Faculty of Science
Final Examination

Computer Science 308-251B
Data Structures and Algorithms

Examiner:	Prof. Claude Crépeau	**Date:**	April 30, 2010
Associate Examiner:	Prof. Patrick Hayden	**Time:**	09:00 – 12:00

INSTRUCTIONS:

This examination is worth 50% of your final grade.
The total of all questions is 100 points.
Each question is assigned a value found in brackets next to it.
<u>O P E N • B O O K S •/• O P E N • N O T E S</u>
Faculty standard calculator permitted only.
This examination consists of 5 pages including title page.
This examination consists of 6 questions.

<u>SUGGESTION</u> : read all the questions and their values before you start.

[20%]

1) In an *Integer Minimum Spanning Tree* problem the input instance is a graph **G=(V,E)**, an integer **U** and an integer weight function, **w: E --> {0,1,2…,U}** (the weight of each edge is an integer between **0** and **U**). When considering the running time of algorithms solving this problem, you have three parameters to consider: **|V|=n** the number of vertices of your graph, **|E|** the number of edges of your graph and **U** the upper bound on the weights of the edges.

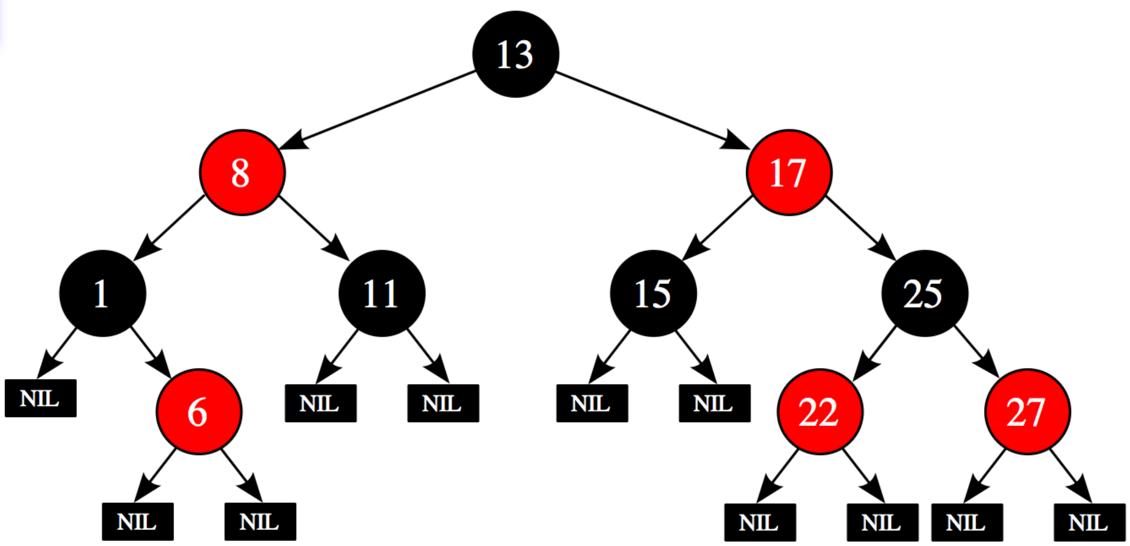
Discuss extensively, which algorithm should be favored for its efficiency at solving an Integer Minimum Spanning Tree problem as a function of **n, |E|** and **U**.

Example (*false*):

|E| is $\Theta(n^3)$	**U** is $\Theta(n \log n)$	Prim is better because it runs in time $\Theta(n^{17})$ while Kruskal is $\Theta(n^{42})$.
|E| is $\Theta(n)$	**U** is $\Theta(1)$	Kruskal is better because it runs in time $\Theta(n^{1/2})$ while Prim is $\Theta(n^{\log n})$.

[15%]

2) Consider the following red-black tree: (for clarity: nodes **8**, **17**, **6**, **22** and **27** are "red")



a) Find the MINIMUM and MAXIMUM of this tree.

b) Find the SUCCESSOR of node **11** and PREDECESSOR of node **17**.

c) Show the result of removing node **25**. Indicate colors.

d) Show the result of inserting a node for value **10**. Indicate colors.

[10%]

3) For each statement, say if it is *true* or *false*.

Correct = +1 pt,	**Incorrect = -0.5 pt,**	**No answer = 0 pt,**	**Minimum Total= 0 pt.**

(a) Let $\mathbf{f : N \rightarrow N}$ be the function $\mathbf{f(n) = \lfloor n / 5 \rfloor + 1}$. In an array of $\mathbf{n}$ unsorted elements we can find the element of rank $\mathbf{f(n)}$ in worst case time $\mathbf{O(n)}$.

(b) Any comparison-based sorting cannot possibly be correct on all inputs of size **n** if the time it takes **is not Ω(n log n)**.

(c) A dynamic programming algorithm can always run in time at most $\mathbf{O(n^2)}$.

