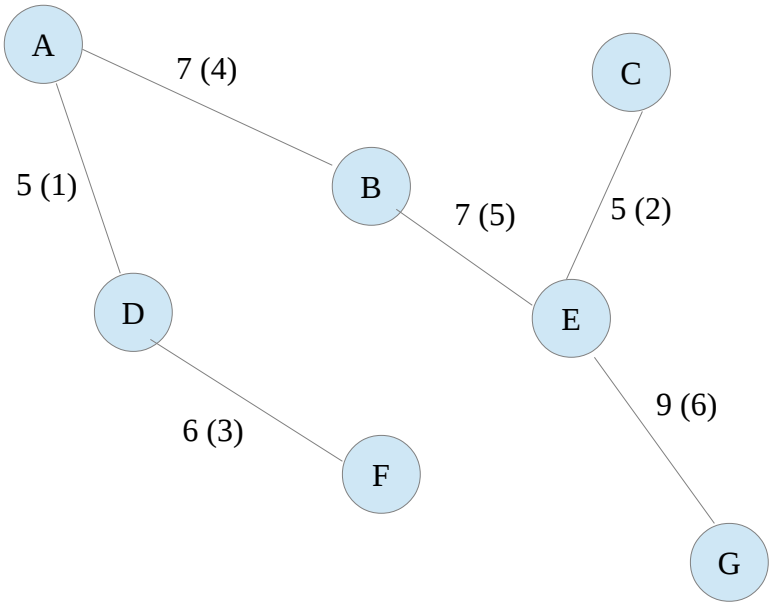
Joshua Rinaldi
101902285
10/22/2014
HW6

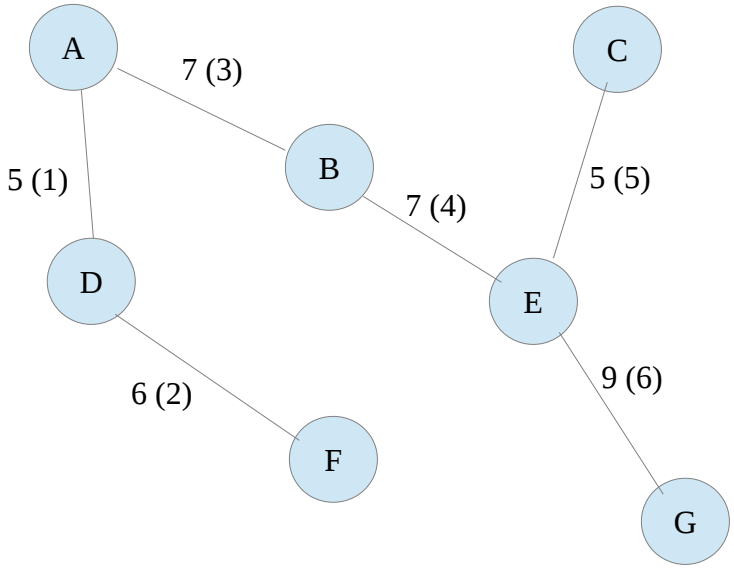
Honor Code Pledge: _____

Question 1

Kruskal's Algorithm (39)



Prim's Algorithm (39)



Question 2

In order to find the greatest value of $\prod_{i=1}^{n} a_i^{(b_i)}$ for two sets A and B, you would partner up the highest value in A with the highest value in B, then the second highest from A with the second highest from B and so on. So, you first sort A and B to go from highest to lowest. You then iterate over A and B and pair up the respective *i* from each set. So, given the sets *A = [10, 13, 9, 12]* and *B = [1, 3, 2, 6]*, the steps of this algorithm would look like this:

A = [10, 13, 9, 12] B = [1, 3, 2, 6]

