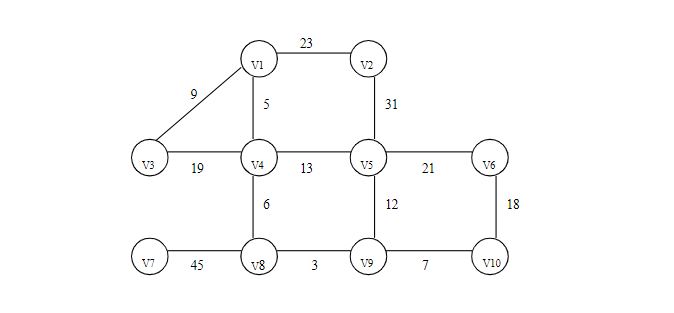
CS.3310 Homework 3

1. Construct Minimum Spanning Tree (MST) using Prim’s algorithm and find the cost of that MST



\*\*\*Notation – v12 = 23 means edge weight from v1 to v2 is 23

Step 1 – Start from v1, add it to the SetMST and compute adjacent edges weight.

V12 = 23; v13 = 9; v14 = 5. Connect v1 and v4

Step 2 – Travel to v4, add it to SetMST, and compute adjacent edges weight.

V12 = 23; v13 = 9; v45 = 13; v48 = 6. Connect v4 and v8

Step 3 – Travel to v8, add it to SetMST and compute adjacent edges weight.

V12 = 23; v13 = 9; v45 = 13; v87 = 45; v89 = 3. Connect v8 and v9

Step 4 – Travel to v9, add it to SetMST and compute adjacent edges weight.

V12 = 23; v13 = 9; v45 = 13; v87 = 45; v95 = 12, v910 = 7. Connect v9 and v10

Step 5 – Travel to v10, add it to SetMST and compute adjacent edges weight.

V12 = 23; v13 = 9; v45 = 13; v87 = 45; v95 = 12, v106 = 18. Connect v1 and v3

Step 6 – Travel to v3, add it to SetMST and compute relevant adjacent edges weight.

V12 = 23; v45 = 13; v87 = 45; v95 = 12, v106 = 18 Connect v9 and v5

Step 7 – Travel to v5, add it to SetMST and compute relevant adjacent edges weight.

