Data Structures Bonus Assignment

Kolby Sarson – 104232327

**1.** In a fundraising campaign, we wish to raise an x amount of money we have the list of potential donors and their budgets. Design an algorithm, to find the amount that each person donates to achieve the fundraising target. You must find this optimal distribution that minimize the highest contribution.

X = money to raise;

While(x>0){

If(All donor budgets are above the average)

All pay the average;

Else

Donor with the lowest budget pays his max budget;

X = X – lowest donor budget;

Recalculate average between remaining donors;

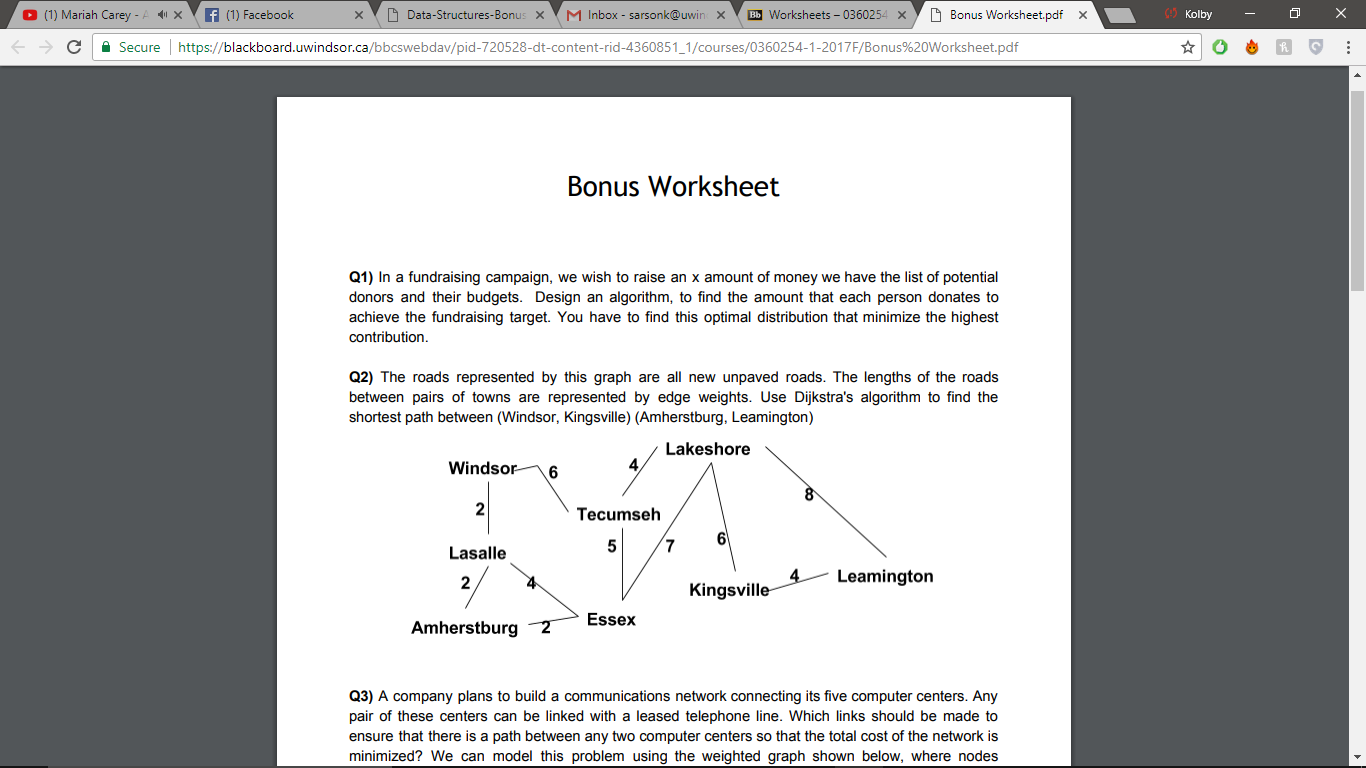
}

**2.** The roads represented by this graph are all new unpaved roads. The lengths of the roads between pairs of towns are represented by edge weights. Use Dijkstra's algorithm to find the shortest path between (Windsor, Kingsville) and (Amherstburg, Leamington).

