CS325 HW4

Yihong Liu

Problem1.

a) a = 0, b = 10, c = 110, d = 1110, e = 1111

b) length = 10^6/2 * 1 + 10^6/4 * 2 + 10^6/8 * 3 + 10^6/16 * 4 + 10^6/16 * 4 = = 1875000

Problem2.

n is the number of items; W is the total weight can be carried

```
Knapsack (n, W)
    K[][] = (n+1) (W+1)
    for i = 0 to n
        K[i][0] =0
    for j = 1 to W
        K[0][j] = 0
    for i = 1 to n
        for j = 1 to W
            if j < i.weight
                K[i][j] = K[i-1][j]
            else
                K[i][j] = max(K[i-1,j],K[i-1][j-i]- i.weight+i.value)
```

Problem3.

a) To prove the greedy algorithm of picking the largest denomination is always the optimal solution is equal to prove that the number of coins of any denomination c^i except c^k used is less than c.

Assume (x_0 to x_k) is the optimal solution, where x_i represents the number of coins denomination c^i. assume for some j, $x_j >= c$, then we can replace c number of c_j denomination by one c_{j+1} denomination coin. Thus, decrease x_j by c and increase x_{j+1} by 1. Now the number of coins used decreased by c-1. Which is contradiction that (x_0 to x_k) is the optimal solution.

Therefore, an optimal solution must have xi < c for any denomination c^i

b)

```
int coinChange(int amount, int[] coins) { // Check if there is no more change to make.

    if (amount == 0) {

    return 0; }

    //sort first then reverse it, become greatest to smallest set of coins[]

    Collection.sort(coins[])

    Collections.reverse(coins[])

    // Loop over the change in order of greatest to smallest.

    for (int i = coins.length; i > 0; i--)

    { int coin = coins[i - 1];
```

```
// If the next largest coin is found, print out its value.
 if (amount >= coin)
 { return 1 + coinChange(amount - coin, coins); }
 }
```

Problem4.

See teach files

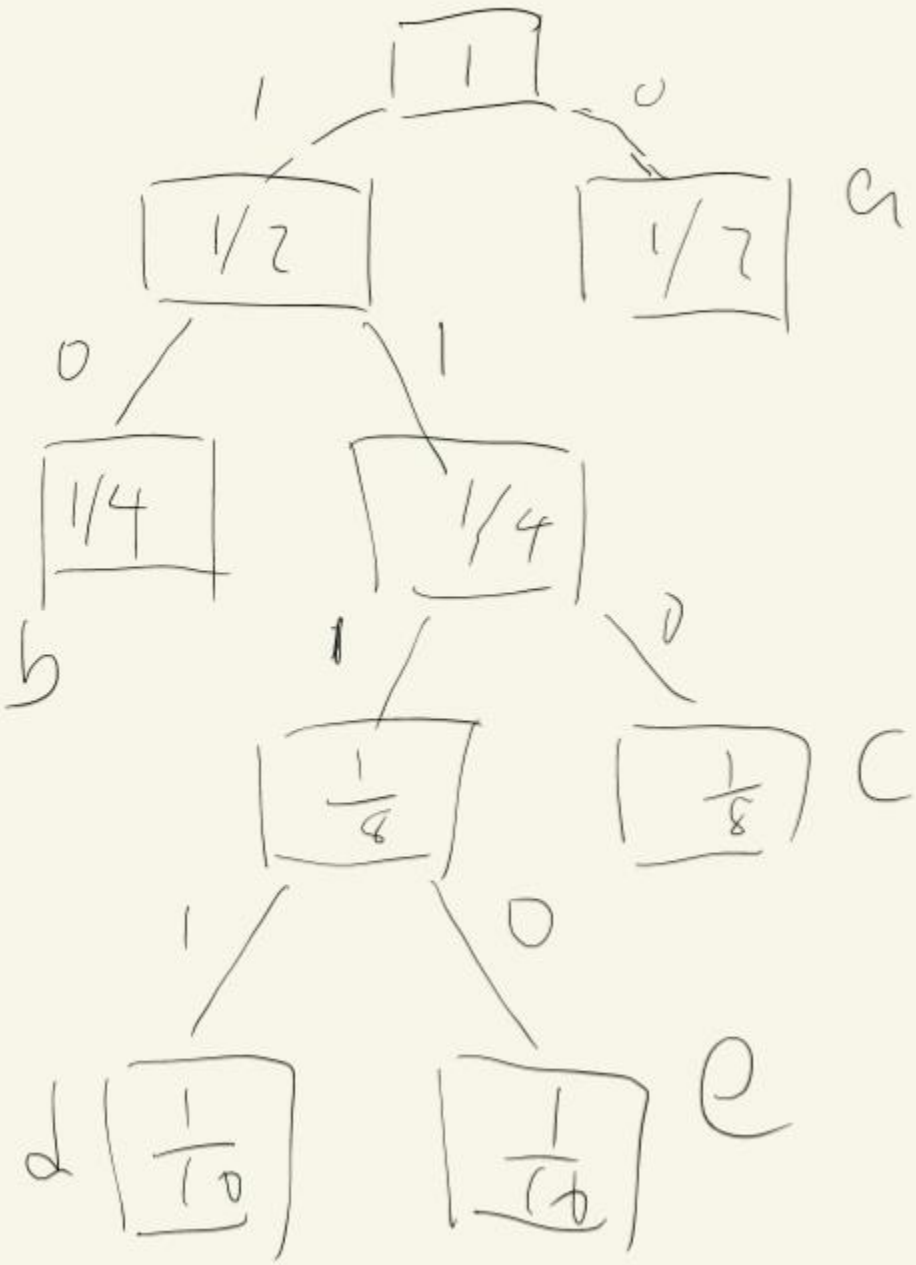
Problem5.

Consider the set of frequencies for the symbol of length “n” . The longest codeword can be of length n-1. When encoding “n” with n-2 of them having probabilities 1/2, 1/4, …, 1/2n-2 and two of them having probability 1/2n-1 achieves this value.

For example (a prefix tree below)

As we can see, there are n = 5 nodes, which are a, b, c, d, e. with frequencies of ½ ,1/4, 1/8, 1/16.

1110 stands for e, which the length is 4, equal to n-1 = 4. Same for d.


1
1
0
1/2
1/2
a
0
1
1/4
1/4
b
1
0
1/8
1/8
c
1
0
1/16
1/16
d
e