NATIONAL TAIWAN NORMAL UNIVERSITY
Department of Computer Science and Information Engineering

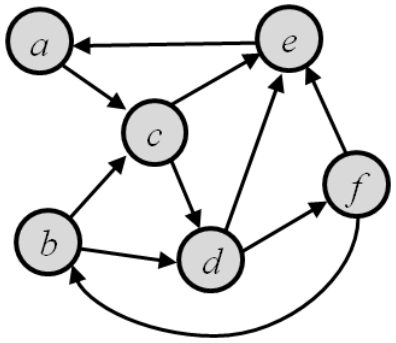
# Introduction to Algorithms

## Final Examination
## Tuesday 06/08/2021

**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.
- **Good luck!**

**Problem #1 (15).** Given a directed graph *G*:




(a) Please perform depth-first-search (DFS) on *G*. Start from *a*. Please write down the time stamps of discovery and finishing time. **(5)**

(b) Please show all strongly connected components of *G*. **(5)**

(c) We can use two passes of DFS to find the strongly connected components in a directed graph. The first pass of DFS, in the original graph, is used to find the finishing time of each vertex. The second pass of DFS is performed on the transpose of the original graph, considering the vertices in order of **decreasing finishing time**. Why in the second pass should we use the order of decreasing finishing time? Please briefly describe the reasons. **(5)**

**Problem #2 (10).** The Dijkstra's algorithm can be used to find the shortest path from a single source vertex to all other vertices in a graph, with the pseudo code given as follows.

```
DIJKSTRA(G, w, s)
// G: input graph, w: edge weights, s: source vertex
1    INITIALLIZE(G, s)
2    S = {}
3    Q = G.V
4    while Q ≠ {}
5        u = EXTRACT-MIN(Q)
6        S = S ∪ {u}
7        for each vertex v ∈ G.Adj[u]
8            if v.d > u.d + w(u, v)
9                v.d = u.d + w(u, v)
```

(a) In line 5, this algorithm finds a vertex with a minimal value (*u.d*). What does the value mean? **(5)**

(b) Please give an example showing that the Dijkstra's algorithm does not work. **(5)**

**Problem #3 (20).** 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)
1   n ← rows[W]
2   D^(0) ← W
3   for k ← 1 to n
4       do for i ← 1 to n
5           do for j ← 1 to n
6               d_ij^(k) ← min(d_ij^(k−1), d_ik^(k−1) + d_kj^(k−1))
7   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 edge weight:

$$W = D^{(0)} = \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix},$$

and the intermediate result $D^{(3)}$:

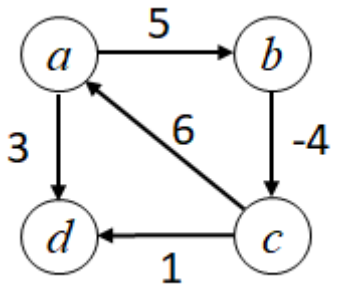
$$D^{(3)} = \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 9 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix}$$

what are the values of "$?_{\mathbf{I}}$" and "$?_{\mathbf{II}}$" in $D^{(4)}$? **(10)**

$$D^{(4)} = \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ ?_{\mathbf{I}} & 7 & 0 & ?_{\mathbf{II}} \\ 6 & 16 & 9 & 0 \end{bmatrix}$$

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

**Problem #4 (25).** Given the graph below, we apply Johnson's algorithm (a re-weighting method) to find the shortest paths between all pairs of vertices. Please answer the following questions regarding each step of Johnson's algorithm.




(a) We first add a new node $q$ to the input graph, and connect it with zero-weight edges to each of the other nodes. Next, we run the Bellman-Ford algorithm to find for each vertex $v \in \{a, b, c, d\}$ the minimum weight $h(v)$ of a path from $q$ to $v$. The Bellman-Ford algorithm makes several passes over all edges of the graph. Please explain:

i. Why the Bellman-Ford algorithm is used, rather than using Dijkstra's algorithm? **(3)**

ii. How many passes do we need for processing the graph above (including the augmented vertex $q$)? **(2)**

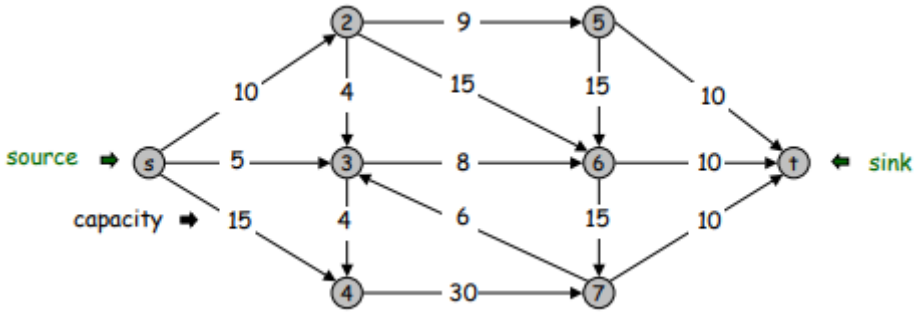
(b) The previous step returns the results: $h(a) = 0$, $h(b) = 0$, $h(c) = -4$, and $h(d) = -3$. An edge from $u$ to $v$, having a length $w(u, v)$, is now given the new length $w(u, v) + h(u) - h(v)$. This step clearly reweights the edges of the original graph. Please draw the reweighted graph. **(5)**

(c) Finally, we apply Dijkstra's algorithm to find the shortest paths from each node to every other vertex in the reweighted graph. Please write down the shortest path lengths using vertex $a$ as the source. In other words, please compute the shortest path length $\delta(a, b)$ between $a$ and $b$, and similarly, $\delta(a, c)$ and $\delta(a, d)$. **(5)**

(d) Professor Yeh says that there is a simpler way than Johnson's method to reweight edges. Let $c = \min\{w(u, v)\}$, and define $\hat{w}(u, v) = w(u, v) - c$ for all edges $(u, v)$ of the input graph. What is wrong with Yeh's reweighting method? **(5)**

(e) Professor Yeh says again that there is no need to create a new source vertex *q* in the first step of Johnson's algorithm. Instead, she says that we can just use *G*' = *G* and let the source be *any* vertex. Is she right? If the answer is no, please give an example for which using Yeh's idea into Johnson's algorithm causes incorrect answers. **(5)**

**Problem #5 (20).** Given a network with a source *s* and a sink *t*, the Ford-Fulkerson algorithm for solving the maximum-flow problem iteratively finds an augmenting path in a residual network. Now we have an input network shown below.




(a) What is the maximum flow of the network? **(5)**

(b) What is the minimum cut of the network? Please show the partition of vertices of the cut. **(10)**

(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)**

**Problem #6 (10).** Professor Yeh discovers a new problem X, which can be verified in polynomial time. She finds two other problems: Y and Z, where Y is a P problem and Z is a NP-complete problem. Given four possible actions described below, please answer the following two questions.

i. Show X is polynomial-time reducible to Y. ($X \leq_P Y$)

ii. Show Y is polynomial-time reducible to X. ($Y \leq_P X$)

iii. Show X is polynomial-time reducible to Z. ($X \leq_P Z$)

iv. Show Z is polynomial-time reducible to X. ($Z \leq_P X$)

(a) If Prof. Yeh would like to show that problem X is a NP-complete problem, which action should she take (i, ii, iii, or iv)? Why? **(5)**

(b) If Prof. Yeh would like to show that problem X is computational tractable, which action should she take (i, ii, iii, or iv)? Why? **(5)**