CSC 413 Introduction to Computer Algorithms

Spring 2021

Final Examination (Take-home)

Release Time: 10:00 AM, April 15

Due Time: 11:59 PM, April 25

Total points: 100

NAME Jackie Diep

STUDENT ID# W10076331

1. (20 points) **Filling the blanks** (1 point for each blank).

(1) Like divide and conquer, dynamic programming (DP) solves problems by combining solutions to subproblems: true or false? **True**; unlike divide and conquer, DP subproblems are independent: true or false? **False**.

(2) For a graph *G*=<*V*, *E*>, the total running time of BFS is O(**V + E**); the total running time of DFS is θ(**V + E**). If *G* is an undirected graph, $\sum_{v \in V} degree(v) =$ **2e**; if *G* is a directed graph, $\sum_{v \in V} degree(v) =$ **|E|**.

(3) Heap sort algorithm is stable or not? **not**, in-place or not? **in-place**. Its worst case time complexity is θ(   ). **nlogn** The height of a heap $A[1..n]$ is **floor($\log_2$n) + 1**.

(4) For a sorting algorithm for an array of *n* elements, in the worst case the number of inversions is **(n(n-1))/2**, the average number of inversions is **(n(n-1))/4**. Shell sort algorithm is stable or not? **not**, in-place or not? **in-place**.

(5) The worst case time complexity of *BuildMaxHeap* is O(**n**). The worst case time complexity of *MaxHeapify* is O(**lgn**). The worst case time complexity of Heap-Extract-Max is O(**lgn**). The worst case time complexity of Heap-Insert is O(**lgn**).

(6) Dynamic programming has four steps:

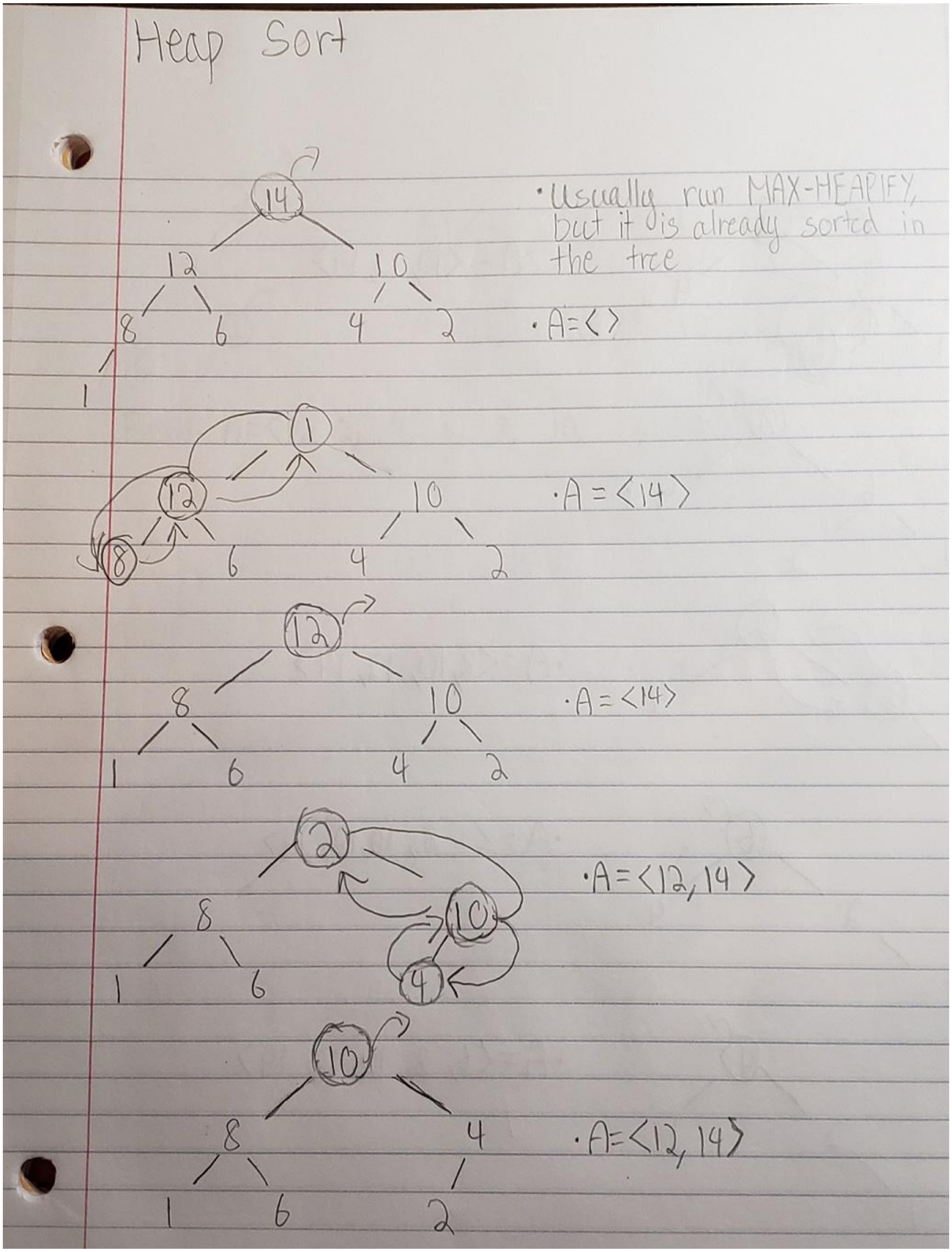
(1) **Characterize the structure of an optimal solution**;

