CS 58000_01/02I Design, Analysis and Implementation Algorithms (3 cr.)

Assignment As_03 (Exam 02)

Student Name: _____

This assignment As_03 is due at 10:30 am, Tuesday, December 13, 2022. Please submit your assignment to Brightspace (purdue.brightspace.com). No late turn-in is accepted. Please write your name on the first page of your assignment. Your file name should be your last name such as Ng.docx. Please number your problem-answer clearly such as Problem 1a, 1b, 1c, Problem 2a, 2b, …. The problems' answers must be arranged in order. Please answer your questions using only a Word file (.docx file only). No pdf file will be accepted. Without using a Word file (.docx file) the submitted problems' answers would not be graded, or take 10 points off.

The total number of points for this Assignment_03 (Exam 02) is 340 points.

Problem 1[30 points]:

1a. Show that for any real constants $a < 0$ and $b > 0$,

$$(n + a)^b = \Theta(n^b).$$

1b. Explain why the statement "The running time of algorithm A is at least $O(n^2)$" is meaningless.

1c. Is $2^{n+1} = O(2^n)$? Justify your answer.

Is $2^{2n} = O(2^n)$? Justify your answer.

Problem 2 [30 points]:

Order of the following functions according to their order of growth (from the lowest to the highest)

$(n-2)!$, 2^{2n}, $0.002\,n^4 + 3n^2 + 1$, 2^n, e^n, $n2^n$, $1n^2n$, $\sqrt[3]{n}$, 3^n, $2^{\log n}$, n^2, $4^{\log n}$, $\sqrt{\log n}$

{Hint: $1n^2\,n = (\log_e n)(\log_e n)$ where $e = 2.71828$.}

Problem 3[40 points]:

Construct the string-matching automaton for the pattern P = aaabbaba over the alphabet Σ = {a, b, x |x is any letter other than a and b}; and illustrate its operation on the text string T = aaaabbabaaabbaaabbabaab.

3a. Construct the string-matching automation for the pattern P over the alphabet Σ = {a, b, x}in terms of the state transition table (Complete the state transition table)

	input			P
state	a	b	x	
0				
1				
2				
3				
4				
5				
6				
7				
8				

3b. Show the operation on the text string T, computed by the state transition table.

Complete the following table, in which T[i] is the letter at the position i of the text string; and State Φ(T[i]) stands for the state transition Φ (s, T[i]) = s'.

i																								
T[i]																								
State Φ(T[i])																								

3c. Complete the following sentence.

The result is _____ matches of the pattern at shift = ____ (i = ___) and shift = ___ (i = _____). Note that shift = i – 1.

3d. Draw a state transition diagram for a string-matching automaton for the pattern P over the alphabet $\Sigma = \{a, b, x \mid x \text{ is any letter other than a and b}\}$.

Problem 4[30 points]

Consider the following Algorithm Quicksort:

```
Algorithm  Quicksort(A[p .. r])
    //Quicksort(A[0 .. n - 1]) is the initial call for sorting an entire array A.
    Input:   A subarray  A[p .. r] of  A[0 .. n-1], defined by its left and right indices p and r.
    Output: Subarray  A[p .. r]  sorted in nondecreasing order
    {
    if (p < r )
              {s ← Partition(A[p .. r]);  //s ← j is a split position
               Quicksort(A[p .. s-1]);    //there is s - p elements
               Quicksort(A[s+1 .. r]);}   //there is r - s elements
    }//end of Quicksort()
```

```
Algorithm Partition(A[p .. r])
    //Partitions a subarray by using its first element as a pivot.
    Input:    A subarray A[p .. r] of A[0 .. n-1], defined by its left and right indices p and r,
              (p < r).
    Output:  A partition of A[p .. r], with the split position returned as this function's value
    {
    x ← A[p];        //set x be the leftmost element of A[p .. r].
    i ← p; j ← r+1;  //set left and right pointers pointing at p and r+1
    repeat
        repeat i ← i+1 until A[i] ≥ x; //move i towards right until …
        repeat j ← j-1 until A[j] ≤ x; //move j towards left until …
        swap(A[i], A[j]);
    until i ≥ j;
    swap(A[i], A[j]);
    swap(A[p], A[j]);
    return j;
    }
```

In the Algorithm Partition(A[p .. r]), there are three swap() procedures.

4a. When and why the first swap(A[i], A[j]) is needed?

4b. When and why the second swap(A[i], A[j]) is needed?

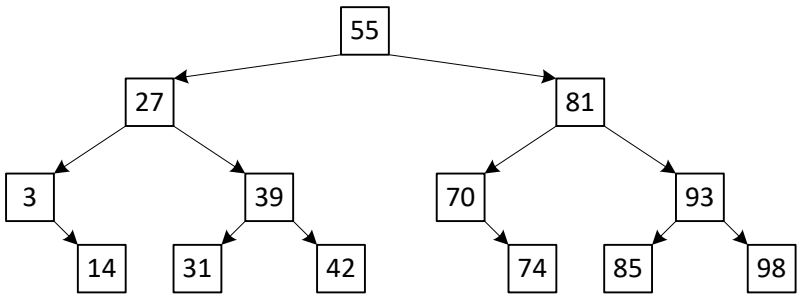
4c. When and why the third swap(A[p], A[j]) is needed?

