Algorithm Final Exam (Spring 2023)

Total：125 %

Name: _____

Student ID #: _____

Note: Short answers or answers with no procedure to demonstrate how you obtain your answers is not recommended except for the first question (the first 40%).

Question	Score
1 (40%)	
2 (5%)	
3 (10%)	
4 (10%)	
5 (25%)	
6 (15%)	
7 (20%)	
Total	

--- Introduction & Simple Questions ---

1. **[Simple questions: 40%]** Only simple answers (such as one to three sentences) are necessary for the following questions.
   (a) (5%) Name one sorting method that has its time complexity insensitive to the inputs.

   (b) (5%) How different between the problem of using the minimum number of coins to pay an amount and the *0-1 knapsack* problem? It means the answer for one cannot be converted to the answer for the other.

   (c) (5%) How different between the problem of using the minimum number of coins to pay an amount and the *fractional knapsack* problem? It means the answer for one cannot be converted to the answer for the other.

(d) (5%) When performing UNION operation in disjoint set data structure, if we obtain the same tree height before and after the UNION operation, what you can conclude from here? We assume the tree height is chosen as the tree rank.

(e) (5%) If we reverse the direction for one of the edges in a directed graph without cycles, is it possible to obtain the same topological sort result?

(f) (5%) If we reverse all the directed edges in a directed graph to the opposite directions, can we obtain the same strongly connected components?

(g) (4%) Explain the meaning of the equality: $O(n) + \Theta(n^2) = \Theta(n^2)$. Hint: you should remember that all the notations over here imply a set of functions.

(h) (4%) Explain the meaning of having the time complexity $\Theta(V + E)$ for a graph with V vertices and E edges.

(i) (2%) How do you think of the lectures of this semester? Any constructive comments are welcome.

--- Before the Midterm ---

2. **[Solve Recurrence Equation: 5%]** Find asymptotic behavior for the recurrence equation given by:

$$T(n) = 2T\left(\frac{n}{5}\right) + n^2 \lg n$$

3. **[Hashing 10%]** Explain by examples about why quadratic probing could be better than linear probing and double hashing could be better than quadratic probing. You may assume a primary cluster starting from the positions no. 10 to no. 15 from linear probing and a secondary cluster starting from the position 10. No need to give all step-by-step results but the details enough to explain your answer.

--- Dynamic Programming & Greedy Algorithms ---

4. **[Greedy Algorithms 10%]** In the activity selection problem, if someone wrongly sorting the activity starting time instead of the finishing time, can you suggest a strategy to help for this situation and you can still obtain the correct result in the end?

5. **[Dynamic Programming 25%]**

(a) (10%) Given six matrices $A_1, A_2, \ldots, A_6$ with their dimensions equal to 1×20, 20×3, 3×10, 10×2, 2×50, 50×4, find the slowest way to compute the product of $A_1A_2A_3A_4A_5A_6$. You should write down your answer by putting full parentheses to the product.

(b) (10%) When finding LCS for two sequences, we may focus on the recurrent equation given by:

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \max\big(c[i-1,j-1] + (x_i == y_j), c[i-1,j], c[i,j-1]\big) & \text{if } i,j > 0 \end{cases}$$

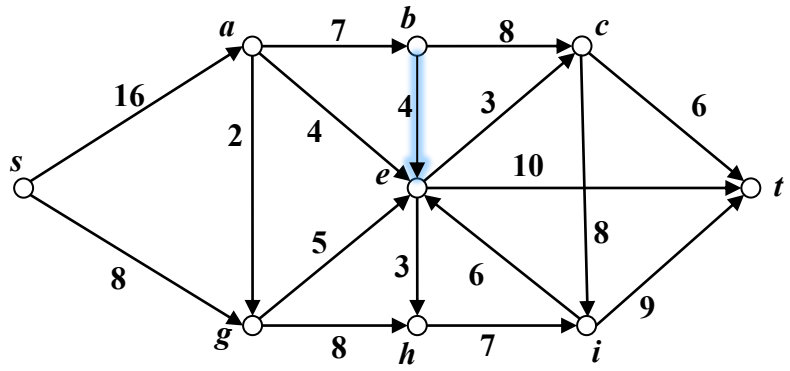
to record the length of LCS for two partial sequences $X_i = (x_1, x_2, \ldots x_i)$ and $Y_j = (y_1, y_2, \ldots y_j)$. Note that $(x_i == y_j)$ means the identity check. Can you follow the similar format to write down the recurrent equation when we want to find the LCS for three sequences X_i, Y_j, and Z_k.

