Assignment 7
Kai, Shi Yunfan, Li
12/05/2014

1. We cannot simply increase the size of the table to seven, because it will cause collision. Amy and Andy will have collision.
2. a. Abel = 4, Abigail = 8, Abraham = 17, Ada = 0, Adam = 0, Adrian = 17, Adrienne = 17, Agnes = 13, Albert = 1, Alex = 4, Alfred = 5, Alice = 8

b. Tablesize = 6

0	1	2	3	4	5
Ada	Agnes	Abigail		Abel	Abraham
Adam	Albert	Alice		Alex	Adrian
					Adrienne
					Alfred

Tablesize = 13

0	1	2	3	4	5	6	7	8	9	10	11	12
Ada	Albert			Abel	Alfred			Abigail				
Adam				Alex				Alice				
Agnes				Abraham								
				Adrienne								
				Adrian								

c. Load factor:

Tablesize = 6: 12/6 = 2
Tablesize = 13: 12/13 = 0.92

3. The value range of cosine is [-1, 1], and after getting integer there will only be three values, 0, -1 and 1. The negative value cannot be used. Therefore, only 0 and 1can be used and it’s highly possible to cause collision.

4. Adding the every first three letters of every weekday (a=0, z=25). We can get these values: Monday = 39, Tuesday = 43, Wednesday = 29, Thursday = 46, Friday = 31, Saturday = 37, Sunday = 51.
   By modding 9, we can get:

0	1	2	3	4	5	6	7	8
	Thursday	Wednesday	Monday	Friday		Sunday	Tuesday	

Adding every letters of months (a = 0, z= 25), we can get these values:
January = 90, February = 96, March = 43, April = 56, May = 39, June = 50, July = 68, August = 89, September = 103, October = 78, November = 94, December = 55
By modding 15, we can get:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Jan			Oct	Nov	Jun	Feb		Jul	May	Dec	Apr		Mar	Aug
													Sep	

5.

```
 int containsValue(hashMap * ht, valueType * val)
{
        int i;
        for(i = 0; i < ht->tableSize; i ++)
        {
                if(ht->table[i])
                {
                        hashLink * currLink;
                        while(table[i}
```

Big (O): O (n). Because it needs to run a loop.

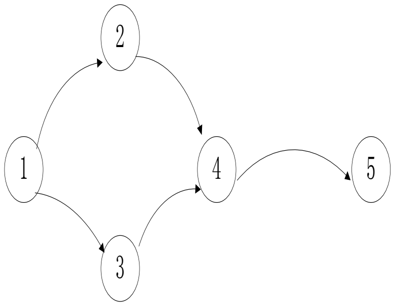
Graphs:

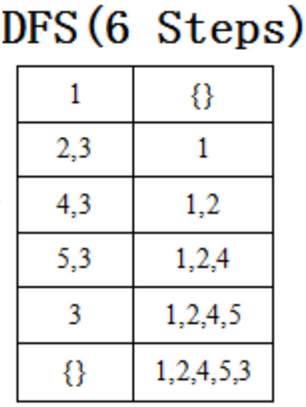
6. Adjacency matrix

	1	2	3	4	5	6	7	8
1	?	1	0	1	0	0	0	0
2	0	?	1	0	0	0	0	0
3	0	0	?	0	1	1	0	0
4	0	0	0	?	1	0	0	0
5	0	0	0	0	?	0	0	0
6	0	0	0	0	0	?	1	1
7	0	0	0	0	1	0	?	0
8	0	0	0	0	0	0	0	?

Edge list:
1: {2, 4}
2: {3}
3: {5, 6}
4: {5}
5: {}
6: {7, 8}
7: {5}
8: {}

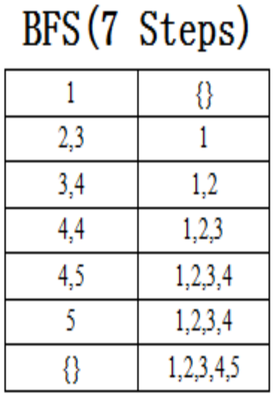
7. DFS has fewer steps than BFS:





DFS (6 Steps)