10

X

∞



18

16

6

X

X

X

X

X

X

2

4

6

0

∞

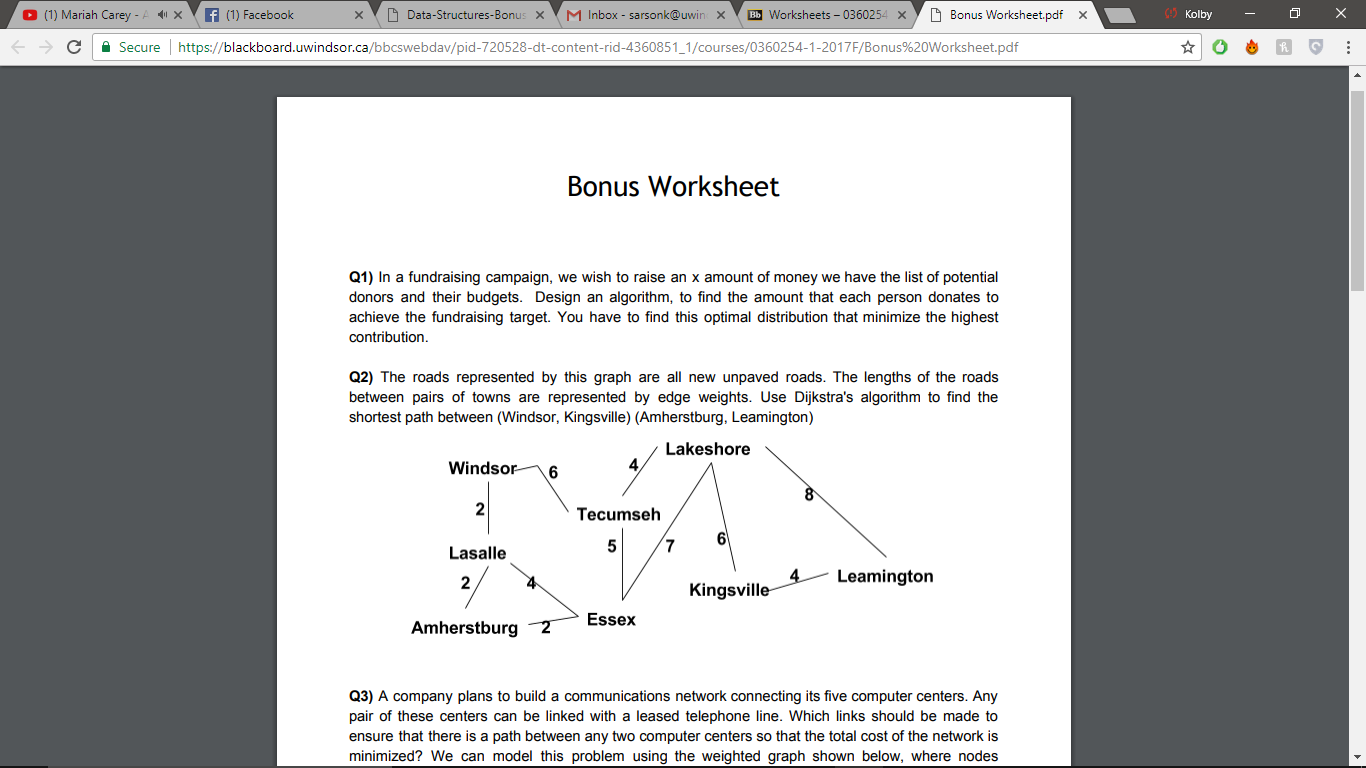
∞

∞

∞

∞

∞



X

X

X

X

X

X

X

4

9

∞

∞

17

15

2

2

7

0

∞

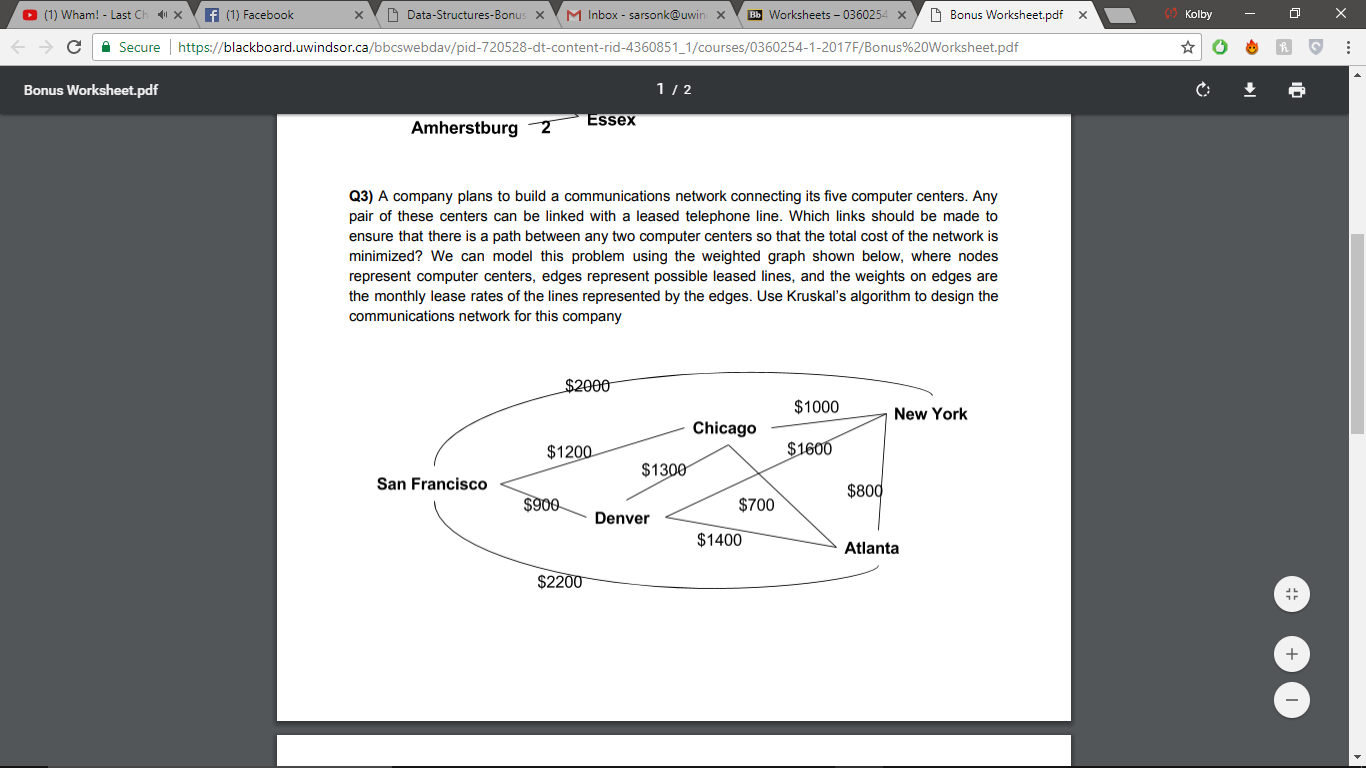
∞

∞

∞

∞

3. A company plans to build a communications network connecting its five computer centers. Any pair of these centers can be linked with a leased telephone line. Which links should be made to ensure that there is a path between any two computer centers so that the total cost of the network is minimized? We can model this problem using the weighted graph shown below, where nodes represent computer centers, edges represent possible leased lines, and the weights on edges are the monthly lease rates of the lines represented by the edges. Use Kruskal’s algorithm to design the communications network for this company.