V12 = 23; v87 = 45; v106 = 18; v56 = 21. Connect v10 to v6

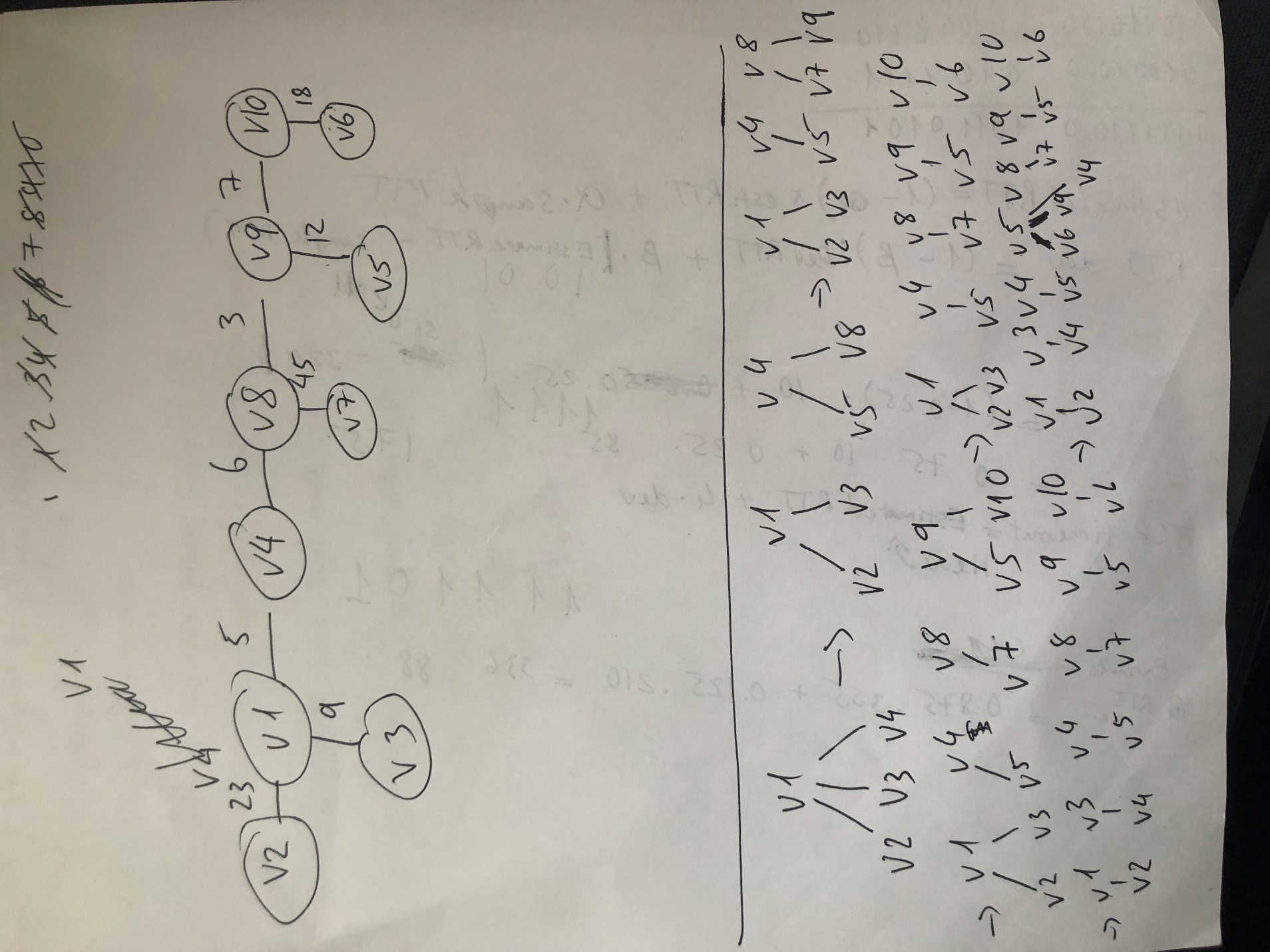
Step 8 – Add v6 to SetMST and check remaining edges weight.

V12 = 23; v87 = 45. Connect v1 to v2

Step 9 – Add v2 to SetMST and check remaining edges weight.

V87 = 45. Connect v8 to v7. **Completed MST as shown below with 9 edges**

**Total cost/weight** = 23+5+9+6+45+3+12+7+18 = **128**



1. Construct MST using Kruskal’s algorithm and cost of that MST

Sort the edges

(v8,v9) = 3

(v1,v4) = 5

(v4,v8) = 6

(v9, v10) = 7

(v1, v3) = 9

(v5,v9) = 12

(v4, v5) = 13 X

(v6, v10) = 18

(v3, v4) = 19 X

(v5, v6) = 21 X

(v1, v2) = 23

(v2, v5) = 31 X

(v7, v8) = 45

Step 1 – connect v8 and v9. New set

Step 2 – connect v1 and v4. New set

Step 3 – connect v4 and v8. Since they came from different sets, there’s no cycle created.

Step 4 – connect v9 and v10. Since they came from different sets, there’s no cycle created.

Step 5 – connect v1 and v3. Since they came from different sets, there’s no cycle created.

Step 6 – connect v5 and v9. Since they came from different sets, there’s no cycle created.

Step 7 – discard (v4, v5) because a cycle would be created

Step 8 – connect v6 and v10. Since they came from different sets, there’s no cycle created.

Step 9 – discard (v3, v4), (v5, v6) because a cycle would be created.

