CS25100 Homework 2: Spring 2017

Due Monday, April 17, 2017, before 11:59 PM. Please edit directly this document to insert your answers. You can use any remaining slip days for his homework. Submit your answers on Vocareum.

1. True/False Questions (18 pts)

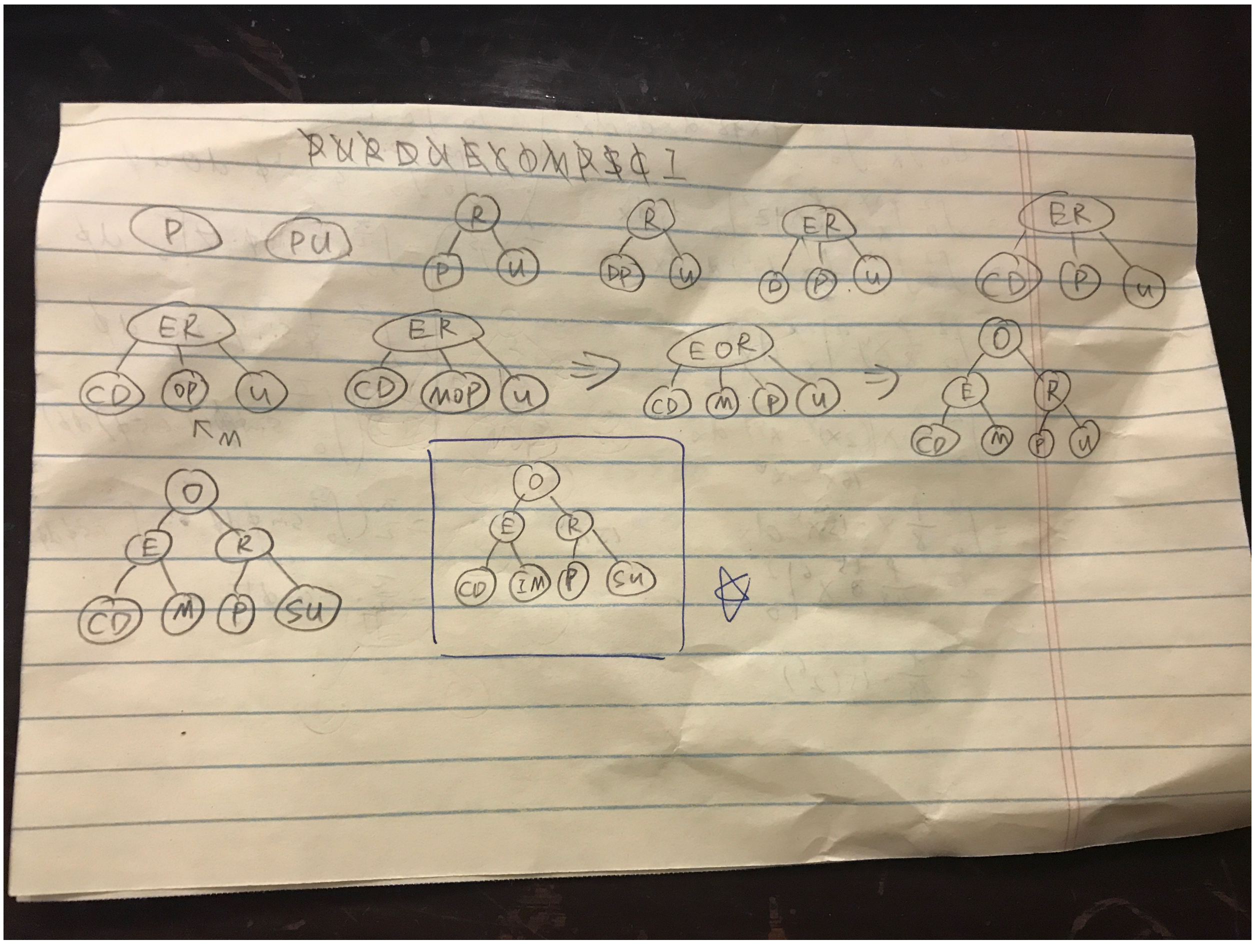
1. _F_ The reverse postorder of a digraph's reverse is the same as the postorder of the digraph.
2. _F_ Adding a constant to every edge weight does not change the solution to the single-source shortest-paths problem.
3. _T_ An optimization problem is a good candidate for dynamic programming if the best overall solution can be defined in terms of optimal solutions to subproblems, which are not independent.
4. _T_ If we modify the Kosaraju algorithm to run first depth-first search in the digraph G (instead of the reverse digraph G^R) and the second depth-first search in G^R (instead of G), the algorithm will find the strong components.
5. _T_ If you insert keys in increasing order into a red-black BST, the tree height is monotonically increasing.
6. _F_ A good hash function should be deterministic, i.e., equal keys produce the same hash value.
7. _T_ In the situation where all keys hash to the same index, using hashing with linear probing will result in $O(n)$ search time for a random key.
8. _F_ Hashing is preferable to BSTs if you need support for ordered symbol table operations.
9. _T_ In an adjacency list representation of an undirected graph, v is in w's list if and only if w is in v's list.
10. _F_ Every directed, acyclic graph has a unique topological ordering.
11. _F_ Preorder traversal is used to topologically sort a directed acyclic graph.
12. _T_ MSD string sort is a good choice of sorting algorithm for random strings, since it examines $N \log_R N$ characters on average (where R is the size of the alphabet).
13. _F_ The shape of a TST is independent of the order of key insertion and deletion, thus there is a unique TST for any given set of keys.
14. _T_ In a priority queue implemented with heaps, N insertions and N removeMin operations take $O(N \log N)$.
15. _T_ An array sorted in decreasing order is a max-oriented heap.
16. _T_ If a symbol table will not have many insert operations, an ordered array implementation is sufficient.
17. _T_ The floor operation returns the smallest key in a symbol table that is greater than or equal to a given key.

