CS 483 Analysis of Algorithms - Sample Exam

1. Asymptotic Bounds [20 pts]

1. Show that if $f(n) = O(g(n))$ then $g(n) = \Omega(f(n))$, using the appropriate definitions.

if $f(n) = O(g(n))$ then $f(n) \leq c_0 g(n)$ for some c_0 and n_0 such that the equality holds for a all $n \geq n_0$. The definition of $g(n) = \Omega(f(n))$ is $g(n) \geq c_1 f(n)$ which exactly as the inequality above when we divide both side with c_0 and hence $c_1 = \frac{1}{c_0}$

2. Show that $5 \log n = \Theta(\log(n^3))$, using the appropriate definitions.

$$c_0 \log(n^3) \geq 5 \log n \leq c_1 \log(n^3))$$

$$c_0 3 \log(n) \geq 5 \log n \leq c_1 3 \log(n))$$

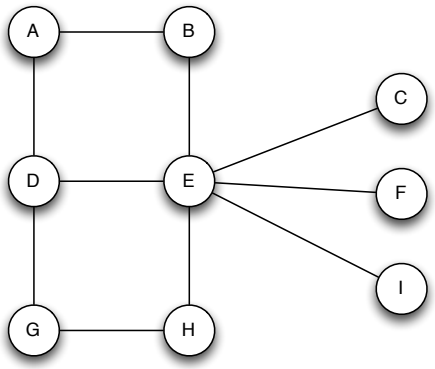
which holds for $c_0 = 1$ and for $c_1 = 2$ for all n greater then $n_0 = 1$.

3. For the following functions show whether $f_1 = O(f_2)$ or $f_1 = \Omega(f_2)$ or $f_1 = \Theta(f_2)$.
a) $\log n, n + \log n$, $\log n = O(n + \log n)$
b) $\log n, \sqrt{n}$, $\log n = O(\sqrt{n})$
c) $2^n, 3^{3/2}$, $2^n = \Omega(3^{3/2})$

2. Search in a Graph [15 pts]

Using the graph given below draw:

- A Depth First Search spanning tree beginning from vertex A.
- A Breadth First Search spanning tree beginning from vertex A.

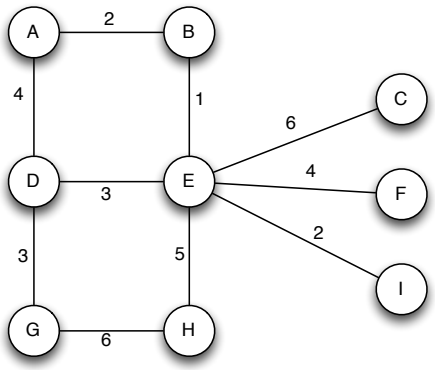


3. Minimum Spanning Trees [20 points]

Use Prim's algorithm to find a minimum spanning tree for the following graph. Start from vertex A, and show the edges added to the tree one by one.

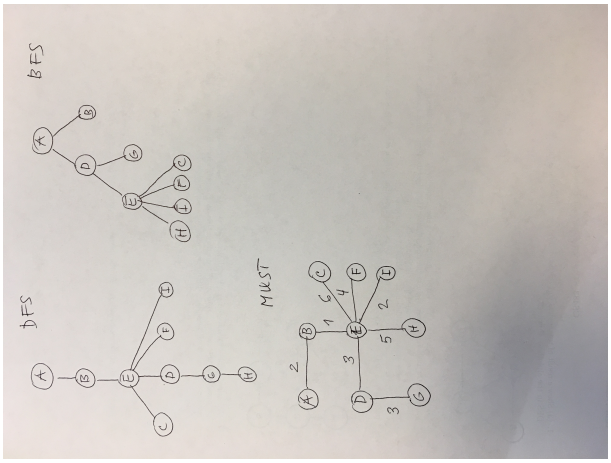
Show the final minimum spanning tree. What is its cost?

How many minimum spanning trees does the given graph have?



There is a single minimum weight spanning tree with the cost of 26, obtained by Prim's algorithm. You can write down the stages of the priority queue as the algorithm progresses to show your work in for Prim's algorithm. (see result on the next page).

In order to check whether there are additional MWST we could only obtain them by swapping some of the edges of the same cost with the edges which are already in the tree, without increasing the cost.



4. Binary/Ternary Search [15 pts]

Consider the binary search problem where we need to find whether an element with value k exist in the sorted array. The binary search algorithm splits the array A in half and then recursively calls itself on one of the halves. The recurrence for normal binary search is $T_1(n) = T_1(n/2) + 1$. Consider the variation of binary search where you initially split the array in the three sets of approximately same size. Write down the recurrence for the this ternary search algorithm, three levels of recursion tree and it's asymptotic time complexity (using a method of your choice. e.g. appealing to recursion tree).

The recurrence is :

$$T(n) = T(n/3) + O(1)$$

At each recursive call the number in question is checked with two elements of the array A[n/3] and A[2n/3] . If k is smaller then A[n/3] we look in the first third, if its greater then A[2n/3] then we look in the last, otherwise we look in the middle third. The comparisons take constant time. The recirsion tree has $\log_3 n$ levels, each level is $O(1)$ so the running time of the algorithm is $O(\log n)$.

5. Cycles in a Directed Graph [10 pts]

Give an $O(m + n)$ time algorithm to determine whether a given directed graph contains any cycles. This was already a homework problem.