Krishna Sai Chemudupati **COSC 3020_Assignment02**

Problem 1:

Part1:

1. It omits all the lower order terms like coefficients from the analysis. For example: $T_1(n) = n^2$, and $T_2(n) = 10000n^2 + 10000$ will follow the same asymptotic complexities.
2. During asymptotic analysis, we consider only very large values of n. For example: $T_1(n) = 0.01n^2$, and $T_2(n) = 0.0001n^3$
   Through asymptotic analysis, we say that the $T_2(n)$ is faster than $T_1(n)$. But, we are only considering high values of n for which $T_2(n)$ is faster than $T_1(n)$. In this example, $T_1(n)$ is faster than $T_2(n)$ till $n = 100$. Which means that, for values of n less than 100, algorithm1 will be more efficient. But, asymptotic analysis misleads us in this case.
3. It doesn't take into factor how much time it 'actually' takes for the CPU to complete a command. For example, if we are working with very large trees, not all memory locations can be loaded into the cache. Therefore, the memory must be retrieved from the main memory and it takes more time to do the same node accesses in the tree.

Part2:

Search in a balanced binary search tree follows the asymptotic complexity of O(logn). The run time depends on much more complicated function than the Big-O approximation as it neglects the lower order terms.

Given, binary search in a tree with 1000 elements takes 5 seconds. Below shown is my attempt to find the neglected lower order terms in the equation.

Let us assume that there is a constant k coefficient for (logn) which is omitted.

$$T(n) = k * log_2^n$$

In the information given, n=1000 resulted in 5 seconds.

$$T(n) = k * log_2^{1000} = 5$$

$$k * 9.97 = 5$$

$$k = \frac{5}{9.97} = 0.5017$$

Therefore,

$$T(n) = 0.5017 * log_2^n$$

Therefore the time taken for n=10,000:

$$T(n) = 0.5017 * log_2^{10000}$$

$$T(n) = 0.5017 * 13.2877$$

$$T(n) = 6.6664s$$

Therefore, it will take approximately 6.7 seconds to search in a tree with 10,000 elements.

Part3:

1. Tree wasn't well balanced. Mostly, the elements formed linked list structure (also known as a degenerate tree).
2. Mostly all elements in this tree were lesser than the element we are searching for (only a contributing factor).
3. There might be other processes running at the same time, or there is some other hardware issue like a failing hard drive.
4. Each node of the tree is stored in random memory locations, and as the tree gets large, the memory location of nodes might not be loaded onto the cache. This can cause further delay in processing the same search.

Problem 2:

Part1:

Definition of isomorphism: Two undirected graphs are isomorphic if they have the same structure. $\mathrm{G1} = (\mathrm{V1}, \mathrm{E1})$ is isomorphic to $\mathrm{G2} = (\mathrm{V2}, \mathrm{E2})$ if there is a one-to-one and onto function $\mathrm{f}: \mathrm{V1} \rightarrow \mathrm{V2}$ such that $(\mathrm{u}, \mathrm{v}) \in \mathrm{E1}$ iff $(\mathrm{f(u)}, \mathrm{f(v)}) \in \mathrm{E2}$.

Let there be two graphs G1, and G2 with *n* nodes each. Since both graphs are completely connected, every node is connected to every other node. Now below is the proof that G1, and G2 are isomorphic.

Graph 1: $V1 = \{v_1, v_2, v_3, \dots \dots . v_n\}$

$E1 = \{(v_1, v_2), (v_1, v_3) \dots . . (v_1, v_n), (v_2, v_3), (v_2, v_4), \dots . (v_2, v_n), \dots . (v_{n-1}, v_n)\}$

$= \{ \left(v_i, v_j\right) | 1 \leq \ i, j \leq n \ and \ i \neq j \ )$

$$no. of \ edges \ in \ Graph1 = k = \frac{n(n-1)}{2}$$

Graph 2: $V2 = \{v_1', v_2', v_3', \dots \dots . v_n'\}$

$E2 = \{(v_1', v_2'), (v_1', v_2') \dots . . (v_1', v_2'), (v_2', v_3'), (v_2', v_4'), \dots . (v_2', v_n'), \dots . (v_{n-1}', v_n')\}$

$= \left(v_i', v_j'\right), \ | 1 \leq \ i, j \leq n \ and \ i \neq j \ )$

Therefore, in this case too, $no. of \ edges \ in \ Graph2 = k = \frac{n(n-1)}{2}$

Consider a bijective function such that:

$f(v_i) = \ v_i'$, such that if $v_i \ \in \ \mathrm{V1}$ then $v_i' \ \in \ V2$

Therefore, if $\left(v_x, v_y\right) \in \ \mathrm{E1}$, then it implies that $\left(f(v_x), f\left(v_y\right)\right) \in \ \mathrm{E2}$

This is because, a completely connected graph has edges between every distinct node and since, v_x and v_y have an edge between them, we can say v_x and v_y are distinct. Since there exists a *one-one* function $f(v_i) = v_i'$, we can say that $f(v_x)$ and $f(v_y)$ must also be distinct nodes in G2. As, G2 is also a completely connected graph, and $f(v_x)$ and $f(v_y)$ are distinct, there must also exist an edge between them.