18. _T_ The root node in a tree is always an internal node.

2. Questions on Tracing the Operation of Algorithms (30 pts)

1. (4 pts) Draw the 2-3 tree that results when you insert the following keys (in order) into an initially empty tree:

   P U R D U E C O M P S C I



2. (5 pts) Give the contents of the hash table that results when you insert the following keys into an initially empty table of $M = 5$ lists, using separate chaining with unordered lists. Use the hash function $11k \bmod M$ to transform the k-th letter of the alphabet into a table index, *e.g.*, $hash(I) = hash(9) = 99 \% 5 = 4$. Use the conventions from Chapter 3.4 (new key-value pairs are inserted at the beginning of the list).

Answer:

	M	I	T	C	H	D	A	N	I	E	L	S
k	13	9	20	3	8	4	1	14	9	5	12	19
hash	3	4	0	3	3	4	1	4	4	0	2	4

Index	Chained list
0	T E
1	A
2	L
3	M C H
4	I D N S

3. (4 pts) List the vertices in the order in which they are visited (for the first time) in DFS for the following undirected graph, starting from vertex 0. For simplicity, assume that the Graph implementation **always iterates through the neighbors of a vertex in increasing order**. The graph contains the following edges:

```
0-1 1-2 1-7 2-0 2-4 3-2 3-4 4-5 4-6 4-7 5-3 5-6 7-8 8-6
```

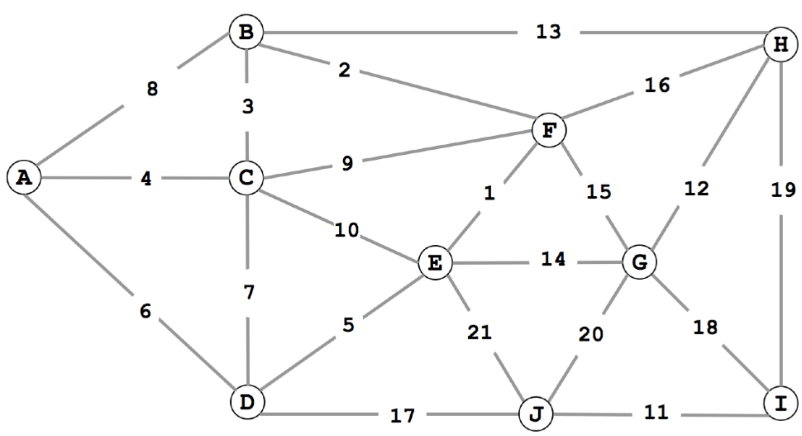
```
Answer: 0 1 2 3 4 5 6 8 7
```

4. (7 pts) Consider the following weighted graph with 10 vertices and 21 edges. Note that the edge weights are distinct integers between 1 and 21. Since all edge weights are distinct, identify each edge by its weight (instead of its endpoints).

(a) List the sequence of edges in the MST in the order that Kruskal's algorithm includes them (starting with 1)

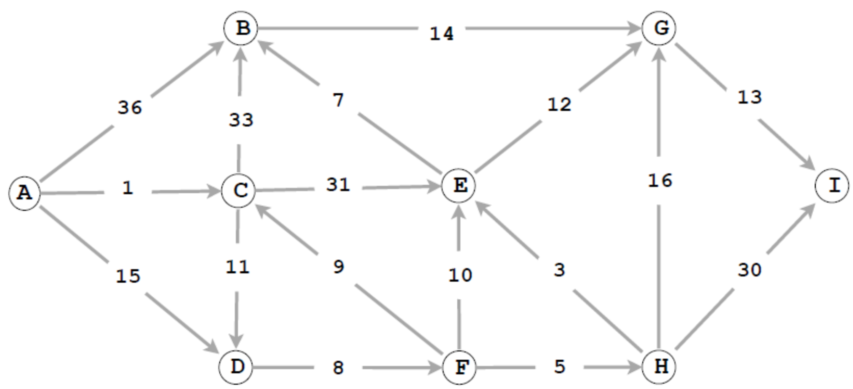
Answer: 1,2, 3, 4, 5,11, 12

(b) List the sequence of edges in the MST in the order that Prim's algorithm includes them. Start Prim's algorithm from vertex A.