Problem 5 [70 points]

Given the following array A[0..15] contains 13 elements.

	3	14	27	31	39	42	55	70	74	81	85	93	98		

It is helpful when answering these questions 5a through 5d to recognize what an equivalent binary search tree looks like:



5a. What is the largest number of key comparisons made by binary search in searching for a key in the following array?

5b. List all the keys of this array that will require the largest number of key comparisons when searched for by binary search.

5c. Find the average number of key comparisons made by binary search in a successful search in this array. (Assume that each key is searched for with the same probability.)

5d. Find the average number of key comparisons made by binary search in an unsuccessful search in this array. (Assume that searches for keys in each of the 14 intervals formed by the array's elements are equally likely.)

5e. Assume that the arrival of the elements is in the order 3, 14, 27, …., 98 of a given array A[0..15]. Rearrange the contents of 13 elements such that array A forms an AVL tree. Show step-by-step in terms of the intermediate resulting arrays.

5f. What is the largest number of key comparisons in searching for a key in array A which has an AVL tree?

5g. List all the keys of this array that will require the largest number of key comparisons when searched for by binary search.

Problem 6 [20 points]

Given the following array A[0..15] contains 13 elements.

	3	14	27	31	39	42	55	70	74	81	85	93	98		

6a. Use Max Heapify(A, i), 0< i to maintain given array A[0..15] a max-heap.

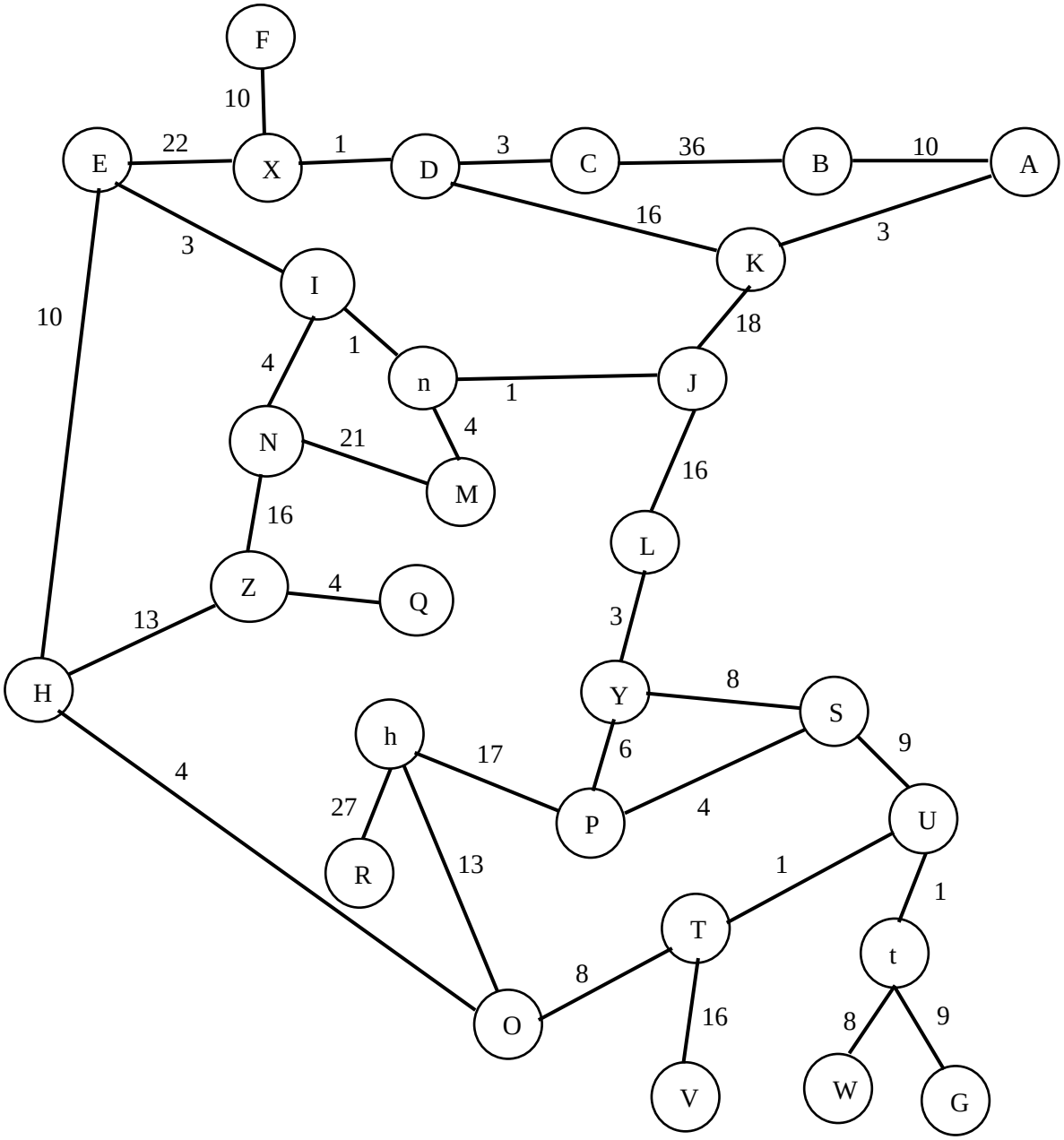
6b. Using the resultant max-heap array obtained from 6a, apply the heapsort algorithm to obtain a sorted array A in descending order. Show step-by-step in terms of the intermediate resulting arrays.

