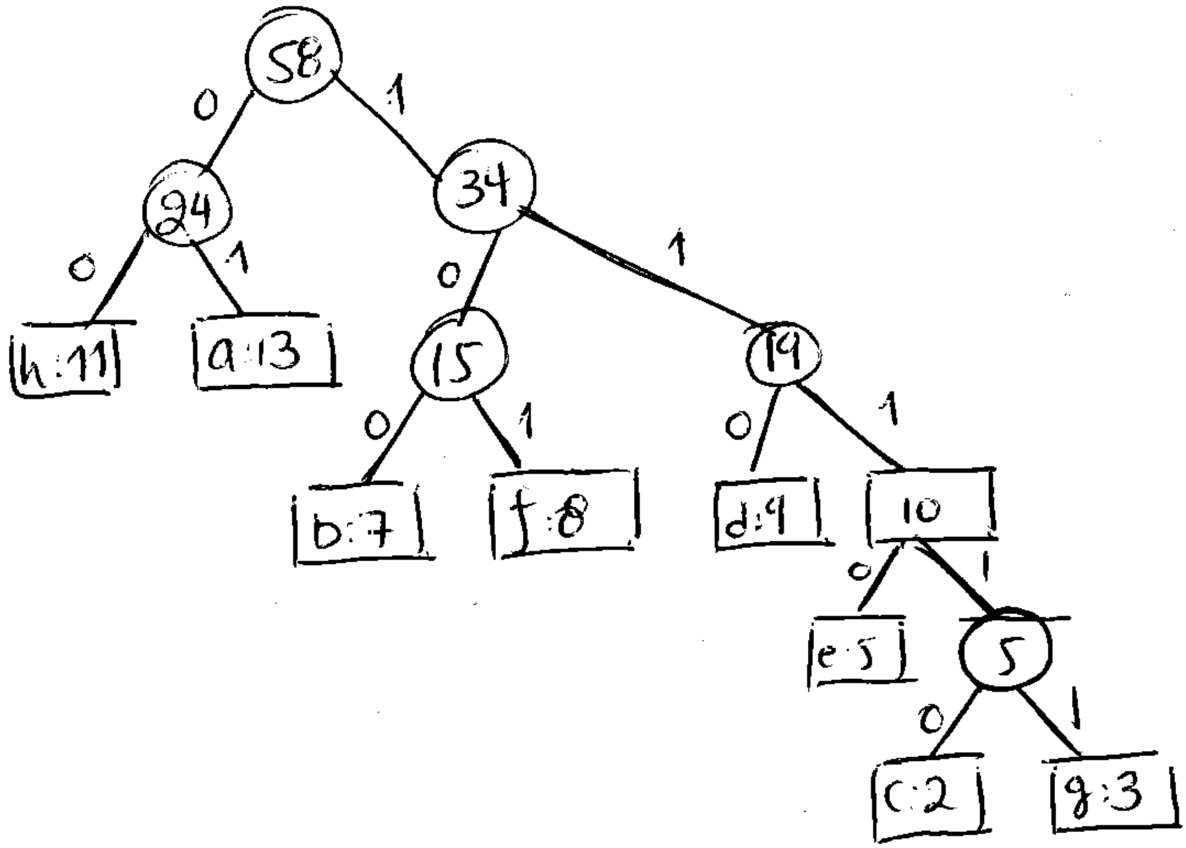
HW6 Solutions 2018

Problem 1



Huffman Code

h	00
a	01
b	100
f	101
d	110
e	1110
c	11110
g	11111

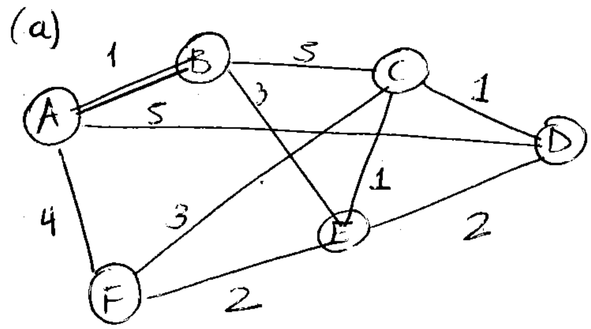
Problem 2

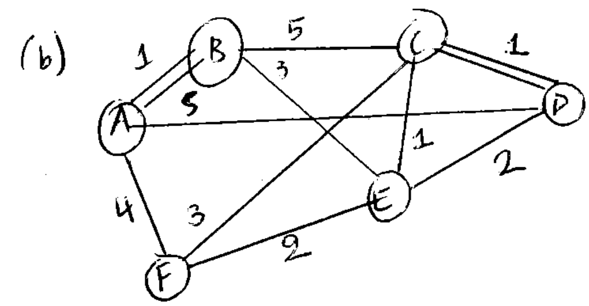
a) Use the highest denomination coin that you can without exceeding the total value. Repeat there is no change left
b) Given an optimal solution $(x_0, x_1, \ldots, x_k)$ where x_i indicates the number of coins of denomination c^i. We will first show that we must have $x_i < c$ for every $i < k$. Suppose that we had some $x_i \geq c$, then, we could decrease x_i by c and increase x_i+1 by 1. This collection of coins has the same value and has $c - 1$ fewer coins, so the original solution must have been non-optimal. This configuration of coins is exactly the same as you would get if you kept greedily picking the largest coin possible. This is because to get a total value of V, you would pick $x_k = \lfloor V c^{-k} \rfloor$ and for $i < k$, $x_i \lfloor (V \bmod c^{i+1}) c^{-i} \rfloor$. This is the only solution that satisfies the property that there aren't more than c of any but the largest denomination because the coin amounts are a base c representation of V mod c^k.