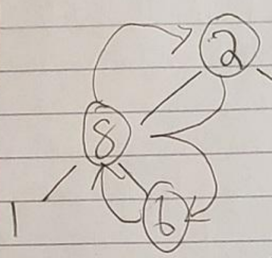
(2) **Recursively define the value of an optimal solution**;

(3) Compute optimal solution values either top-down with caching or bottom-up in a table;

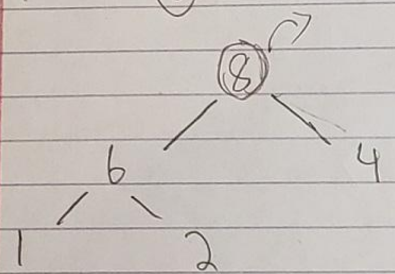
(4) Construct an optimal solution from computed values.

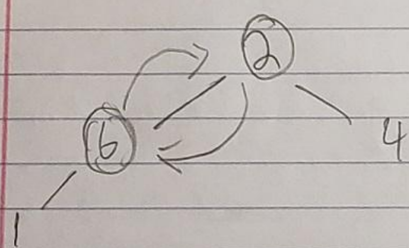
2. (20 points). **Heap Sort applications.**
   Illustrate the operations of Heapsort on the following array: A = <14, 12, 10, 8, 6, 4, 2, 1>. For each iteration, use binary trees or arrays to show the intermediate data structures or results.



