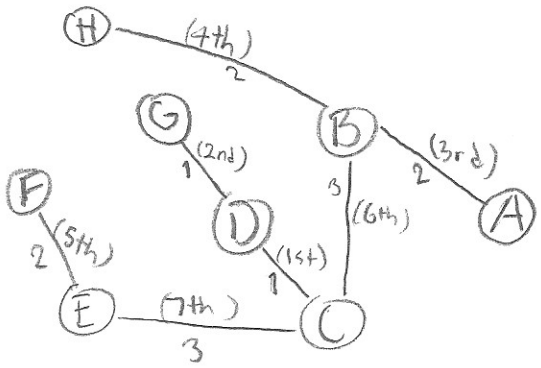
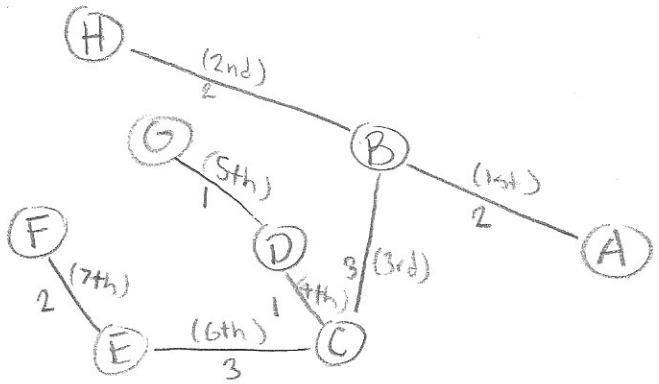
Jake Traut
103134737

Algorithms HW6

1. Kruska's algorithm



Prim's algorithm
(starting with A)



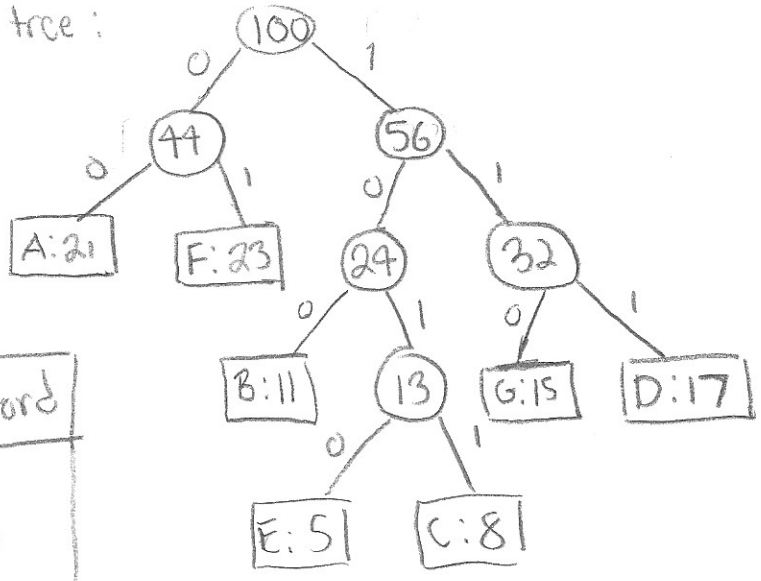
2. Greedy algorithm

For this problem we will design an algorithm that will sort A and B such that $a_1 \leq a_2 \leq \cdots \leq a_n$ and $b_1 \leq b_2 \leq \cdots \leq b_n$, being "greedy" in always selecting the next immediate smallest integer for sorting. This should ensure that the smallest value in A occurs with the smallest value in B and so on until the largest two values (a_n and b_n) occur together. This will maximize $\Pi_{i=1}^{n} a_i^{b_i}$ by allowing the larger values of A to be matched with the larger values of B for $a_i^{b_i}$. A larger integer with a larger exponent will produce the greatest values.
This is optimal with a runtime of sorting A and B of $O(n \log n)$

Honor Code Pledge : X

A: 21% B: 11% C: 8% D: 17% E: 5% F: 23% G: 15%

3. a) Huffman encoding tree:



b) Huffman Encoding:

Symbol	Codeword
A	00
B	100
C	1011
D	111
E	1010
F	01
G	110

c) Length of encoded string in bits = 2,690,000 bits

A ⇒ 1,000,000 x .21 = 210,000
B ⇒ 1,000,000 x .11 = 110,000
C ⇒ 1m x .08 = 80,000
D ⇒ 1m x .17 = 170,000
E ⇒ 1m x .05 = 50,000
F ⇒ 1m x .23 = 230,000
G ⇒ 1m x .15 = 150,000

210,000 + 330,000 + 320,000 + 510,000 + 200,000 + 460,000 + 450,000 = 2,690,000

4. a) Earliest start time: Ordering by increasing start time may not account for a finishing time being later than the next start.

counterexample:

i =	Job 1	Job 2
Times s_i	1	3
Times f_i	10	4

Job 1 will start first even though the next job is due before.

b) Earliest finish time: Guarantees no idle time because by considering increasing finish times will always start with job that is due first then will move on to next soonest deadline right as previous task is finished. Also guarantees no inversions in schedule because the earlier finish time will always precede the later finish time. Thus this ordering is optimal.

c) Shortest interval: This ordering may give precedence to a job thats deadline is later than another job thus produces poor scheduling.

counterexample:

i =	Job 1	Job 2
Times s_i	1	4
Times f_i	3	5

Will start with job 2 but job 1 needs to be done first.