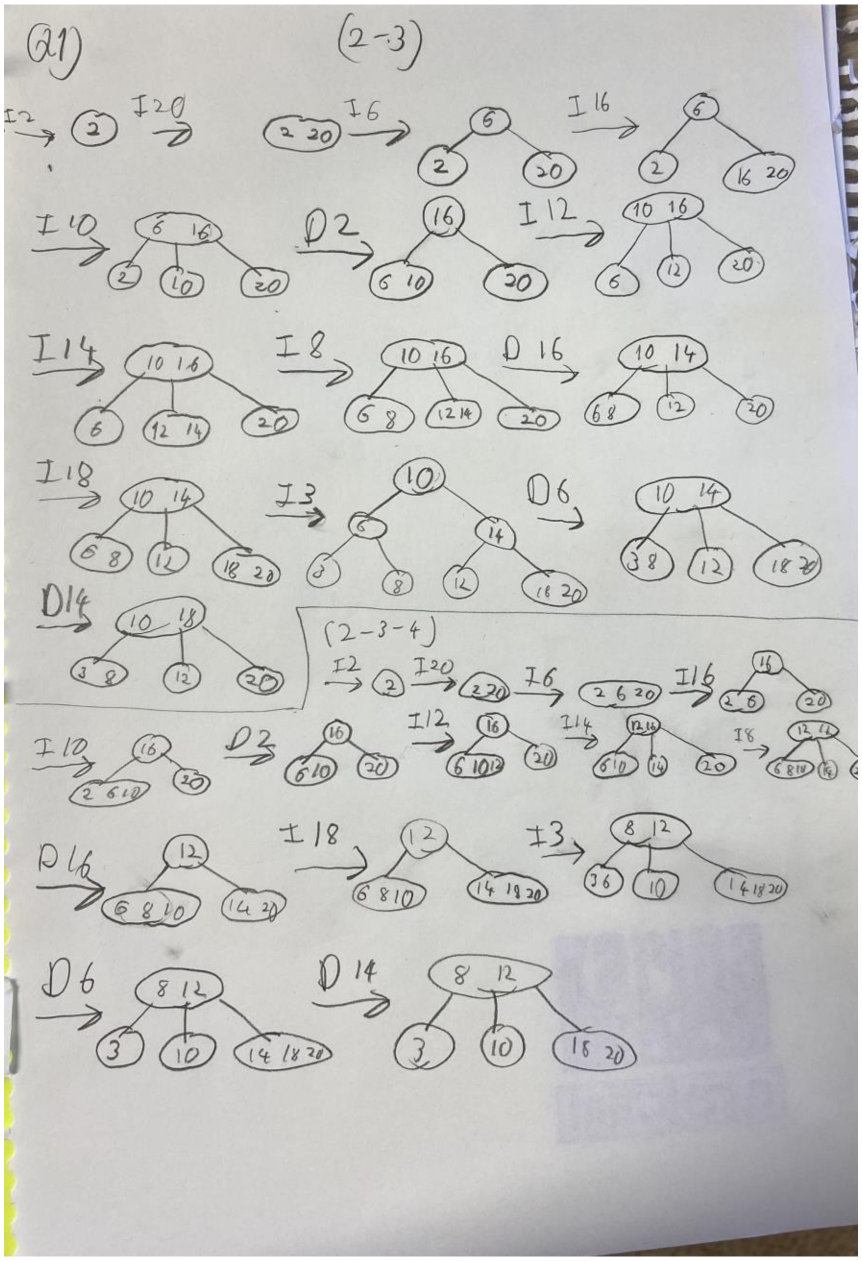