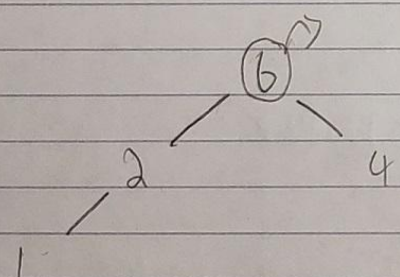


$\cdot A = \langle 10, 12, 14 \rangle$

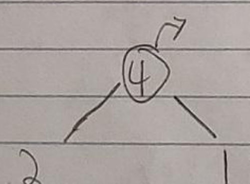




$\cdot A = \langle 8, 10, 12, 14 \rangle$



$\cdot A = \langle 8, 10, 12, 14 \rangle$



$\cdot A = \langle 6, 8, 10, 12, 14 \rangle$

2

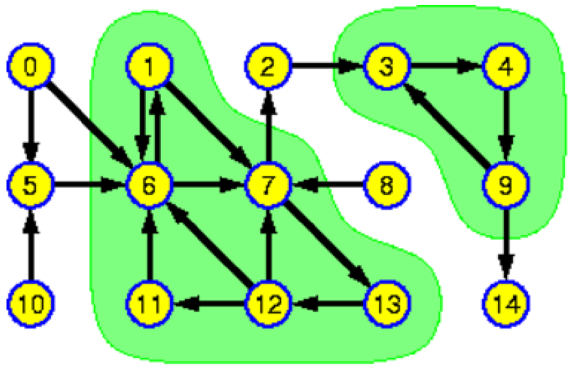
$\cdot A = \langle 4, 6, 8, 10, 12, 14$

1

$\cdot A = \langle 2, 4, 6, 8, 10, 12, 14 \rangle$

Final: $A = \langle 1, 2, 4, 6, 8, 10, 12, 14 \rangle$

3. (20 points). **Graph Algorithm.**
   (1) (10 points) Show the *d* and π values that result from running breadth-first search on this following graph using **vertex 0** as the source.
   (2) (10 points) Assume the depth-first search (DFS) procedure considers the vertices in **numerical** order, and each adjacency list is already ordered **numerically**.
      a) (6 points) Show the discovery and finishing times for each vertex.
      b) (2 points) Write the classification of each edge for the depth-first search.
      c) (2 points) Show the parenthesis structure of the depth-first search.



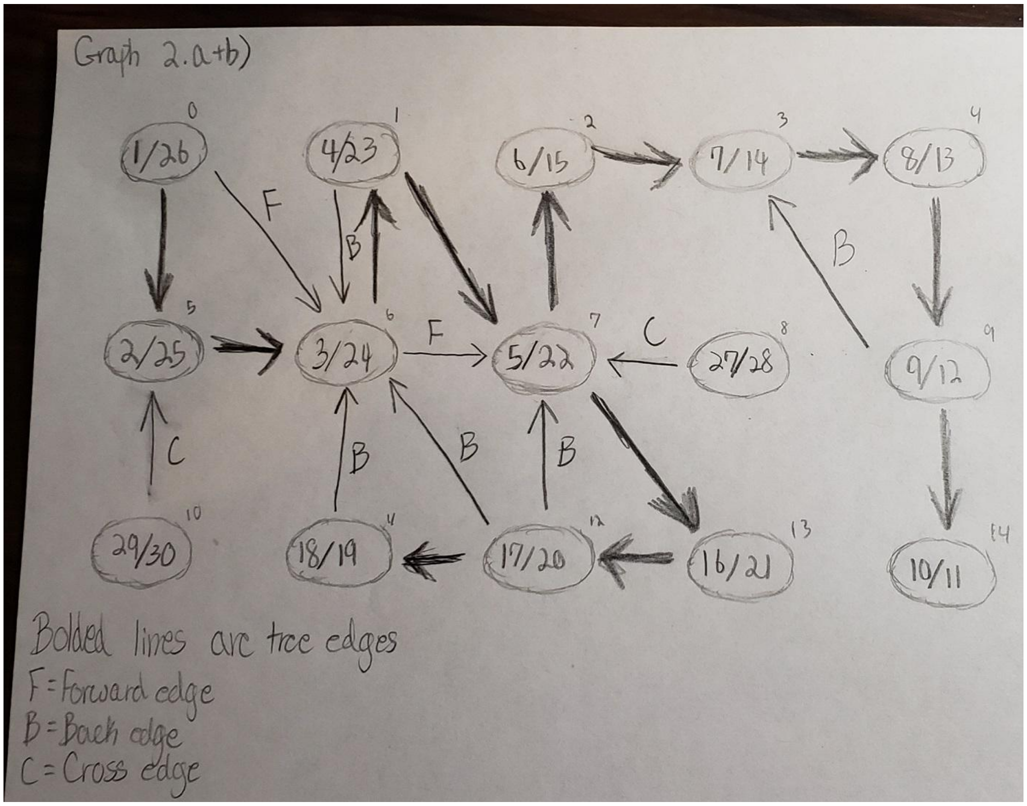
1)

Vertex	π	d
0	NIL	0
5	0	1
6	0	1
1	6	2
7	6	2
2	7	3
13	7	3
3	2	4
12	13	4
4	3	5
11	12	5
9	4	6
14	9	7

8	**NIL**	∞
10	**NIL**	∞

2)

(a + b)



c)

{0, 5} -> Tree Edge
{5, 6} -> Tree Edge
{6, 1} -> Tree Edge
{1, 7} -> Tree Edge
{7, 3} -> Tree Edge
{2, 3} -> Tree Edge
{3, 4} -> Tree Edge
{4, 9} -> Tree Edge
{9, 14} -> Tree Edge
{7, 13} -> Tree Edge
{13, 12} -> Tree Edge

{12, 11} -> Tree Edge
{12, 7} -> Tree Edge

{0, 6} -> Forward Edge
{6, 7} -> Forward Edge

{1, 6} -> Back Edge
{9, 3} -> Back Edge
{12, 6} -> Back Edge
{11, 6} -> Back Edge
{12, 7} -> Back Edge

{10, 5} -> Cross Edge
{8, 7} -> Cross Edge

4. (20 points). **Dynamic Programming:** Determine a Longest Common Subsequence (LCS) for the following two strings using dynamic programming approach. You need to illustrate the **step-by-step** procedure based on a **table** and also illustrate the **path** to reconstruct the LCS you have found by drawing lines through the centers of the grids on the path in the table.