(d) Insertion in a generic Binary Search Tree always takes time **O(n / log n)**.

(e) The data structure for Disjoint Sets is used by Dijkstra's algorithm.

(f) Simple problems always have simple (efficient) algorithms with simple running times.

(g) Johnson's algorithm for All-Pairs Shortest Paths uses both Bellman-Ford's and Prim's algorithms.

(h) Optimal Sub-structure is relevant to both Dynamic programming and Greedy algorithms.

(i) Suppose **T** is a MST. Let **e** be an edge of weight **w** not in **T**. If we add **e** to **T**, it closes a cycle **c**. Suppose there exists another edge **e'** of **c** with weight **w**. The new tree **T' = T – {e} ∪ {e'}** is also a MST.

(j) "Sorting out Sorting" is the title of an animation movie demonstrating several sorting algorithms.

[15%]

4) Prove using (constructive) induction that the following **T(n) is Θ(n)** :

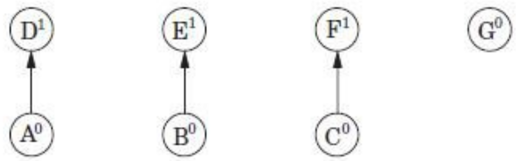
$$T(n) = \begin{cases} 1 & \text{if } n = 1, 2 \\ 3T(\lfloor n/5 \rfloor) + T(\lfloor n/4 \rfloor) + \Theta(n) & \text{if } n > 2 \end{cases}$$

[20%] 5)

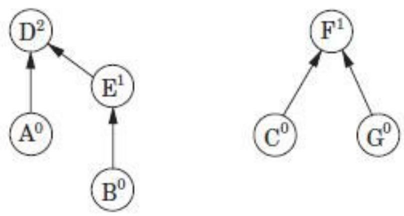
A sequence of disjoint-set operations. Superscripts denote rank.

1) After `makeset`(A), `makeset`(B), ..., `makeset`(G):

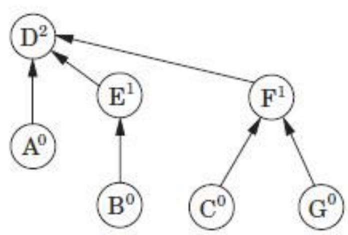
2) After `union`(A,D), `union`(B,E), `union`(C,F):



3) After `union`(   ), `union`(   ):



4) After `union`(B,G):



Consider the above sequence of operations on a disjoint-sets data structure.

A) In step **3)** I removed the arguments to the **union** operations. Give correct arguments to these two calls to **union** that would produce the observed result. If there is more than one solution to this question **GIVE ALL THE CORRECT SOLUTIONS.**

B) What is the set of sets represented after step **3)** ?

Example: the set of sets after step **1)** is **{ {A}, {B}, {C}, {D}, {E}, {F}, {G} }**.

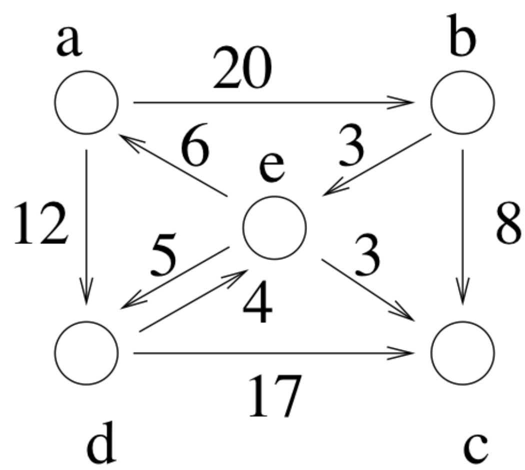
C) If we insert the operation **find-set(B)** before step **4)** what would be the result after **union(B,G)** as in step **4)** ?

D) This is the book version of **find-set** that implements the *path-compression* heuristic.

```
find-set(x)
if x ≠ x.p
        x.p = find-set(x.p)
return x.p
```

Write an iterative version of **find-set** instead of recursive as in the book.
Why should we prefer iterative ?

[20%] **6)** Consider the following directed graph:



In Chapter 25, we considered recursive definitions of two families of matrices, $\mathbf{L}^{(m)}$, which is computed in a way analogous to matrix multiplication, and $\mathbf{D}^{(m)}$, in the Floyd-Wharshall algorithm. Both of these algorithms assume the vertices are labeled as **{1,2,…,n}**; thus when considering the graph above you may assume **a=1 , b=2,…, e=5**.

A. Provide the definitions of $\mathbf{L}^{(m)}$ and $\mathbf{D}^{(m)}$.

B. Provide explicitly matrices $\mathbf{L}^{(0)}$ and $\mathbf{D}^{(0)}$.

C. Provide explicitly matrices $\mathbf{L}^{(2)}$ and $\mathbf{D}^{(3)}$.

D. What is the longest simple path in this graph ?
Why don't we have a book chapter on the longest simple path problem ??