Homework 3

Kevin Martin
CIS675 - Syracuse University

February 11, 2020

1. Question 1

   (a) Table showing the intermediate distance values of all nodes at each iteration:

Iteration	Node	Distance
0	A	0
	B	∞
	C	∞
	D	∞
	E	∞
	F	∞
	G	∞
	H	∞
	I	∞

Iteration	Node	Distance
1	**A**	0
	B	15
	C	21
	D	∞
	E	∞
	F	∞
	G	∞
	H	∞
	I	∞

Iteration	Node	Distance
2	*A*	0
	B	15
	C	21
	D	∞
	E	25
	F	∞
	G	∞
	H	19
	I	37

Iteration	Node	Distance
3	*A*	0
	B	15
	C	21
	D	∞
	E	25
	F	∞
	G	∞
	H	19
	I	36

Iteration	Node	Distance
4	*A*	0
	B	15
	C	21
	D	30
	E	25
	F	∞
	G	∞
	H	19
	I	36

Iteration	Node	Distance
5	*A*	0
	B	15
	C	21
	D	30
	E	25
	F	34
	G	30
	H	19
	I	36

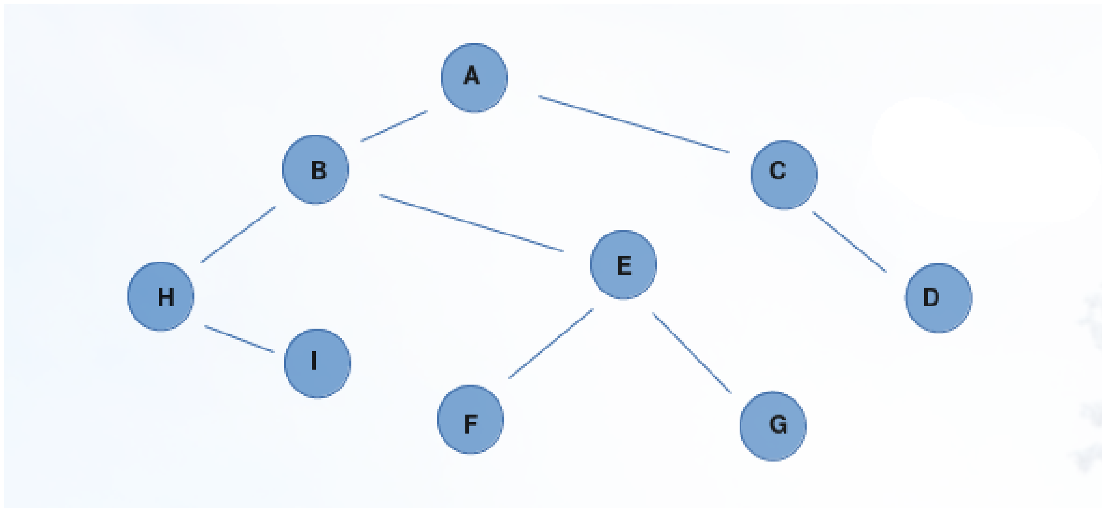
Iteration	Node	Distance
6	*A*	0
	B	15
	C	21
	D	30
	E	25
	F	34
	G	30
	H	19
	I	36

Iteration	Node	Distance
7	*A*	0
	B	15
	C	21
	D	30
	E	25
	F	34
	G	30
	H	19
	I	36

Iteration	Node	Distance
8	*A*	0
	B	15
	C	21
	D	30
	E	25
	F	34
	G	30
	H	19
	I	36

Iteration	Node	Distance
9	*A*	0
	B	15
	C	21
	D	30
	E	25
	F	34
	G	30
	H	19
	I	36

   (b) Final shortest-path tree:



(c) Adjaceny matrix representation with costs:

	A	B	C	D	E	F	G	H	I
A	0	15	21	0	0	0	0	0	0
B	15	0	0	0	10	0	0	4	0
C	21	0	0	9	0	0	0	0	0
D	0	0	9	0	0	0	0	0	0
E	0	10	0	0	0	9	5	0	0
F	0	0	0	0	9	0	0	0	0
G	0	0	0	0	5	0	0	0	0
H	0	4	0	0	0	0	0	0	17
I	0	0	0	0	0	0	0	17	0

(d) Adjaceny list representation with costs:

