Algorithm Final Exam (Spring 2020)

總分：<u>120 %</u>

Name: ______________________

Student ID #: ________________

Question	Score
1 (25%)	
2 (5%)	
3 (10%)	
4 (10%)	
5 (10%)	
6 (20%)	
7 (20%)	
8 (20%)	
Total	

1. **[Simple questions: 25%]** Only simple answers are necessary for the following questions.
   (a) (5%) Order the functions based on their asymptotically behavior starting from the smallest one to the largest one: (i) $\alpha(n)$, (ii) $\lg n$, (iii) $\lg(n!)/\sqrt{n}$, (iv) $\lg\lg n$, (v) $2^{\lg n}$

   (b) (5%) Name a sorting method that is *sorted-in-place* and with the complexity not sensitive to its inputs. A few explanation is needed.

   (c) (5%) What is the length of the shortest path in the decision tree model for a quicksort? Why?

(e) (5%) Name a case or a situation when adjacency matrix is more appropriate to be used than adjacency list. Some explanation is necessary.

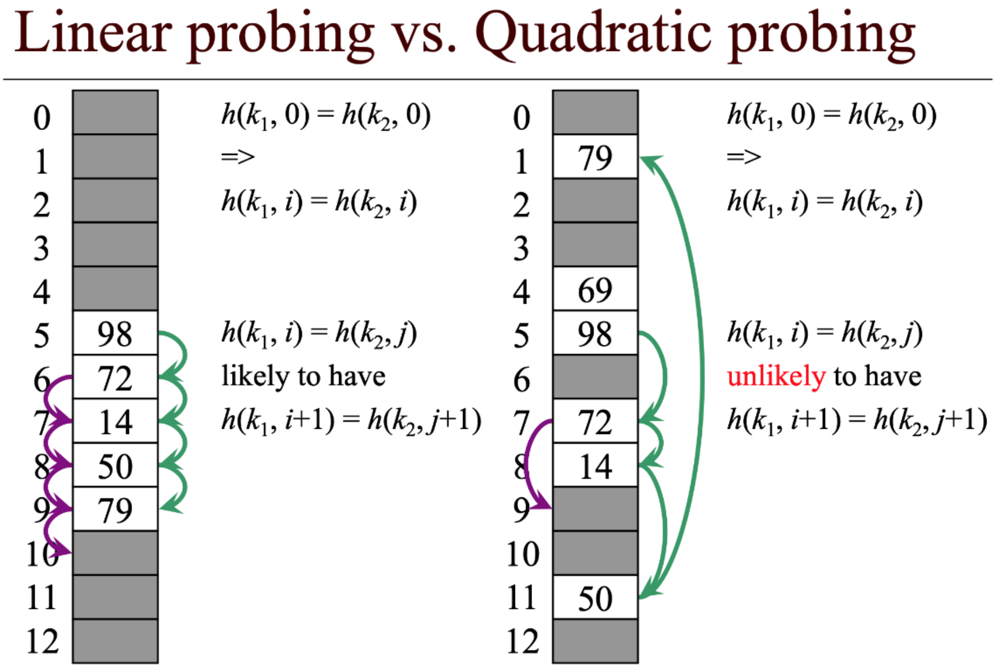
(f) (5%) How do you think of this course, too easy, too difficult, you learned anything interesting, comments?

2. **[Recurrence equation: 5%]** Solve the following recurrence equation.

$$T(n) = 9T\left(\frac{n}{3}\right) + n^3 \lg n$$

Your answer should be as tight as possible. You can use the master theorem when there is a need.

3. **[Hashing : 10%]** Explain why quadratic hashing is better than linear hashing given the following example.



Linear probing vs. Quadratic probing

0	
1	
2	
3	
4	
5	98
6	72
7	14
8	50
9	79
10	
11	
12	

$h(k_1, 0) = h(k_2, 0)$

=>

$h(k_1, i) = h(k_2, i)$

$h(k_1, i) = h(k_2, j)$

likely to have

$h(k_1, i{+}1) = h(k_2, j{+}1)$

0	
1	79
2	
3	
4	69
5	98
6	
7	72
8	14
9	
10	
11	50
12	

$h(k_1, 0) = h(k_2, 0)$

=>

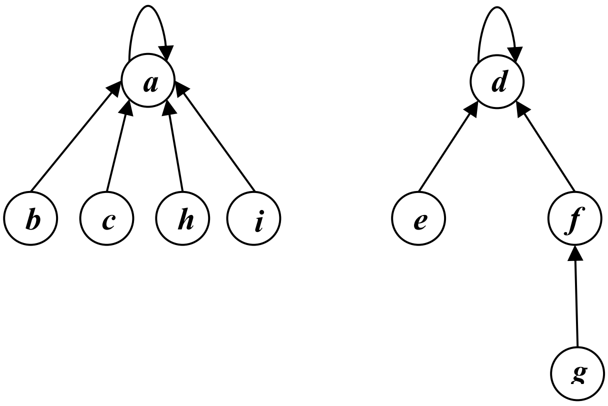
$h(k_1, i) = h(k_2, i)$

$h(k_1, i) = h(k_2, j)$

unlikely to have

$h(k_1, i{+}1) = h(k_2, j{+}1)$

4. **[Disjoint Set 10%]** Draw the result after performing operation *LINK*(*b*, *g*) given the following disjoint set. We assume *Union by Rank* and *Path Compression techniques.* Some details about how to obtain the result should be given.



5. **[Greedy Algorithms 10%]** For the activity-selection problem, if we prefer to choose the activities starting from the latter ones and going back to the earlier ones, can you suggest a greedy algorithm in this case? Some explanation about the correctness of the algorithm is preferred.