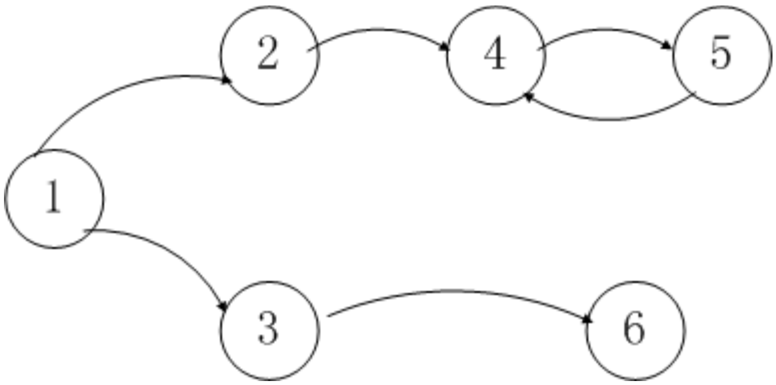
1	{}
2,3	1
4,3	1,2
5,3	1,2,4
3	1,2,4,5
{}	1,2,4,5,3



BFS (7 Steps)

1	{}
2,3	1
3,4	1,2
4,4	1,2,3
4,5	1,2,3,4
5	1,2,3,4
{}	1,2,3,4,5

BFS has fewer steps:



8. DFS

Step	Stack	Reachable
0	1	
1	6,2	1
2	11,2	1,6
3	16,12,2	1,6,11
4	21,12,2	1,6,11,16
5	22,12,2	1,6,11,16,21
6	23,17,12,2	1,6,11,16,21,22
7	17,12,2	1,6,11,16,21,22,23
8	12,12,2	1,6,11,16,21,22,23,17
9	13,12,2	1,6,11,16,21,22,23,17,12
10	18,8,12,2	1,6,11,16,21,22,23,17,12,13
11	8,12,2	1,6,11,16,21,22,23,17,12,13,18

BFS

Step	Queue	Reachable
0	1	
1	2,6	1
2	6,3,7	1,2
3	3,7,11	1,2,6
4	7,11,4,8	1,2,6,3
5	11,4,8	1,2,6,3,7

6	4,8,12,16	1,2,6,3,7,11
7	8,12,16,5,9	1,2,6,3,7,11,4
8	12,16,5,9,13	1,2,6,3,7,11,4,8
9	16,5,9,13,13,17	1,2,6,3,7,11,4,8,12
10	5,9,13,13,17,21	1,2,6,3,7,11,4,8,12,16
11	9,13,13,17,21,10,14	1,2,6,3,7,11,4,8,12,16,5
12	13,13,17,21,10,14	1,2,6,3,7,11,4,8,12,16,5,9
13	13,17,21,10,14,18	1,2,6,3,7,11,4,8,12,16,5,9,13

9.

Iteration	Priority Queue	Reachable with Costs
0	Pensacola (0)	{}
1	Phoenix (5)	Pensacola (0)
2	Pueblo (8), Peoria (9), Pittsburgh (15)	Phoenix (5)
3	Peoria (9), Pierre (11), Pittsburgh (15)	Pueblo (8)
4	Pierre (11), Pittsburgh (14), Pittsburgh (15)	Peoria (9)
5	Pendleton (13), Pittsburgh (14), Pittsburgh (15)	Pierre (11)
6	Pittsburgh (14), Pittsburgh (15)	Pendleton (13)
7	Pittsburgh (15)	Pittsburgh (14)
8	{}	---

10. Because Dijkstra's algorithm should get the most short distant form start vertex. The priority queue can get the most short distant - first element of priority queue.

11. The big-O of an edge-list is depend on the number of the edge of the graph. Big O is O(e). "e" is the number of the edge.

12. The big-O of an adjacency matrix is depend on the number of V. We should construct a V*V matrix to represents the relationship between every vertices. So the big-$O(V^2)$.

13. Only breadth-first search is guaranteed to find the path. Because breadth-first search first checks all paths of length1, then of length2, then of length3, etc....it's guaranteed to find a path containing the least steps from start to goal. But depth will stuck in the infinite path if the infinite path is traveled before the finite path.