A:	$\xrightarrow{15}$	B	$\xrightarrow{21}$	C		
B:	$\xrightarrow{15}$	A	$\xrightarrow{10}$	E	$\xrightarrow{4}$	H
C:	$\xrightarrow{21}$	A	$\xrightarrow{9}$	D		
D:	$\xrightarrow{9}$	C				
E:	$\xrightarrow{10}$	B	$\xrightarrow{9}$	F	$\xrightarrow{5}$	G
F:	$\xrightarrow{9}$	E				
G:	$\xrightarrow{5}$	E				
H:	$\xrightarrow{4}$	B	$\xrightarrow{17}$	I		
I:	$\xrightarrow{17}$	H				

2. Question 2

   (a) The intermediate values of the delay array in each iteration, using Prim's algorithm:

Primm's Algorithm

Iteration	Node	Distance
0		1 0
		2 ∞
		3 ∞
		4 ∞
		5 ∞
		6 ∞
		7 ∞
		8 ∞
		9 ∞

Iteration	Node	Distance
1		**1** 0
		2 4
		3 ∞
		4 ∞
		5 ∞
		6 ∞
		7 ∞
		8 ∞
		9 ∞

Iteration	Node	Distance
2		1 0
		2 4
		3 10
		4 ∞
		5 ∞
		6 ∞
		7 ∞
		8 ∞
		9 ∞

Iteration	Node	Distance
3		1 0
		2 4
		3 10
		4 ∞
		5 7
		6 ∞
		7 ∞
		8 ∞
		9 ∞

Iteration	Node	Distance
4		1 0
		2 4
		3 10
		4 ∞
		5 7
		6 2
		7 ∞
		8 ∞
		9 ∞

Iteration	Node	Distance
5		1 0
		2 4
		3 10
		4 ∞
		5 7
		6 2
		7 ∞
		8 ∞
		9 3

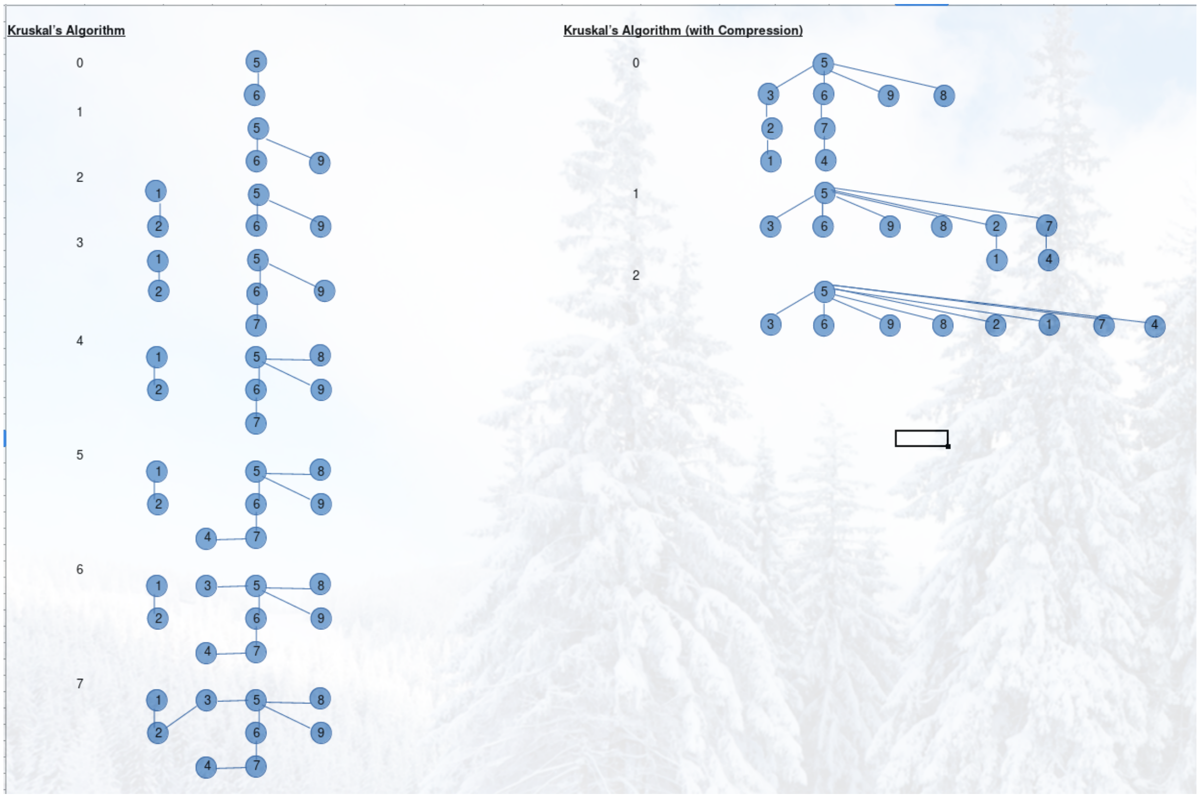
Iteration	Node	Distance
6		1 0
		2 4
		3 10
		4 ∞
		5 7
		6 2
		7 ∞
		8 ∞
		9 3