Answer: 4,3,2,1,5,13,12,17,11

5. (5 pts) Consider the following weighted digraph and consider how Dijkstra's algorithm will proceed starting from vertex A. List the vertices in the order in which the vertices are dequeued (for the first time) from the priority queue and give the length of the shortest path from A to each vertex.



Vertex	A	C	_D_	_F_	_H_	_E_	_B_	_G_	_I_
Distance	0	1	_12_	_20_	_25_	_28_	_34_	_40_	_53_

6. (5 pts) Sort the 12 names below using LSD string sort. Show the result (by listing the 12 full words) at each of the four stages of the sort:

`John, Jane, Alex, Eric, Will, Nick, Jada, Jake, Nish, Luke, Yuan, Emma`

Answer:

4th char sort: Jada Emma Eric Jane Jake Luke Nish Nick Will John Yuan Alex

3rd char sort: Yuan Nick Jada Alex John Eric Jake Luke Will Emma Jane Nish

2nd char sort: Jada Jake Jane Nick Will Nish Alex Emma John Eric Yuan Luke

1st char sort: Alex Emma Eric Jada Jake Jane John Luke Nick Nish Wil Yuan

3. The Right Data Structure (8 pts)

Indicate, for each of the problems below, the best data structure from the following options: binary search tree, hash table, linked list, heap. Provide a brief justification for each answer.

1. Find the k^{th} smallest element.

   Heap. MaxHeap has O(1) for finding minimum element.

2. Find the last element inserted.

   Stack, LIFO

3. Find the first element inserted

   Queue. FIFO.

4. Guarantee constant time access to any element.

   Hash Table. Look up time is O(n)

4. Design/Programming Questions (34 pts)

1. (7 pts) Give the pseudocode or Java code for a linear-time algorithm to count the parallel edges in an undirected graph.

```
Identify parallel edges called adj_list;
Declear edges[n];
for I = 0 to n – 1 {
        edges[i] = -1;
}
count = 0;
for vertex v = 0 to n – 1 {
        for each vertex adjacent to v, adjacent E(adj_list[v]) {
                if edges[adjacent] = v {
                        count++;
                } else edges[adjacent] = v;
        }
}
count = count / 2;
```

2. (7 pts) Given an MST for an edge-weighted graph G and a new edge e, describe how to find an MST of the new graph in time proportional to V .

If edge isn't in MST
Old MST = MST of the updated graph
else
remove edge from MST and leaves two connected components
add the min weight edge with one vertex in each opponent

3. (10 pts) Design a data type that supports:
   - insert in logarithmic time,
   - find the median in constant time,
   - remove the median in logarithmic time.

   Give pseudocode of your algorithms. Discuss the complexities of the three methods. Your answer will be graded on correctness, efficiency, and clarity.

Using **MaxHeap**

```
insertion function {
Heap[++size] = element;
Int current = size;
While(Heap[current] > Heap[parent(current)] {
        Swap current and parent(current);
        Current = parent(current)
        }
}
```

Complexity: O(log(n))

```
sorting function {
int temp;
for i=0 to leap.length {
        if heap[i] != 0, sorted[i] = Heap[i]; // sorted is sorted array
for i = 0 to sorted.length
        for j = 1 to sorted.length
                if sorted[j-1] less than sorted[j]
                        temp = sorted[j-1] //swap elements
                        sorted[j-1] = sorted[j];
                        sorted[j] = temp;
}
```

Complexity: O(N^2)

```
remove function(int[] originalarray, int element#) {
int temp[] = new int[original array. Length – 1]
Arraycopy(original, 0, temp, 0, element);
Arraycopy(original, element#+1, temp, element, original array.length – element# - 1)
}
```

To remove median:

new array = remove(sortedarray, mid);   // mid = len/2

Complexity: O(log(n))

4. (10 pts) The 1D nearest neighbor data structure has the following API:
   - `constructor`: create an empty data structure.
   - `insert(x)`: insert the real number x into the data structure.
   - `query(y)`: return the real number in the data structure that is closest to y (or null if no such number).

   Design a data structure that performs each operation in logarithmic time in the worst- case. Your answer will be graded on correctness, efficiency, clarity, and succinctness. You may use any of the data structures discussed in the course provided you clearly specify it.

   Answer: