NATIONAL TAIWAN NORMAL UNIVERSITY
Department of Computer Science and Information Engineering

Introduction to Algorithms

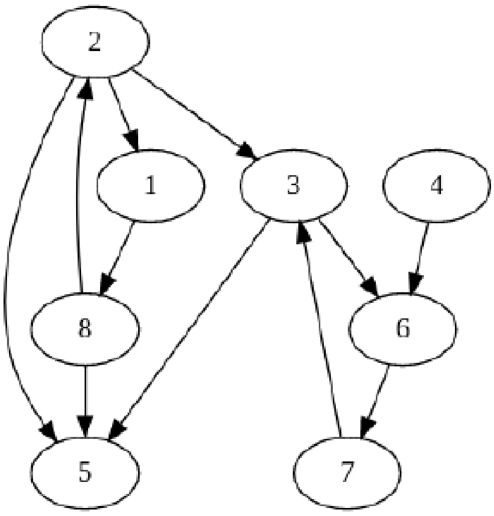
Final Examination
Tuesday 05/31/2022

Instructions:

- Examination time: 10:20-12:15 (100 minutes + 15 minutes for manipulating answer sheets).
- This exam contains 6 problems, some with multiple parts.
- This exam is closed book. **No tolerance for cheating.**
- **Show your work, as partial credit will be given.** You will be graded not only on the correctness of your answer, but also on the clarity with which you express it.
- You may find the total score exceeds 100. Those extra points are bonus! Consider the time you need to solve for each problem, and make a good decision that will give you maximal points. The maximal score you can get is 100.
- **Good luck!**

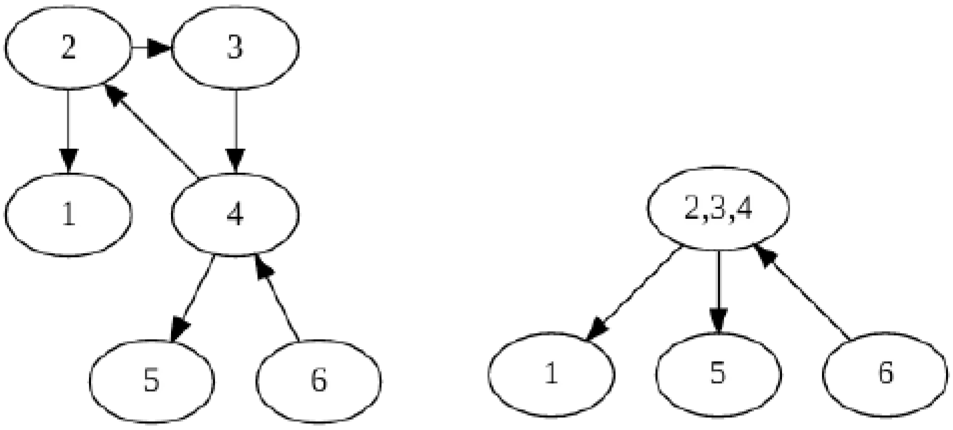
Problem #1 (25). A directed graph is strongly connected if there is a path between all pairs of vertices. The **strongly connected components (SSCs)** of a directed graph form a partition into subgraphs that are themselves strongly connected. Please answer the questions regarding SSCs.

(a) Please traverse the graph below by using depth-first search (DFS). Write down the time stamps of discovery and finishing time. Start from vertex 1. Resolve ties by ascending order. **(5)**



(b) Please find the SSCs of the graph above. **(5)**

(c) Given a directed graph G, we can shrink each SSC down to a single vertex. Here is an example.

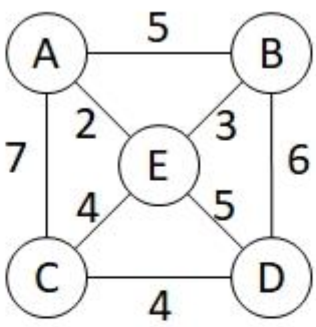


Since the vertices 2, 3, 4 are strongly connected, they can be shrunk into a vertex. Denote the graph whose SCCs are all shrunk as G^{SCC}, and G^T is the transpose of G. G^T is obtained by reversing the directions of all edges in graph G. Please prove that $(G^{SCC})^T = (G^T)^{SCC}$. **(10)**

(d) How many strongly connected components does a dag have? **(5)**

Problem #2 (10). Please answer the questions regarding Prim's algorithm.

(a) Please apply Prim's algorithm to the following graph. **(5)**



(b) What happens if the input graph to Prim's algorithm has negative weights? **(5)**

Problem #3 (25). The Floyd-Warshall algorithm solves the all-pairs shortest paths problem with time complexity $\Theta(n^3)$. The pseudo code is provided as follows:

Floyd-Warshall(W)
$n \leftarrow \text{rows}[W]$
$D^{(0)} \leftarrow W$
for $k \leftarrow 1$ to n
do for $i \leftarrow 1$ to n
do for $j \leftarrow 1$ to n
$d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $D^{(n)}$

(a) Explain the meaning of $d_{ij}^{(k)}$. **(**Your answer should include the meaning of i, j and k). **(5)**

(b) Given a graph with the weight matrix W:

$$\begin{bmatrix} 0 & 3 & \infty & 2 & 6 \\ 5 & 0 & 4 & 2 & \infty \\ \infty & \infty & 0 & 5 & \infty \\ \infty & \infty & 1 & 0 & 4 \\ 5 & \infty & \infty & \infty & 0 \end{bmatrix}$$

please write down $D^{(1)}$. **(5)**

(c) As it appears above, the Floyd-Warshall algorithm requires a 3D array, since we compute $d_{ij}^{(k)}$ for $i, j, k = 1, 2, ..., n$. Now, we use a 2D array instead, and modify the algorithm as follows:

```
Floyd-Warshall-Modified(W)
n ← rows[W]
D ← W
for k ← 1 to n
    do for i ← 1 to n
        do for j ← 1 to n
            d_ij ← min(d_ij, d_ik + d_kj)
return D
```

Does the modified algorithm return correct results? Briefly explain your answer. **(5)**

(a) Does the Floyd-Warshall algorithm give the correct result if the input graph has negative weights (but no negative cycle)? Briefly explain your answer. **(5)**

(b) Please describe how do we use the Floyd-Warshall algorithm to check whether the input graph has a negative-weight cycle? **(5)**

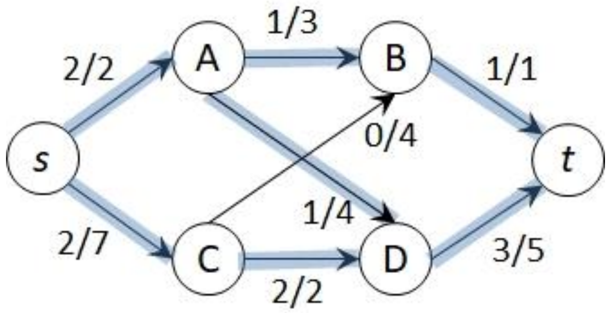
Problem #4 (15). To use the reweighting technique for solving the all-pairs shortest paths problem, professor Yeh says that there is a simpler way to reweight edges than the method used in Johnson's algorithm. Let $c = \min\{w(u, v)\}$, and define $\hat{w}(u, v) = w(u, v) - c$ for all edges (u, v) of G.

(a) What is wrong with the professor's method for reweighting? **(5)**

(b) In the class we introduced Johnson's method that finds all-pairs shortest paths via $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$. Now suppose that $w(u, v) \geq 0$ for all edges $(u, v) \in E$, what is the relationship between the original and the updated weights (w and $\hat{w}$)? **(5)**

(c) Continuing (b), $h(u)$ is the minimum weight of a path from q to u, $u \in V$, where q is a new node added to the input graph. We use the Bellman Ford algorithm to compute $h(u)$. Please briefly explain the reason why do we use the Bellman Ford algorithm, rather than Dijkstra's algorithm? **(5)**

Problem #5 (20). Given the input network *G* and a flow *f* below, please answer the following questions. The numbers (*i* / *j*) on each edge denote the flow value (*i*) and the edge capacity (*j*).



(a) Please write down the residual network of *G* induced by *f*. **(5)**

(b) Does the flow *f* have maximal value? If your answer is "yes", prove it by marking a cut with a capacity equal to the flow value. If your answer is "no", show how to get a flow with a higher value. **(5)**

(c) In Ford-Fulkerson's algorithm, two types of edges are created in computing the residual network, given a flow and a network. Describe the reasons why "backward edges" are required? **(5)**

(d) Describe how the Ford-Fulkerson method can be used to solve the maximum bipartite matching problem. **(5)**

Problem #6 (10). Please answer the questions regarding NP completeness.

(a) Describe the difference between NP-complete and NP-hard. **(5)**

(b) Suppose P = NP, please draw the Venn diagram of the complexity classes P, NP, NP-complete and NP-hard. **(5)**