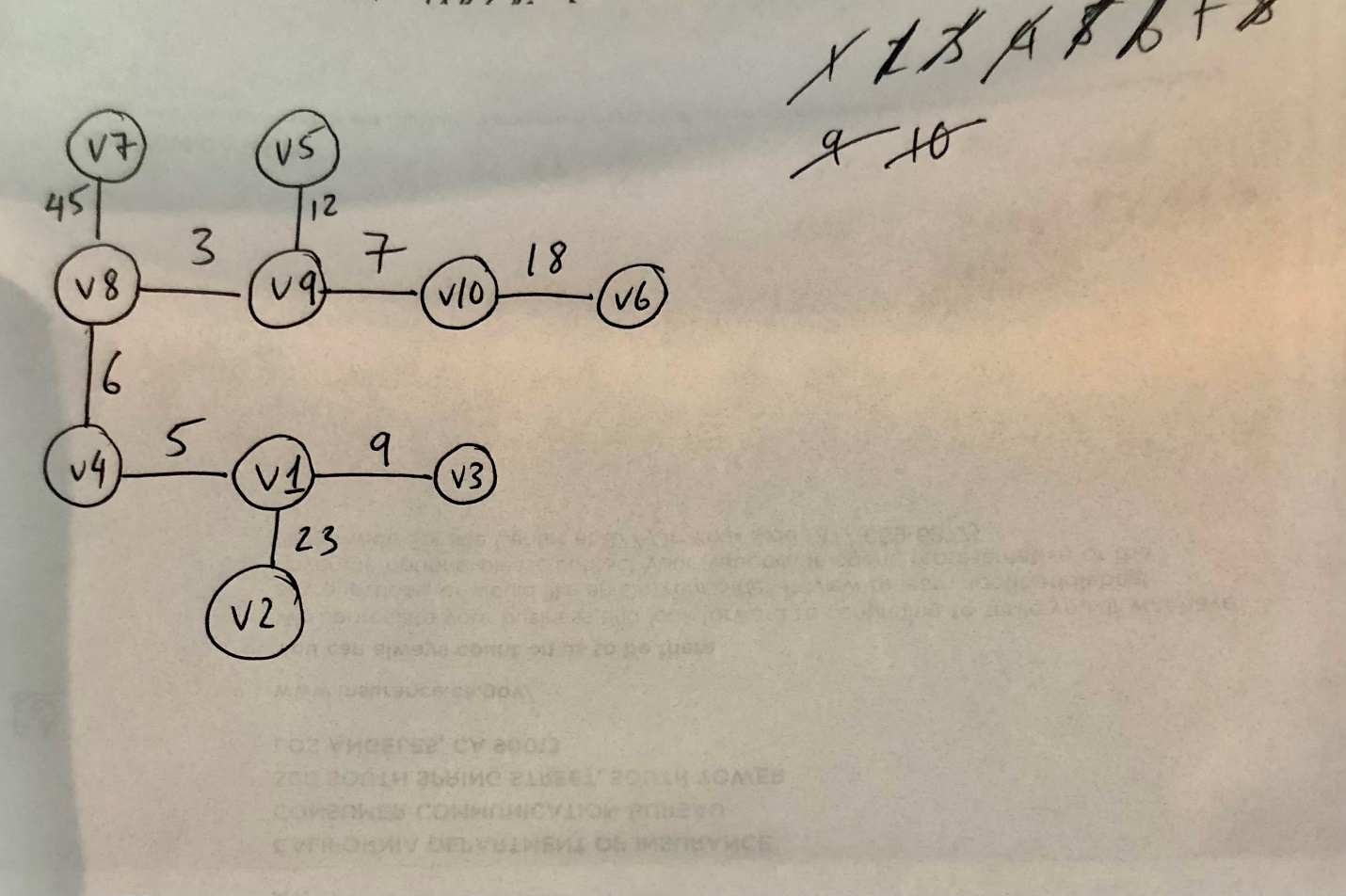
Step 10 – connect v1 and v2. Since they came from different sets, there’s no cycle created.

