CS 321 Data Structures (Fall 2016)

Homework #3, final exam preparation

- **Q1(11 points): Running time, asymptotic notations and recurrence equations**

(a)(3 pts)(Multiple choice) If an algorithm's running time can be expressed as a function $f(n) = \sqrt{n} + \log n^n$, then which one of the following is NOT a correct bound for the running time?

1. $O(n^2)$
2. $O(\sqrt{n})$
3. $O(\log n^n)$
4. $\Omega(n)$

(b)(3 pts) Assume an algorithm runs in $\Theta(n^2)$, then which one of the following is a wrong asymptotic notation for it?

1. $O(n^3)$
2. $O(n^2)$
3. $O(n)$
4. $\Omega(n^2)$

(c)(5 points) Please re-arrange (sort) the following functions based on their growth rates, from the least to the greatest. In addition, please underline those functions which have the same asymptotic bound.

$$2^{n/2},\ \log_e n,\ n \log n,\ n!,\ 2^n,\ \log_2 n^2,\ \sqrt{n}$$

- **Q2(19 points): Basic Data Structures**

(a)(3 pts)(Multiple choice) Which one of the following is the FIRST true statement for linked lists (assume there is only one entry point to the list - `Head`)?

1. It takes linear time to insert an element in the front of a list.
2. It takes linear time to insert an element in the end of a singly circular list.
3. It takes linear time to insert an element in the end of a doubly non-circular list.
4. It takes linear time to insert an element in the end of a doubly circular list.

(b)(3 pts)(Multiple choice) If we read a sequence of items $< A, B, C, D, E, F >$ from a file and maintain some of the items in a stack by `push` and `pop` operations, which one of the following stack is possible? (Assume that each item can be pushed into the stack only once).

1. top $< D, E, F >$ bottom
2. top $< F, C, A >$ bottom
3. top $< C, D, E >$ bottom
4. top $< E, D, F >$ bottom

(c)(3 pts)(Multiple choice) For hashing by open addressing, we need to decide the table size m and some parameters so that the table can be fully utilized. What does that actually mean?

1. Allocate a table with size m equal to the number of elements stored.
2. Decide m and other parameters so that for any element, the initial m probe positions are distinct.
3. Decide m and other parameters to reduce the probability of collisions.
4. Decide m and other parameters so that no collision could happen.

(d)(10 pts) Suppose we would like to insert a sequence of numbers $< 22, 38, 7, 70, 46 >$ into a hash table with table size 8 using the three open addressing methods, with the primary hash function $h_1(k) = k \bmod 8$, the secondary hash function

$$h_2(k) = \begin{cases} 1 + (k \bmod 7) & \text{if } k \bmod 7 \text{ is even} \\ k \bmod 7 & \text{if } k \bmod 7 \text{ is odd} \end{cases}$$

and the constants $c_1 = c_2 = 1/2$ (in quadratic probing). Please insert numbers into the tables below.

index	linear	quadratic	double
0			
1			
2			
3			
4			
5			
6			
7			

- **Q3(18 points): AVL Trees**

  (a)(9 pts) What are the sequence of AVL tress after inserting each character in the list $< j, i, h, g, f, e, d, c, b, a >$ to an initially empty AVL tree? Note: please draw only one tree after each insertion.

(b)(9 pts) From the AVL tree you have built in part (b), what are the sequence of trees after deleting each character in the list (i.e., delete j , delete i, ...)? Note: please draw only one tree after each deletion.

- **Q4(7 points): Priority Queues**

  Using a Max-heap to implement a priority queue requires a procedure to decide the relative order of events in queue. An event a has a higher priority than another event b if a's priority value $>$ b's priority value. If both a and b have the same priority, then the event with earlier arrival time has a higher priority. Two events cannot arrive at the same time. Write a Compare procedure to decide the relative order of two events.

```
Compare(a, b)    // both a and b are events
// a.priority, b.priority: their priority values.
// a.time, b.time: arrival time (earlier time has a smaller numeric value)
// return 1 if a has a higher priority than b; Otherwise, return -1
{

}
```

- **Q5(7 points): Recursion in Tree Structures**

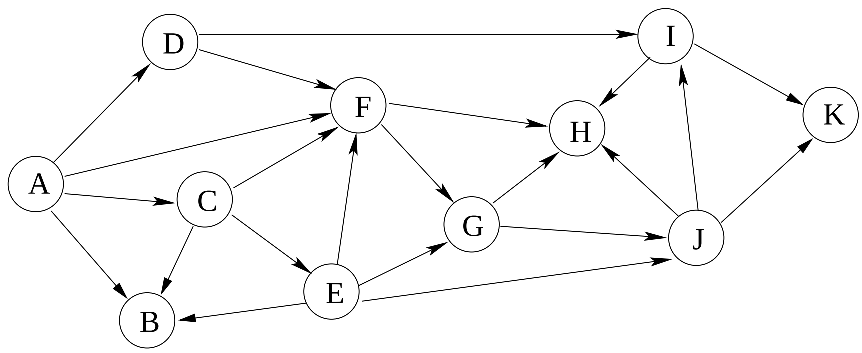
  Given a binary tree, please write a pseudocode to calculate the weight of the tree (sum of the weights of all edges in the tree). Let $w(u, v)$ denote the weight of the edge (u, v), where u is the parent of another node v.

```
Tree-Weight(x)    // x is the root of the tree
{

}
```

- **Q6(18 points): Graph Algorithms**

  A weighted and directed graph is given below.



(a)(4 pts) Based on the DAG above, please find a (any) discovering sequence of vertices in a BFS search if A is the source vertex.

(b)(4 pts) Based on the DAG above, please find a (any) discovering sequence of vertices in a DFS search.

(c)(4 pts) Based on the DAG above, please find a (any) topological sequence of vertices.

(d)(6 pts) Please find a (any) minimum spanning tree of the graph below