(c) (5%) Following part (b), can you describe the time complexity used in (b), for three given sequences of length n?

--- Graph Algorithms ---

6. **[DFS & BFS 15%]** Answer two questions regarding DFS and BFS
   (a) (8%) If someone was not satisfied with a Topological Sorting (TS) result, can you modify the algorithm to output another alternative TS result if exists?
   (b) (7%) Instead of finding all Strongly Connected Components given a directed graph, can you modify the SCC algorithm to output the number of SCCs?

7. **[Maximum Flow: 20%]** Answer three questions related to maximum flow.



(a) (10%) Find the maximum flow of the above network with step-by-step result.

(b) (5%) Can you modify the procedure if we claim that the capacity on edge be must be utilized as much as we can? It means if we can obtain the same amount of maximum flow, we should give higher priority to use the capacity on edge be if possible.

(c) (5%) What is the minimum cut in part (b)?

We list several codes for your reference.

MAKE-SET(x)

```
1  p[x] ← x
2  rank[x] ← 0
```

UNION(x, y)

```
1  LINK(FIND-SET(x), FIND-SET(y))
```

LINK(x, y)

```
1  if rank[x] > rank[y]
2      then p[y] ← x
3      else p[x] ← y
4           if rank[x] = rank[y]
5               then rank[y] ← rank[y] + 1
```

FIND-SET(x)

```
1  if x ≠ p[x]
2      then p[x] ← FIND-SET(p[x])
3  return p[x]
```

```
BFS(G, s)
1   for each vertex u ∈ V(G) − {s}
2         do color[u] ← WHITE
3            d[u] ← ∞
4            π[u] ← NIL
5   color[s] ← GRAY
6   d[s] ← 0
7   π[u] ← NIL
8   Q ← {s}
9   while Q ≠ ∅
10        do u ← DEQUEUE(Q)
11           for each v ← Adj[u]
12                do if color[v] = WHITE
13                     then color[v] ← GRAY
14                          d[v] ← d[u] + 1
15                          π[v] ← u
16                          ENQUEUE(Q, v)
17        color[u] ← BLACK
```

```
DFS(G)
1   for each vertex u∈V[G]
2         do color[u] ← WHITE
3            π[v] ← NIL
4   time ← 0
5   for each vertex u∈V[G]
6         do if color[u] = WHITE
7               then DFS-VISIT(u)

DFS-VISIT(u)
1   color[u] = GRAY
2   d[u] ← time ← time +1
3   for each v∈Adj[u]
4         do if color[v] = WHITE
5               then π[v] ← u
6                    DFS-VISIT(v)
7   color[u] = BLACK
8   f[u] ← time ← time + 1
```

TOPOLOGICAL-SORT(G)

1 Call DFS(G) to compute finishing time $f[v]$ for each vertex v.

2 As each vertex is finished, insert it onto the front of a link list.

3 **return** the link list of vertices.

STRONGLY-CONNECTED-COMPONENT(G)

1 call DFS(G) to compute finishing time $f[u]$ for each vertex u

2 compute G^T (transpose of G)

3 call DFS(G^T), but in the main loop of DFS, consider the vertices in the order of decreasing $f[u]$.

4 Output the vertices of each tree in the depth-first forest of step 3 as a separate SCCs.