6. **[Dynamic Programming 20%]** Answer two questions related to Longest Common Subsequence (LCS).

(a) (10%) Find all the LCSs for two sequences 32457458 and 21436587 with the following recurrence:

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

Note that $(x_i \; == \; y_j)$ is 1 if $x_i = y_j$ or 0 if not.

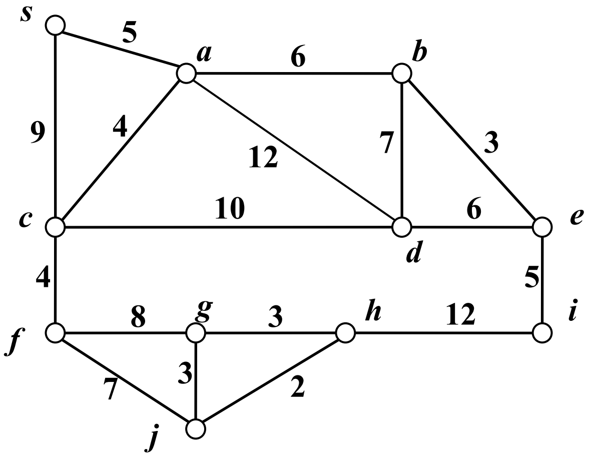
You may use the following table for your computation.

		2	1	4	3	6	5	8	7
3									
2									
4									
5									
7									
4									
5									
8									

(b) (10%) Can you suggest an algorithm to find the Longest Common Monotonically Increasing Subsequence? No need to show the result by sketching your approach is good enough.

7. **[DFS & BFS 20%]** Answer the questions related to DFS and BFS. You need to explain your answer for all the questions.
   (a) (10%) In the BFS and DFS pseudo codes, add a few lines to let the codes mark all the *tree* edges and *cross* edges.
   (b) (10%) In the algorithm to find the Strongly Connected Components, if someone forgot to reverse the edge directions and run the DFS twice with the original code (the second one choosing the vertex according to the finish time), what is the result in this case?

8. **[Minimum Spanning Tree: 20%]** Given the following graph, answer the questions related to minimum spanning tree.



(a) (5%) In the Prim's algorithm (starting from the vertex *s*), how many times Line 10 was executed when $v = d$.

(b) (5%) How to implement Line 11 of the Prim's algorithm. Your answer may not be long, but enough details should be provided.

(c) (10%) Can you modify the Kruskal algorithm and the Prim's algorithm so that we can find a minimum spanning tree with a given edge always included in the tree? You may use edge *cd* as an example to illustrate your idea.

We list several codes for your reference.

```
MAKE-SET($x$)
$p[x] \leftarrow x$
$rank[x] \leftarrow 0$
```

```
UNION($x$, $y$)
LINK(FIND-SET($x$), FIND-SET($y$))
```

```
LINK($x$, $y$)
if $rank[x] > rank[y]$
    then $p[y] \leftarrow x$
    else $p[x] \leftarrow y$
            if $rank[x] = rank[y]$
                then $rank[y] \leftarrow rank[y] + 1$
```

```
FIND-SET($x$)
if $x \neq p[x]$
    then $p[x] \leftarrow$ FIND-SET($p[x]$)
return $p[x]$
```

```
BFS($G$, $s$)
for each vertex $u \in V(G) - \{s\}$
        do $color[u] \leftarrow$ WHITE
            $d[u] \leftarrow \infty$
            $\pi[u] \leftarrow$ NIL
$color[s] \leftarrow$ GRAY
$d[s] \leftarrow 0$
$\pi[u] \leftarrow$ NIL
$Q \leftarrow \{s\}$
while $Q \neq \emptyset$
        do $u \leftarrow DEQUEUE(Q)$
            for each $v \leftarrow Adj[u]$
                do if $color[v]$ = WHITE
                    then $color[v] \leftarrow$ GRAY
                        $d[v] \leftarrow d[u] + 1$
                        $\pi[v] \leftarrow u$
                        $ENQUEUE(Q, v)$
        $color[u] \leftarrow$ BLACK
```

```
DFS($G$)
for each vertex $u \in V[G]$
        do $color[u] \leftarrow$ WHITE
           $\pi[v] \leftarrow$ NIL
$time \leftarrow 0$
for each vertex $u \in V[G]$
        do if $color[u]$ = WHITE
              then DFS-VISIT($u$)
```

```
DFS-VISIT($u$)
$color[u]$ = GRAY
$d[u] \leftarrow time \leftarrow time + 1$
for each $v \in Adj[u]$
        do if $color[v]$ = WHITE
              then $\pi[v] \leftarrow u$
                   DFS-VISIT($v$)
$color[u]$ = BLACK
$f[u] \leftarrow time \leftarrow time + 1$
```

```
MST-KRUSKAL($G$, $w$)
$A \leftarrow \emptyset$
for each vertex $v \in V[G]$
    do MAKE-SET($v$)
sort the edge of $E$ into nondecreasing order by weight $w$
for each edge $(u, v) \in E$, taken in nondecreasing order by $w$
    do if FIND-SET($u$) $\neq$ FIND-SET($v$)
         then $A \leftarrow A \cup \{(u, v)\}$
              UNION($u$, $v$)
return $A$
```

```
MST-PRIM($G$, $w$, $r$)
for each $u \in V[G]$
        do $key[u] \leftarrow \infty$
           $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
        do $u \leftarrow$ EXTRACT-MIN($Q$)
              for each $v \in Adj[u]$
                    do if $v \in Q$ and $w(u, v) < key[v]$
                          then $\pi[v] \leftarrow u$
                               $key[v] \leftarrow w(u, v)$
```