Iteration	Node	Distance
7		1 0
		2 4
		3 10
		4 ∞
		5 7
		6 2
		7 5
		8 ∞
		9 3

Iteration	Node	Distance
8		1 0
		2 4
		3 10
		4 6
		5 7
		6 2
		7 5
		8 ∞
		9 3

Iteration	Node	Distance
9		1 0
		2 4
		3 10
		4 6
		5 7
		6 2
		7 5
		8 6
		9 3

   (b) The disjoint-sets data structure at every intermediate stage, both regular intervals as well as assuming compression, using Kruskal's algorithm:



3. Question 3

   (a) Pseudocode for a greedy algorithm to compute optimal order and minimize wait time:

   Sort the service time in ascending order, and serve them in order of increasing scheduling times.
   Let n be the list of customers that need to be solved, with each time required t_i as the relevent index in the array:

   def greedySort(n [])
   sort(n); // in ascending order
   currentJob = n[1]
   for i = 1 to n:
           currentJob = currentJob + t_i

   (b) Claim: the running time of the proposed algorithm is $O(nlogn)$

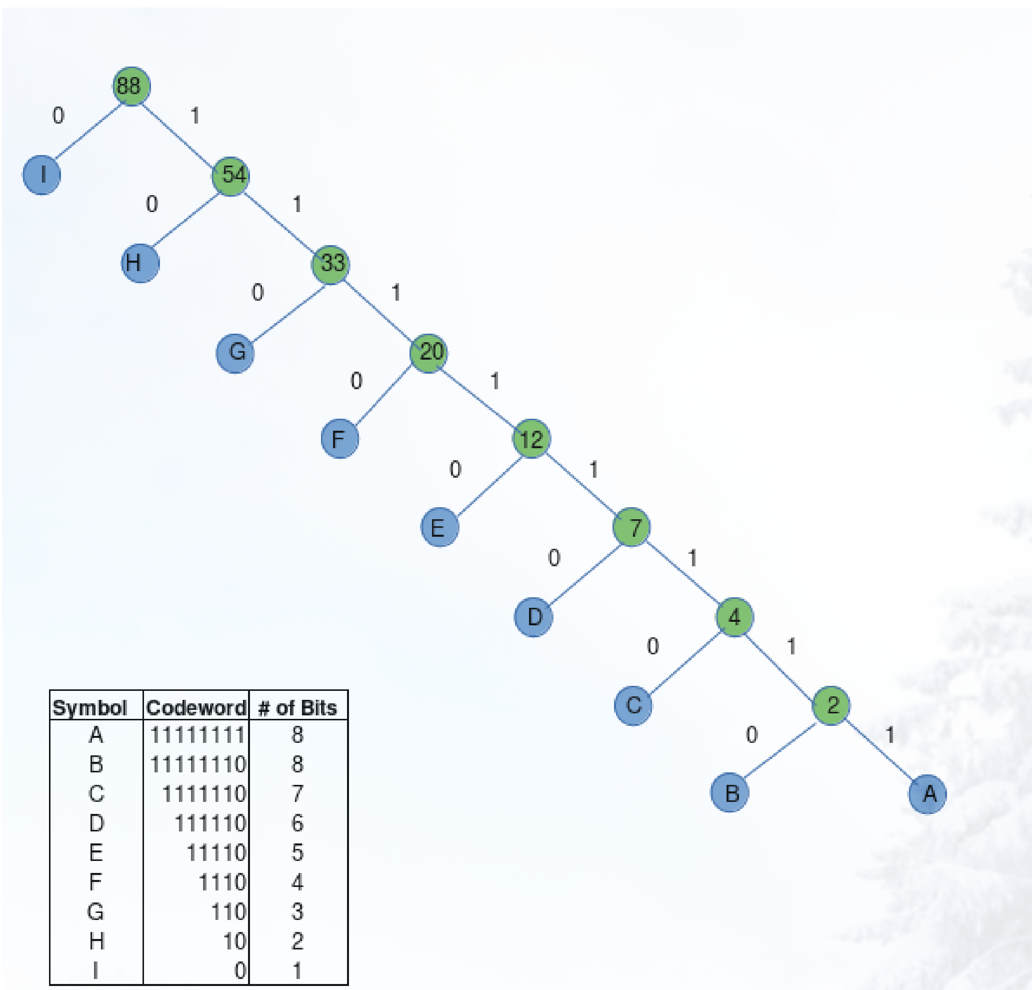
   Proof. Sorting the list to get the algorithm started is where most of the time is required. Sorting a list of n elements takes $O(nlogn)$

time. The rest of the algorithm can be completed in constant time, $O(1)$. Because the algorithm can be written as $c_1 + (c_1 + c_2) + (c_1 + c_2 + ...c_n)\frac{1}{n}$, we can see that c_1 repeats itself the most times. As such, because it is the shortest time, then no other ordering could be correct. Therefore the greedy strategy holds true.

