Title: Hashing and Graphs
Author: EFE ACER
ID: 21602217
Section: 3
Assignment: 4
Description: This pdf contains answers to Questions 1 and 3

Question 1:

(a)

i) Linear Probing

insertions:

- *12 mod 13 = 12*
- *15 mod 13 = 2*
- *20 mod 13 = 7*
- *30 mod 13 = 4*
- *41 mod 13 = 2 -> 2 + 1 = 3*
- *29 mod 13 = 3 -> 3 + 1, 3 + 2 = 5*
- *17 mod 13 = 4 -> 4 + 1, 4 + 2 = 6*
- *25 mod 13 = 12 -> (12 + 1) mod 13 = 0*
- *22 mod 13 = 9*

final hash table:

indexes:	
0	**25**
1	
2	**15**
3	**41**
4	**30**
5	**29**
6	**17**
7	**20**
8	
9	**22**
10	
11	
12	**12**

ii) Quadratic Probing

insertions:

- *12 mod 13 = 12*
- *15 mod 13 = 2*
- *20 mod 13 = 7*
- *30 mod 13 = 4*
- *41 mod 13 = 2 ->* $2 + 1^2 = 3$
- *29 mod 13 = 3 ->* $3 + 1^2$, $3 + 2^2$, $3 + 3^2$,
  $(3 + 4^2) \bmod 13 = 6$
- *17 mod 13 = 4 ->* $4 + 1^2 = 5$
- *25 mod 13 = 12 ->* $(12 + 1^2) \bmod 13 = 0$
- *22 mod 13 = 9*

final hash table:

indexes:	
0	**25**
1	
2	**15**
3	**41**
4	**30**
5	**17**
6	**29**
7	**20**
8	
9	**22**
10	
11	
12	**12**

iii) Separate Chaining

insertions:

- *12 mod 13 = 12*
- *15 mod 13 = 2*
- *20 mod 13 = 7*
- *30 mod 13 = 4*
- *41 mod 13 = 2*
- *29 mod 13 = 3*
- *17 mod 13 = 4*
- *25 mod 13 = 12*
- *22 mod 13 = 9*

final hash table:

indexes:	
0	**-> NULL**
1	**-> NULL**
2	**-> 15 -> 41**
3	**-> 29**
4	**-> 30 -> 17**
5	**-> NULL**
6	**-> NULL**
7	**-> 20**
8	**-> NULL**
9	**-> 22**
10	**-> NULL**
11	**-> NULL**
12	**-> 12 -> 25**

(b)

Linear Probing Calculated Value of Successful Search:

25: 12, 0 *15: 2* *41: 2, 3* *30: 4* *29: 3, 4, 5* *17: 4, 5, 6*

20: 7 *22: 9* *12: 12* *,are searched values and probed indexes*

Average no. of probes = (2 + 1 + 2 + 1 + 3 + 3 + 1 + 1 + 1) / 9 ≃ 1.667

Linear Probing Theoretical Value of Successful Search:

α (load factor) = 9 / 13,

Theoretical no. of probes = $\frac{1}{2} \cdot \left(1 + \frac{1}{1-\alpha}\right)$ *= 2.125*

Linear Probing Calculated Value of Unsuccessful Search:

0: 0, 1 *1: 1* *2: 2, 3, 4, 5, 6, 7, 8* *3: 3, 4, 5, 6, 7, 8*

4: 4, 5, 6, 7, 8 *5: 5, 6, 7, 8* *6: 6, 7, 8* *7: 7, 8*

8: 8 *9: 9, 10* *10: 10* *11: 11*

38: 12, 0, 1 *,are searched values and probed indexes*

Average no. of probes = (2 + 1 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 2 + 1 + 1 + 3) / 13 ≃ 2.923

Linear Probing Theoretical Value of Unsuccessful Search:

α (load factor) = 9 / 13,

Theoretical no. of probes = $\frac{1}{2} \cdot \left(1 + \frac{1}{(1-\alpha)^2}\right) \cong 5.781$

Quadratic Probing Calculated Value of Successful Search:

25: 12, 0	*15: 2*	*41: 2, 3*	*30: 4*	*29: 3, 4, 7, 12, 6*
17: 4, 5	*20: 7*	*22: 9*	*12: 12*	*,are searched values and probed indexes*

Average no. of probes = (2 + 1 + 2 + 1 + 5 + 2 + 1 + 1 + 1) / 9 $\cong$ 1.778

Quadratic Probing Theoretical Value of Successful Search:

α (load factor) = 9 / 13,

Theoretical no. of probes = $\frac{-\log_e(1-\alpha)}{\alpha} \cong 1.703$

Quadratic Probing Calculated Value of Unsuccessful Search:

0: 0, 1	*1: 1*	*2: 2, 3, 6, 11*	
3: 3, 4, 7, 12, 6, 2, 0, 0, 2, 6, 12, 7, 4		*4: 4, 5, 8*	*5: 5, 6, 9, 2, 8*
6: 6, 7, 10	*7: 7, 8*	*8: 8*	*9: 9, 10*
10: 10	*11: 11*	*38: 12, 0, 3, 8*	*,are searched values and probed indexes*

Average no. of probes = (2 + 1 + 4 + 12 + 3 + 5 + 3 + 2 + 1 + 2 + 1 + 1 + 4) / 13 $\cong$ 3.154

Quadratic Probing Theoretical Value of Unsuccessful Search:

α (load factor) = 9 / 13,

Theoretical no. of probes = $\frac{1}{1-\alpha} = 3.25$

Separate Chaining Calculated Value of Successful Search:

25: 12[0], 12[1] *15: 2[0]* *41: 2[0], 2[1]* *30: 4[0]*

29: 3[0] *17: 4[0], 4[1]* *20: 7* *22: 9*

12: 12[0] *,are searched values and probed indexes*

Average no. of probes = (2 + 1 + 2 + 1 + 1 + 2 + 1 + 1 + 1) / 9 $\cong$ *1.333*

Separate Chaining Theoretical Value of Successful Search:

α *(load factor) = 9 / 13,*

Theoretical no. of probes = $1 + \frac{\alpha}{2} \cong 1.346$

Separate Chaining Calculated Value of Unsuccessful Search:

0: - *1: -* *2: 2[0], 2[1]* *3: 3[0]*

4: 4[0], 4[1] *5: -* *6: -* *7: 7[0]*

8: - *9: 9[0]* *10: -* *11: -*

38: 12[0], 12[1] *,are searched values and probed indexes*

Average no. of probes = (0 + 0 + 2 + 1 + 2 + 0 + 0 + 1 + 0 + 1 + 0 + 0 + 2) / 13 $\cong$ *0.692*

Separate Chaining Theoretical Value of Unuccessful Search:

α *(load factor) = 9 / 13,*

Theoretical no. of probes = $\alpha \cong 0.692$

- Thus, the overall table is:

	Successful Search		**Unsuccessful Search**	
	Calculated	Theoretical	Calculated	Theoretical
Linear Probing	1.667	2.125	2.923	5.781
Quadratic Probing	1.778	1.703	3.154	3.250
Separate Chaining	1.333	1.346	0.692	0.692

Question 2:

- I chose **adjacency list** for the underlying data structure to store the graph. The reason for my selection is that the given flight network graph was relatively **sparse**. The flight network graph, say G(V, E), has |V| = 3425 and |E| = 67652. In complete graph we have C(|V|, 2) = |V| . (|V| - 1) / 2 = 5863600 edges and 67652/5863600 ≅ 0.0115. This tells us that only a small portion of all possible connections are actually used. Thus, an adjacency matrix will occupy way too much unnecessary space. In other words O(|V| + |E|) (space requirement of adjacency list) is much less than $O(|V|^2)$ (space requirement of adjacency matrix). Hence, adjacency list implementation will make graph traversal faster and establish a more efficient program.

- For part a,

  adjacency matrix provides **O(1)** constant behavior to test whether an edge exists, since accessing the corresponding array entries take constant time.

  ; however,

  adjacency list provides O**(|D|)** behavior, since all the edges of the specific vertex is traversed in the worst scenario.

- For part c,

  adjacency matrix provides **$O(|V|^2)$** behavior to perform a DFS that checks the connectedness. Since in the worst search all the cells in the matrix are being visited to traverse the entire graph; the number of operations is proportional to the number of cells, which is $|V|^2$.

  ; however,

  adjacency list provides **O(|V| + |E|)** behavior, since in the worst search we visit every node in the adjacency list. There are |V| nodes required for the heads of each sub-list and the nodes in those sub-lists together add up to |E|.

- For part d,

  adjacency matrix provides **$O(|V|^2)$** behavior to perform a BFS that finds the shortest reach. The explanation is same as the previous one. Considering the worst search, we can say that the algorithm visits every cell in the matrix.

  ; however,

  adjacency list provides O**(|V| + |E|)** behavior, since both in DFS and BFS, for the worst search we visit every node in the adjacency list.

➢ Note: Since the airport names are given as strings in this assignment. We cannot directly transfer them to indexes (This is possible by implementing a hash-map with a low load factor though). Hence, it takes an O(|V|) traversal of the adjacency list or the matrix to find the index corresponding to the particular airport name. So, the results of the parts above should be multiplied by O(|V|) if we take that search into account.