would do. Circle T's that is changed from F to make it clearly visible.

| i | j | A[i,j] | i | k | j | A[i,j] | i | k | j | A[i,j] | i | k | j | A[i,j] | i | k | j | A[i,j] | i | k | j | A[i,j] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | **F** | 0 | 0 | 0 | | 0 | 1 | 0 | | 0 | 2 | 0 | | 0 | 3 | 0 | | 0 | 4 | 0 | |
| 0 | 1 | **T** | 0 | 0 | 1 | | 0 | 1 | 1 | | 0 | 2 | 1 | | 0 | 3 | 1 | | 0 | 4 | 1 | |
| 0 | 2 | **F** | 0 | 0 | 2 | | 0 | 1 | 2 | | 0 | 2 | 2 | | 0 | 3 | 2 | | 0 | 4 | 2 | |
| 0 | 3 | **F** | 0 | 0 | 3 | | 0 | 1 | 3 | | 0 | 2 | 3 | | 0 | 3 | 3 | | 0 | 4 | 3 | |
| 0 | 4 | **F** | 0 | 0 | 4 | | 0 | 1 | 4 | | 0 | 2 | 4 | | 0 | 3 | 4 | | 0 | 4 | 4 | |
| 1 | 0 | **F** | 1 | 0 | 0 | | 1 | 1 | 0 | | 1 | 2 | 0 | | 1 | 3 | 0 | | 1 | 4 | 0 | |
| 1 | 1 | **F** | 1 | 0 | 1 | | 1 | 1 | 1 | | 1 | 2 | 1 | | 1 | 3 | 1 | | 1 | 4 | 1 | |
| 1 | 2 | **T** | 1 | 0 | 2 | | 1 | 1 | 2 | | 1 | 2 | 2 | | 1 | 3 | 2 | | 1 | 4 | 2 | |
| 1 | 3 | **F** | 1 | 0 | 3 | | 1 | 1 | 3 | | 1 | 2 | 3 | | 1 | 3 | 3 | | 1 | 4 | 3 | |
| 1 | 4 | **F** | 1 | 0 | 4 | | 1 | 1 | 4 | | 1 | 2 | 4 | | 1 | 3 | 4 | | 1 | 4 | 4 | |
| 2 | 0 | **F** | 2 | 0 | 0 | | 2 | 1 | 0 | | 2 | 2 | 0 | | 2 | 3 | 0 | | 2 | 4 | 0 | |
| 2 | 1 | **F** | 2 | 0 | 1 | | 2 | 1 | 1 | | 2 | 2 | 1 | | 2 | 3 | 1 | | 2 | 4 | 1 | |
| 2 | 2 | **F** | 2 | 0 | 2 | | 2 | 1 | 2 | | 2 | 2 | 2 | | 2 | 3 | 2 | | 2 | 4 | 2 | |
| 2 | 3 | **T** | 2 | 0 | 3 | | 2 | 1 | 3 | | 2 | 2 | 3 | | 2 | 3 | 3 | | 2 | 4 | 3 | |
| 2 | 4 | **F** | 2 | 0 | 4 | | 2 | 1 | 4 | | 2 | 2 | 4 | | 2 | 3 | 4 | | 2 | 4 | 4 | |
| 3 | 0 | **F** | 3 | 0 | 0 | | 3 | 1 | 0 | | 3 | 2 | 0 | | 3 | 3 | 0 | | 3 | 4 | 0 | |
| 3 | 1 | **F** | 3 | 0 | 1 | | 3 | 1 | 1 | | 3 | 2 | 1 | | 3 | 3 | 1 | | 3 | 4 | 1 | |
| 3 | 2 | **F** | 3 | 0 | 2 | | 3 | 1 | 2 | | 3 | 2 | 2 | | 3 | 3 | 2 | | 3 | 4 | 2 | |
| 3 | 3 | **F** | 3 | 0 | 3 | | 3 | 1 | 3 | | 3 | 2 | 3 | | 3 | 3 | 3 | | 3 | 4 | 3 | |
| 3 | 4 | **T** | 3 | 0 | 4 | | 3 | 1 | 4 | | 3 | 2 | 4 | | 3 | 3 | 4 | | 3 | 4 | 4 | |
| 4 | 0 | **F** | 4 | 0 | 0 | | 4 | 1 | 0 | | 4 | 2 | 0 | | 4 | 3 | 0 | | 4 | 4 | 0 | |
| 4 | 1 | **F** | 4 | 0 | 1 | | 4 | 1 | 1 | | 4 | 2 | 1 | | 4 | 3 | 1 | | 4 | 4 | 1 | |
| 4 | 2 | **T** | 4 | 0 | 2 | | 4 | 1 | 2 | | 4 | 2 | 2 | | 4 | 3 | 2 | | 4 | 4 | 2 | |
| 4 | 3 | **F** | 4 | 0 | 3 | | 4 | 1 | 3 | | 4 | 2 | 3 | | 4 | 3 | 3 | | 4 | 4 | 3 | |
| 4 | 4 | **F** | 4 | 0 | 4 | | 4 | 1 | 4 | | 4 | 2 | 4 | | 4 | 3 | 4 | | 4 | 4 | 4 | |

4. What does the following list look like after the first iteration of radix sort's outer loop?

**class, leaks, every, other, refer, embed, array**

5. If we are using Mergesort, what will the following array look like right before the *last* merge

**35 57 53 26 50 15 22 21 25 14 11 2**

6. If we are using Quicksort, what will the result be if we pivot on 35?
(Show the progress of sorting by presenting the result of each level of recursion.)

**35 57 53 26 50 15 22 21 25 14 11 2**

7. Heapify the following list, placing the maximum on top.

**35 57 53 26 50 15 22 21 25 14 11 2**

8. Beginning with the following array, what is the result of running the heapsort procedure (take max put it on the end of the heap, re-heapify) after four iterations?

**57 50 53 26 35 15 22 21 25 14 11 2**