c) Let the coin denominations be {1, 5, 7}, and an initial value of 10. The greedy solution results in the collection of coins {1,1,1,7} but the optimal solution is {5, 5}.
d) Algorithm 2 MAKE-CHANGE(S,v) (Total running time is O(nk))

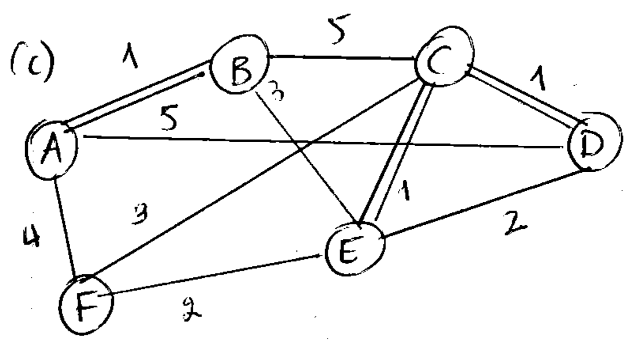
```
Let num_coin and coin be empty arrays of length v
Let change be an empty set
for i from 1 to v do
     best_coin =nil
     best_num = ∞
     for c in S do
          if num_coin[i − c] + 1 < best_num then
               best_num = num_coin[i-c]
               best_coin = c

          end

     end

     num_coin[i] = best_num
     coin[i] = best_coin

end
j = v
while j > 0 do
            add coin[j] to change
            j = j − coin[j]

end
return change
```

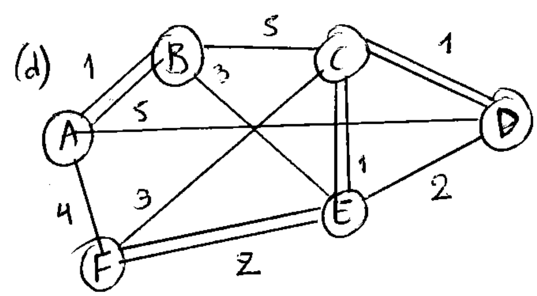
Problem 3

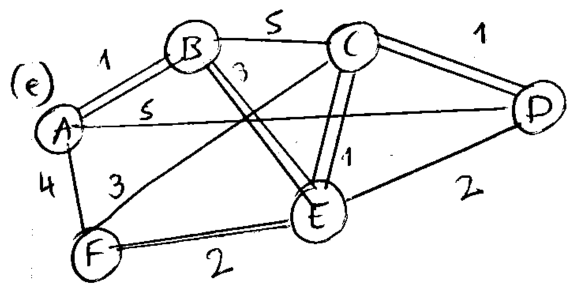
KRUSKAL'S ALGORITHM











Before step (e), we considered edges D-E, F-C, but we discarded them as they would form a loop.

After step (e), we discard all of the other edges as they would form a loop.

PRIM'S ALGORITHM (A is the root node)