As the function $f(v_i) = v_i'$ is bijective, there must exist an inverse function such that $f^{-1}(v_i') = v_i$.

$$\text{Therefore, if } \left(f(v_x), f(v_y)\right) \in \text{ E2 , it implies that } \left(f^{-1}(f(v_x)), f^{-1}\left(f(v_y)\right)\right) \in \text{ E1}$$

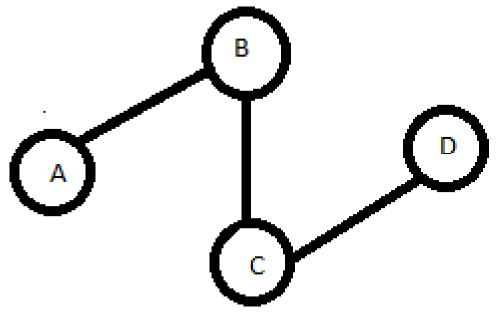
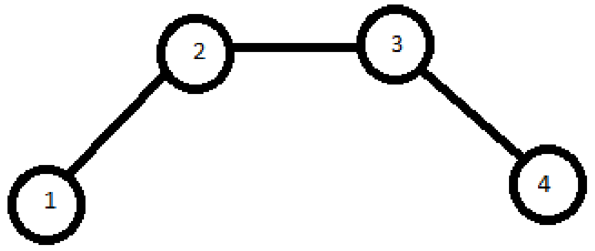
$$\left(f(v_x), f(v_y)\right) \in \text{ E2} \Rightarrow (v_x, v_y) \in \text{ E1}$$

We can prove this case using the same reasoning above.

From this information, we can say that the two graphs $\text{G1} = (\text{V1}, \text{E1})$, and $\text{G2} = (\text{V2}, \text{E2})$ are isomorphic and therefore there exists a one-one and onto function f: V1 $\rightarrow$ V2, such that $(v_x, v_y) \in$ E1 if and only if $\left(f(v_x), f(v_y)\right) \in$ E2 where $1 \leq x, y \leq n \text{ and } x \neq y$.

Part2:

Let us assume that for two graphs A and B to be isomorphic, they have to be completely connected. By proof by contradiction, we need just one case in which two non-completely connected graphs are isomorphic.

Graph 1	Graph 2
	

Consider these two graphs. There is a one-one matching between the vertices and respective edges of the two graphs.

Graph 1	Graph 2	Graph 1	Graph 2
A	1	(A, B)	(1, 2)
B	2	(B, C)	(2, 3)
C	3	(C, D)	(3, 4)
D	4		

This shows that the shown two graphs, Graph 1 and Graph 2 are isomorphic even though they are not completely connected. This contradicts the assumption made in the beginning of the proof. Therefore, it is proved that two graphs need not be completely connected to be isomorphic.

Problem 4:

$A - 1 \to B$ added.

$A - 1 \to B$

$E - 2 \to F$ added.

$A - 1 \to B$
$E - 2 \to F$

$A - 3 \to E$ added.

$A - 1 \to B$
$E - 2 \to F$
$A - 3 \to E$

$C - 3 \to D$ added.

$A - 1 \to B$
$E - 2 \to F$
$A - 3 \to E$
$C - 3 \to D$

$D - 4 \to E$ added.

$A - 1 \to B$
$E - 2 \to F$
$A - 3 \to E$
$C - 3 \to D$
$D - 4 \to E$

$E - 4 \to G$ added.

$A - 1 \to B$
$E - 2 \to F$
$A - 3 \to E$
$C - 3 \to D$
$D - 4 \to E$
$E - 4 \to G$

$A - 5 \to D$ cannot be added because it will form a cycle.

$C - 5 \to F$ cannot be added because it will form a cycle.

$F - 7 \to G$ cannot be added because it will form a cycle.

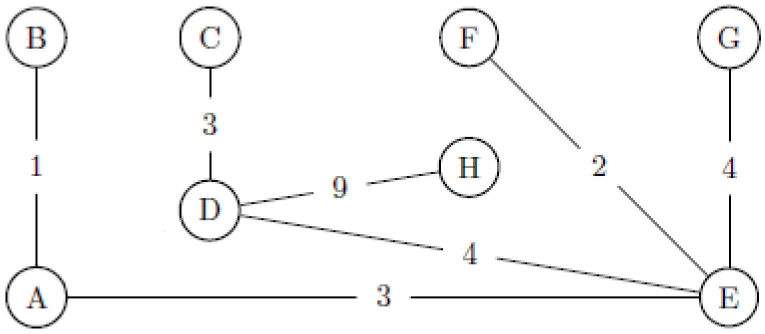
$D - 8 \to F$ cannot be added because it will form a cycle.

$B - 9 \to C$ cannot be added because it will form a cycle.

$D - 9 \to H$ added.

$A - 1 \to B$
$E - 2 \to F$
$A - 3 \to E$
$C - 3 \to D$
$D - 4 \to E$
$E - 4 \to G$
$D - 9 \to H$

Therefore the minimum spanning tree formed is:


B
C
F
G
1
3
H
2
4
9
D
4
A
3
E