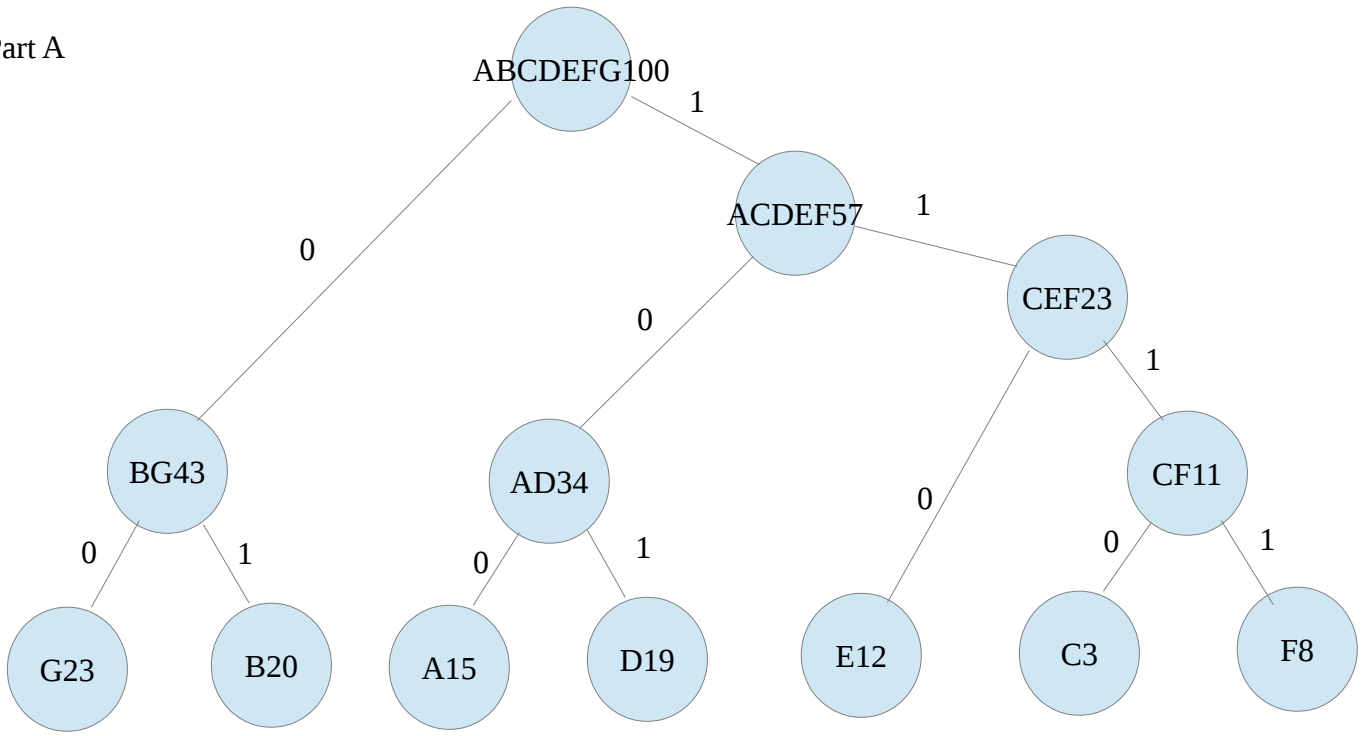
A = [13, 12, 10, 9] B = [6, 3, 2, 1]

$$\prod_{i=1}^{n} a_i^{(b_i)} = 13^6 * 12^3 * 10^2 * 9 = 7.51 * 10^{12}$$

In order to get the largest value of $\prod_{i=1}^{n} a_i^{(b_i)}$ you need to partner up the largest values of A with largest values of B. The best way to do this is to reorganize the arrays in order of largest to smallest. Otherwise, you would need to iterate over the arrays every time in order to find the largest values, and also remove those values from the arrays. This would increase the amount of time needed to run over the algorithm detailed above, where the arrays are organized and then iterated over.

Question 3

Part A



Part B

Character	Encoding
A	100
B	01
C	1110
D	101
E	110
F	1111
G	00

Part C

A = 0.15(1000000)(3) = 450000
B = 0.2(1000000)(2) = 400000
C = 0.03(1000000)(4) = 120000
D = 0.19(1000000)(3) = 570000
E = 0.12(1000000)(3) = 360000
F = 0.08(1000000)(4) = 320000
G = 0.23(1000000)(2) = 460000

Total number of bits required = 2680000

Question 4

(a) An algorithm looking for the earliest start time might not work, because if you can only complete so many task within the given time window, so if the job with the earliest start time were to run the whole length of the time window, it would not allow for any other jobs to start.

(b) An algorithm looking at the earliest end time would work. This is because the algorithm would look at which jobs end soonest, then the next job that ends earliest, without overlapping any of the previous jobs. So, the algorithm would provide essentially the shortest jobs in the queue. So, no jobs would be able to run for the entire length of the time window, so the most number of compatible jobs would be able to run.

(c) An algorithm that looks for the jobs with the shortest time interval would also not work. This is because the job with shortest interval might conflict with every other job in the queue, whereas there might be other jobs in the queue are compatible with each other, just not the job with shortest time interval.