Step 11 – discard (v2, v5) because a cycle would be created.

Step 12 – connect v7 and v8. Since they came from different sets, there’s no cycle created.

MST completed and as shown below with 9 edges

**Total cost =** 3+5+6+7+9+12+18+23+45 = **128**



1. Imagine that the objects and their weights are as below with knapsack M = 10

|  |  |  |  |
| --- | --- | --- | --- |
| Object | I1 | I2 | I3 |
| Weight | 5 | 6 | 7 |
| Profit | 11 | 18 | 20 |
| Profit/weight | 2.2 | 3 | 2.857 |

The order based on nonincreasing profit/weight is I2, I3, I1.

Step 1 – We add I2 to the knapsack. Then, the knapsack only have capacity of 4 left.

Step 2 – We can’t add anymore items to the bag because their weights is greater than 3. This means that the knapsack utilization rate isn’t 100% if it’s a 0/1 Knapsack. In this case, **the profit from this 0/1 knapsack is 18.**

**Profit = 0\*11 + 1\*18 + 0\*20 = 18**

However, the knapsack problem in class allow for partial/fractional inclusion of items so the knapsack utilization rate is always 100%.

The optimal solution is ordering by profit/weight of I2, I3, I1 just as above.

Step 1 – We add I2 to the knapsack. Then, the knapsack only have capacity of 4 left.

Step 2 – Since 4 is less than the weight of any object, we must add a fraction of an object. The next item is I3 so we add 4/7 of I3 to the knapsack. Then, the profit would be:

**Profit = 0\*11 + 1\*18 + 4/7\*20 = 29.429**

By profit comparison, 29.429 > 18 so the proposed 0/1 knapsack with the proposed strategy is not the optimal solution. This will be the usually be the case because 0/1 knapsack doesn’t always have 100% utilization rate whereas the knapsack strategy discussed in class will always have 100% utilization rate which always maximizes profit.

1. Let An = { a1, a2, ..., an } be a finite set of distinct coin types (e.g., a1= 50 cents, a2= 25 cents, a3= 10 cents etc.). We assume each ai is an integer and that a1 > a2 > ... > an. Each type is available in unlimited quantity. The coin changing problem is to make up an exact amount C using a minimum total number of coins. C is an integer > 0
   1. If an ≠ 1, then for C value ending in 1’s such as {1, 11, 21, 31,…}, there is no solution because an ≠ 1
   2. The algorithm is as below

#pass the amount to get changed for C, and A is the finite set of distinct coin type

Def numCoin (C, A):

Count = 0

numOfCoin = 0

n = 1

while (C > 0):

#find greatest amount of coin possible for this type of denomination

count = C // an

#find remaining value after

C -= (count\*an)

#update values

numOfCoin += count

n += 1

return numOfCoin

* 1. A counter example would be to consider the amount of 55 and An = {10,9.8.7,6,1}

The algorithm in part (b) would yield the number of coins is 10 coins used consisting of five 10 coins and five 1 coins. However, there’s another solution consisting of four 10 coins, one 9 coin, and one 6 coin, totaling six coins used instead of 10. The algorithm in part (b) uses more coins than the counterexample so part (b) algorithm isn’t an optimal solution.

* 1. If An = {Kn-1, Kn-2............K0) where n > 1, we can substitute in some value and check. Suppose n = 5 and k = 2, we have An = {16, 8, 4, 2, 1}. Then, suppose we have the amount C = 55, the algorithm would yield three 16 coins, one 4 coin, one 2 coin, and one 1 coin, totaling 6 coins.

We know this is an optimal solution because the complexity when reducing is like a binary search tree where the amount of coins or steps require is approximately O(log n). The algorithm will always yield minimum number of coins.

Lg(55) = 5.78 rounded up to 6