Problem 7 [50 points]

Given a weighted graph G, which is as follows:

7a. Construct

(i) a weighted adjacency list and
(ii) a weighted adjacency matrix



Traversing the given graph, based on its **weighted adjacency list** representation obtained in problem 7a(i), construct its **depth-first search tree** forest **starting from vertex A.** In your obtained DFS tree forest, show the **tree edges** (indicated as solid lines) and **back edges** (indicated as dotted lines) for your trees. **Traversal's stack** contains symbols (such as $V_{i,j}$, the first subscript number indicates the order in which a vertex V was first visited, say at i, (pushed onto the stack, V), where $0 < i \leq n$; the second one indicates the order in which it became a dead-

end, say at j (popped off the stack V), where 0 < j < n. n is the total number of vertices for the given graph. For simplicity's sake, please **use two time-stamps**: one is $0 < i \leq n$, the order for **pushing a vertex onto the stack** counting from 1 through n. The other one is $0 < j \leq n$, the order for **popping off a vertex from the stack** counting from 1 through n. For this problem, you need to answer 7b through 7e, which are as follows:

7b. Show the traversal's stack with time-stamp, and what are the orderings of vertices yielded by the DFS?
7c. Construct the corresponding depth-first search (DFS) tree forest, with indications of tree edges and back edges.
7d. What is the graph called? Is this graph acyclic? Does the graph have articulation points? What is the topological sort ordering for the graph?
7e. What are the time efficiency and space efficiency of the DFS?

Problem 8 [40 points]

Traversing the graph given in Problem 7, based on its **weighted adjacency list** representation obtained in Problem 7a(i), construct its **breath-first search (BFS) tree** forest **starting from vertex A**. For this, you need to use a **queue** (note the *difference from DFS*) to trace the operation of breadth-first search, indicating the order in which the vertices {…, V', V'', …} were visited. i.e., the order of the operation of **adding several vertices to**, or **removing a vertex from the queue** {V_i'',…, V'_{i+1}, V'_{i+2}, …}. The order in which vertices are added to the queue (i.e., enqueue operation) is the same order in which they are removed from it (i.e., dequeue operation). Indicate the **tree edges** (indicated as solid lines) and **cross-edges** (indicated as dotted lines) for your trees. For this problem, you need to answer 8a through 7e, which are as follows:

8a. Show the traversal's queue with a time-stamp indicating the order in which the vertices were visited, and what is the ordering of vertices yielded by the BFS?
8b. Construct the corresponding breadth-first search (BFS) tree forest, with an indication of tree edges and cross edges in addition to back edges and forward edges)
8c. From the obtained BFS tree forest, compute the *shortest distance (smallest number of edges) from* A to vertex G.
8d. What are the time efficiency and space efficiency of the BFS?

Problem 9 [30 points]

From the given graph in Problem 7, given a source vertex A, use Prim's algorithm to find the minimum spanning tree for the graph. For each step, state your **tree vertices V_T and the remaining vertices V - V_T**. More importantly, you need to give a table stating the tree vertices and remaining vertices with their weights (i.e., the corresponding edges with their weights). You do not have to give the intermediate graph as an "Illustration", but you show the final minimum spanning tree (via highlight edges) within the graph given in problem 7.

9a. Compute the minimum spanning tree of the graph given in Problem 7.

Tree Vertices	Remaining Vertices	Illustration (optional)
A(-, -)	B(A, 10), K(A, 3), ?(-, ∞)	
. . .	. . .	

where V_T = { A } and

V - V_T = { B, K, ?}, where "?" is to denote any vertex in the graph, which is not adjacent to A.

If you wish, use the symbol "?" to denote all the vertices in the graph which are not adjacent to every vertex in V_T.

9b. What is your obtained minimum spanning tree with their total weights of branches for the graph given in problem 7? [Highlight the obtained (from 9a) minimum spanning tree in the given graph of Problem 7. For example, the branch of A, K, D, X, F has a weight Secondly, compute the total weight of each branch of the minimum spanning tree. Thirdly, compute the grand total weight of the obtained minimum spanning tree.]

9c. From your obtained minimum spanning tree, what is the minimum distance from vertex A to vertex G?

Note: Good handwriting is required if you provide your answer in your handwriting. Proper numbering of your answer to each problem is strictly required. The problem's solution must be orderly given. (10 points off if not)