'HIEROGLYPHOLOGY' vs. 'MICHAELANGELO'

LCS: I E G L O

	x	M	**I**	C	H	A	**E**	L	A	N	**G**	E	**L**	**O**
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	1	1	1	1	1	1	1	1	1	1
I	0	0	**1**	1	1	1	1	1	1	1	1	1	1	1
E	0	0	1	1	1	1	**2**	2	2	2	2	2	2	2
R	0	0	1	1	1	1	2	2	2	**2**	2	2	2	2
O	0	0	1	1	1	1	2	2	2	**2**	2	2	2	3
G	0	0	1	1	1	1	2	2	2	2	**3**	3	3	3
L	0	0	1	1	1	1	2	3	3	3	3	3	**4**	4
Y	0	0	1	1	1	1	2	3	3	3	3	3	**4**	4
P	0	0	1	1	1	1	2	3	3	3	3	3	**4**	4
H	0	0	1	1	2	2	2	3	3	3	3	3	**4**	4
O	0	0	1	1	2	2	2	3	3	3	3	3	4	**5**
L	0	0	1	1	2	2	2	3	3	3	3	3	4	**5**
O	0	0	1	1	2	2	2	3	3	3	3	3	4	**5**
G	0	0	1	1	2	2	2	3	3	3	4	4	4	**5**
Y	0	0	1	1	2	2	2	3	3	3	4	4	4	**5**

5. (20 points). **Minimum Spanning Trees (MST):** Finding a Minimum Spanning Tree for the following graph based on each of the following algorithm. You need to show the procedures **step-by-step.** You could directly draw the final MST but indicate the sequence of your search by writing a series of **letters**, i.e. (a), (b), (c)… under the edges of the MST. This type of answer is preferred. Or else, you need to draw a graph for each step **separately**.
   (a) (10 points) Kruskal's algorithm.
   (b) (10 points) Prim's algorithm (start with the node '**ORD**).