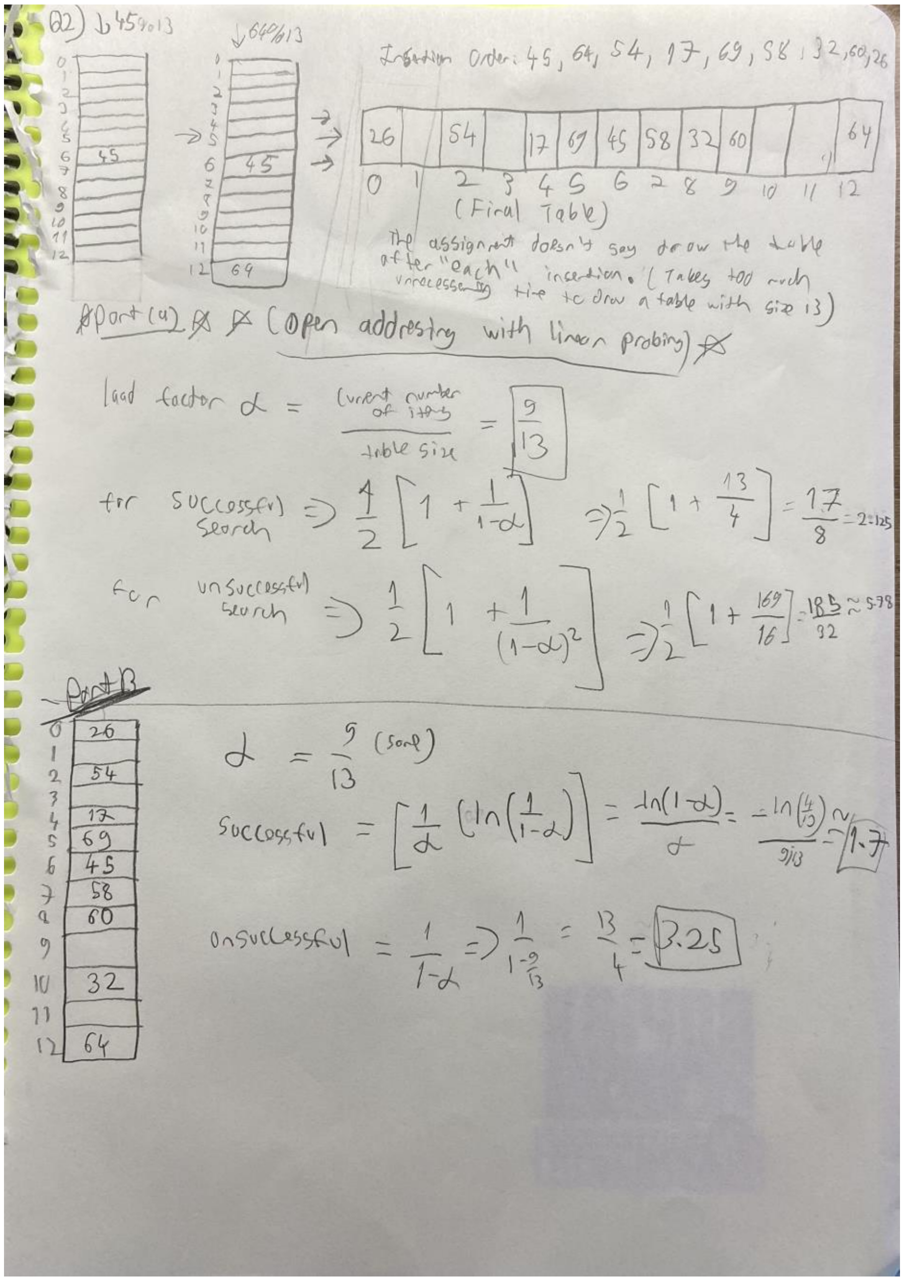
Question1)