ABC

**4.** Given a value N, if we want to make change for N cents, and we have infinite supply of each of S = {S1, S2,..., Sm} valued coins, how many ways can we make the change? The order of coins doesn’t matter. For example, for N = 4 and S = {1,2,3}, there are four solutions: {1,1,1,1}, {1,1,2}, {2,2}, {1,3}. So, output should be 4. For N = 10 and S = {2, 5, 3, 6}, there are five solutions: {2,2,2,2,2}, {2,2,3,3}, {2,2,6}, {2,3,5} and {5,5}. So, the output should be 5. Discuss why using a dynamic programming is appropriate to solve the above problem.

This problem can be solved recursively, but dynamic programming is needed since not all tests will pass due to overlapping subproblems. Using a temporary array to store values dynamically in a bottom up fashion could solve this issue. For example, for N = 4 and S = {1,2,3}, if we add a coin of value 1 to all the combinations for N = 3 and S = {1,2,3}, these solutions would work. If we store these values dynamically, they can be reused. Time would be saved by eliminating the need to calculate twice using dynamic programming.

**5.** Prim's algorithm is an efficient algorithm that finds a minimum spanning tree for a weighted undirected graph. Prim's algorithm is a greedy algorithm explain why we consider Prim's a greedy algorithm.

Prim’s algorithm is greedy because it finds the minimum spanning tree for a weighted, undirected graph using a greedy approach. It takes a greedy approach because given any starting vertex, it selects the smallest weighted edge, and moves to the connected, unvisited vertex. This process is repeated until every vertex has been visited, thus estimating the minimum spanning tree is a reasonable number of steps.

**6.** Which of the following sorting algorithms are stable: insertion sort, merge sort, bubble sort and quicksort? Give a simple scheme that makes any sorting algorithm stable.

Insertion sort, merge sort and bubble sort are all stable algorithms while quicksort is an unstable algorithm. To change an algorithm from unstable to stable, the elements can be numbered before sorting and then sorting those elements with the same value data by the number keys after sorting by data value.

**7.** Illustrate the operations of quicksort on the array A = [5, 7, 4, 3, 2, 9, 8, 6].

*6 is pivot*

|​ 5, 7, 4, 3, 2, 9, 8, **6**

*Partition array*

5, 4, 3, 2 |​ 7, 9, 8, 6

*Swap the value at the partitions index with 6*

5, 4, 3, 2, **6​,** 9, 8 ,7

*2 and 7 as the pivots of the two unsorted partitions and repeat*

| 5, 4, 3, **2​, 6​,** | 9, 8, **7**

*Partition the unsorted parts again and swap values*

2, 4, 3, 5, 6​, 7, ​8, 9

*On left choose a pivot, on the right base case n = 2*

2​,| 4, 3, 5​, **6​, 7​,** 8​, 9

*Partition*

**2**​, 4, 3,| **5​, 6​, 7​, 8​, 9**

*Swap*

**2​,** 4, 3, **5​, 6​, 7​, 8​, 9**

*Base case n = 2, swaps*

**2​, 3​, 4​, 5​, 6​, 7​, 8​, 9**

*Array is sorted*

8. Given a hash table with m=11 entries and the hash function h1(key) = key mod m Insert the keys {22, 1, 13, 11, 24, 33, 18, 42, 31} in the given order (from left to right) to the hash table using chaining and linear probing.

ABC

**9.** Show the AVL tree that results after each of the integer keys 9, 27, 50, 15, 2, 21, and 36 are inserted, in that order, into an initially empty AVL tree. Clearly show the tree that results after each insertion, and make clear any rotations that must be performed.

**LR on 9**

**3)**

**2)**

**4)**

**1)**

**6)**

**5)**

**LR on 9**

**RR on 27**

**8)**

**7)**

**10.** Write an algorithm to find the maximum depth of a binary search tree. What is the run time complexity of your algorithm?

The algorithm is O(logn) complexity

maxDepth();

If(Tree is empty)

return 0;

Else{

1. Get the maxDepth of the left subtree recursively

maxDepth(tree->leftsubtree);

1. Get the maxDepth of the right subtree recursively

maxDepth(tree->rightsubtree);

1. Get the max of the maxDepths of the left and right subtrees and add 1 to it for the current node

maxMaxDepth = max(max depth of left subtree, max depth of right subtree)+1;

1. return maxDepth;

}