■

4. Question 4

   (a) Optimum Huffman encoding (note: very unfavorable/inefficent outcome given the Fibonacci sequencing of the alphabet):



Symbol	Codeword	# of Bits
A	11111111	8
B	11111110	8
C	1111110	7
D	111110	6
E	11110	5
F	1110	4
G	110	3
H	10	2
I	0	1

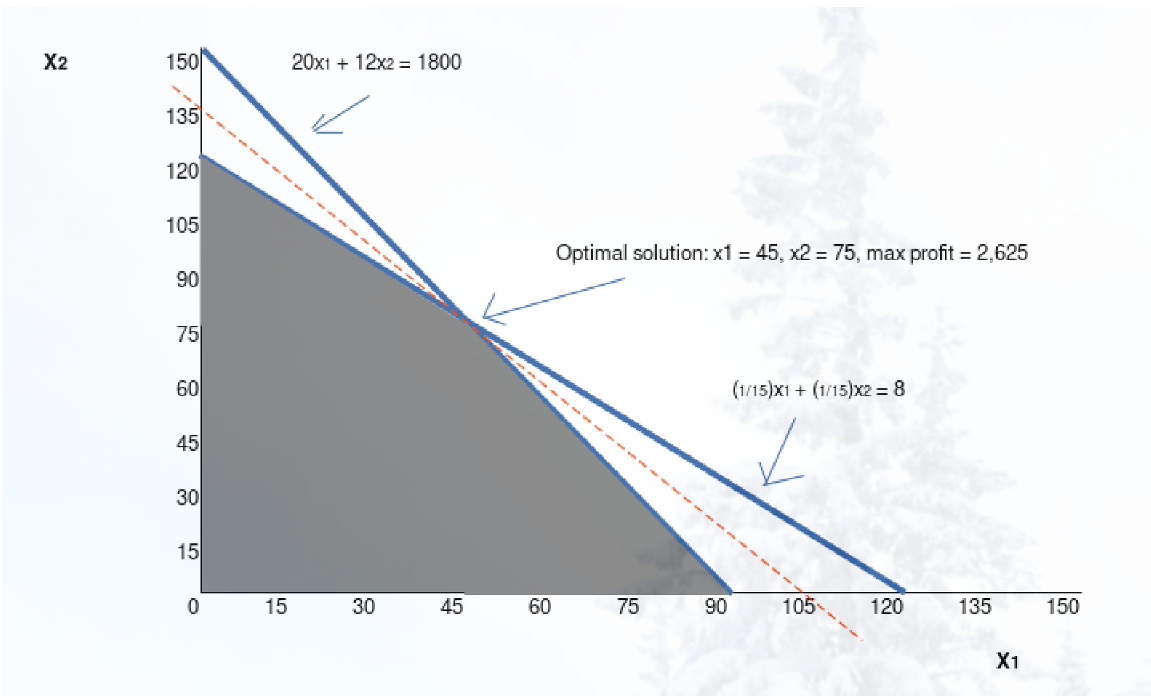
(b) The expected bits per letter are as follows:
A: 8
B: 8
C: 7
D: 6
E: 5
F: 4
G: 3
H: 2
I: 1

5. Question 5

(a) To represent the situation as a linear problem, we formulate as follows, letting x_1 be coffee mugs and x_2 be milk glasses:

Objective function	$\max 25x_1 + 20x_2$
Constraints	$20x_1 + 12x_2 \leq 1800$
	$x_1/15 + x_2/15 \leq 8$
	$x_1, x_2 \geq 0$

(b) Graph of feasible region:



(c) The coordinates of all vertices of the feasible region are:
(0, 0), (90, 0), (45, 75), (0, 120)

(d) The optimal product mix to maximize daily proift is:
45 coffee mugs at \$ 25 each and 90 milk glasses at \$ 20 each, which gives a total profit of \$2,625 per day. This is represented on the graph by the furthest out point on on the feasible region, where the tangential dotted line intercepts the point (45, 75).