Q1)
(2-3)
I2
I20
I6
I16
I10
D2
I12
I14
I8
D16
I18
I3
D6
D14
(2-3-4)
I2
I20
I6
I16
I10
D2
I12
I14
I8
D16
I18
I3
D6
D14

Question2)



Q2) ↓ 45 % 13

0	
1	
2	
3	
4	
5	
6	45
7	
8	
9	
10	
11	
12	

$\rightarrow$

↓ 64 % 13

0	
1	
2	
3	
4	
5	
6	45
7	
8	
9	
10	
11	
12	64

$\rightarrow$ $\rightarrow$ $\rightarrow$

Insertion Order: 45, 64, 54, 17, 69, 58, 32, 60, 26

0	1	2	3	4	5	6	7	8	9	10	11	12
26		54		17	69	45	58	32	60			64

(Final Table)

The assignment doesn't say draw the table after "each" insertion. (Takes too much unnecessary time to draw a table with size 13)

~~Part (a)~~ (Open addressing with linear probing)

load factor $\alpha = \dfrac{\text{current number of items}}{\text{table size}} = \boxed{\dfrac{9}{13}}$

for successful search $\Rightarrow \dfrac{1}{2}\left[1 + \dfrac{1}{1-\alpha}\right] \Rightarrow \dfrac{1}{2}\left[1 + \dfrac{13}{4}\right] = \dfrac{17}{8} = 2.125$

for unsuccessful search $\Rightarrow \dfrac{1}{2}\left[1 + \dfrac{1}{(1-\alpha)^2}\right] \Rightarrow \dfrac{1}{2}\left[1 + \dfrac{169}{16}\right] = \dfrac{185}{32} \approx 5.78$

~~Part B~~

0	26
1	
2	54
3	
4	17
5	69
6	45
7	58
8	60
9	
10	32
11	
12	64

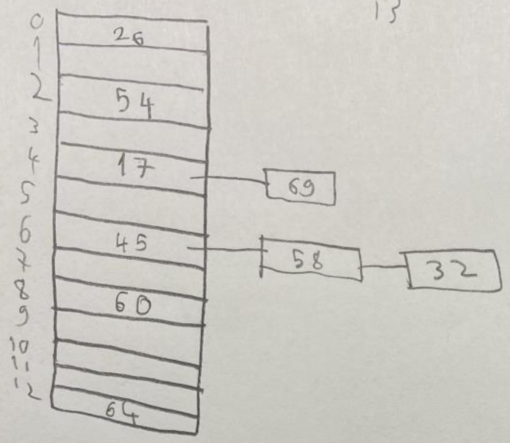
$\alpha = \dfrac{9}{13}$ (same)

Successful $= \left[\dfrac{1}{\alpha}\left(\ln\left(\dfrac{1}{1-\alpha}\right)\right)\right] = \dfrac{-\ln(1-\alpha)}{\alpha} = \dfrac{-\ln\left(\frac{4}{13}\right)}{9/13} \approx \boxed{1.7}$

Unsuccessful $= \dfrac{1}{1-\alpha} \Rightarrow \dfrac{1}{1-\frac{9}{13}} = \dfrac{13}{4} = \boxed{3.25}$

Part C

$\alpha = \frac{9}{13}$



$$\text{Successful} = 1 + \frac{\alpha}{2} = 1 + \frac{9}{26} = \frac{35}{26} \approx 1.35$$

$$\text{Unsuccessful} = \alpha = \frac{9}{13}$$

Question3) Algorithm Analysis For The Methods

A) Inserting a new flight(O(1))

Since we implemented our graph using adjacency list implementation(I used an array and each element in the array has an linked list) we can access the head of the linked list in O(1) time and quickly insert the new node. One bad way to implement this function would be to traverse the all list and add it to the end of the list. Then it would be O(n) where n is the total number of nodes(flights) from one airport.

B) listing flights(O(N))

There isn’t any shorter way to do this, we have to traverse all flights (all linked list). The good thing though (it is true for the first question as well) we don’t need to traverse the all array, since airport names are given as integers from 0 to n-1 we can access them in O(1) time. If we didn’t know where did we put the vertexes in the array we would have to traverse the array as well.

C) Shortest Path(O(V(V+E)))

Dijkstra’s shortest path algorithm is implemented in the same way it is present in our lecture slides. Basically, in the part where we find the least cost edge from an unvisited node, we traverse all the graph and, we do this until all the nodes are visited (from v = 2 to n). But if we did use a priority queue to find the smallest edge instead of traversing all the graph, we might have had a better time-complexity sth like O(V+ELogV). (From N^2 to nlogn).

D) Minimize costs(O(V(V+E)))

Prim’s algorithm is used. The longest part is similar to Dijkstra’s algorithm. For each vertex we search for all the graph, to find the least cost edge.