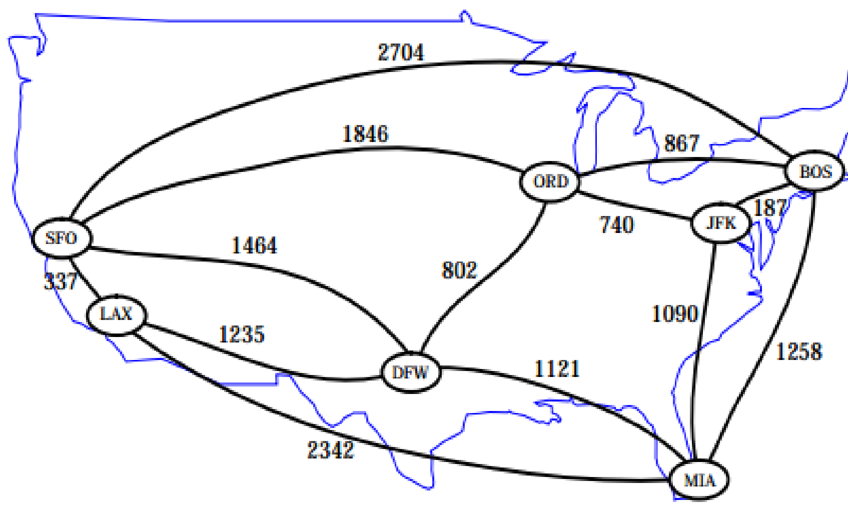
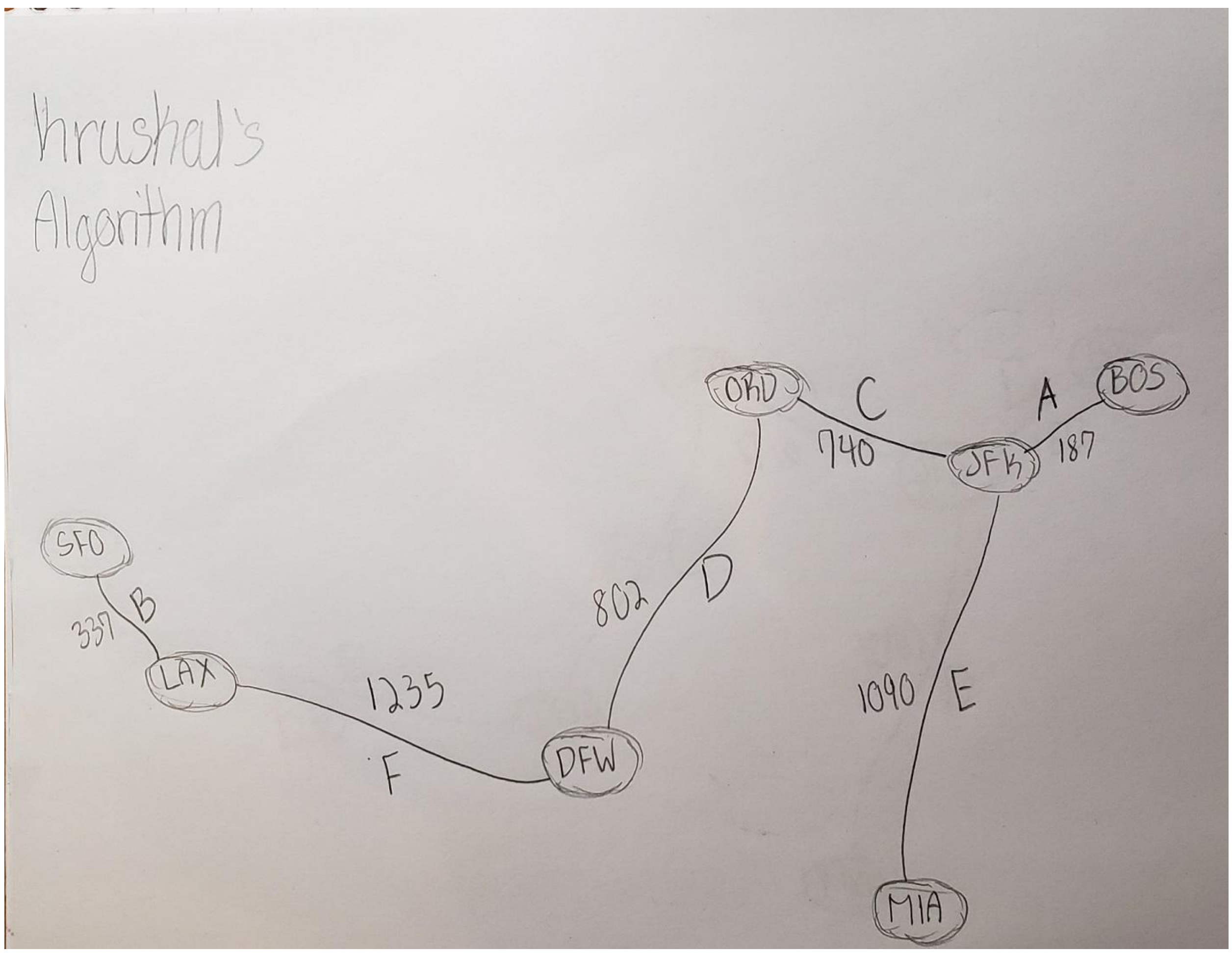


Fig. 1. A weighted graph whose vertices represent major U.S. airports and whose edge weights represent distances in miles.

a) **Kruskal's Algorithm**
Sort by weights:
BOS - JFK = 187
SFO - LAX = 337
ORD - JFK = 740
ORD - DFW = 802
~~ORD - BOS = 867~~
JFK - MIA = 1090
~~DFW - MIA = 1121~~
LAX - DFW = 1235
~~BOS - MIA = 1258~~
~~SFO - DFW = 1464~~
~~SFO - ORD = 1846~~
~~LAX - MIA = 2342~~
~~SFO - BOS = 2704~~



b) Prim’s Algorithm