Joseph Thornton

Professor Cheng

CSC 130

CSUS

11/29/2016

**Exercise 1)** hash table Data element(s) (separate changing)

(Note for all the tables the indices are only for reference and not part of the table blank spaces represent empty cells)

4371

1

0

1989

9679

4344

4199

6173

1323

9

8

7

6

5

4

3

2

Hash table Data element(s) (linear probing)

9679

4371

1

0

1989

4199

1323

9

8

7

6

5

4

3

2

4344

6173

Hash table Data element(s) (quadratic probing)

9679

4371

1

0

4344

6173

4199

1323

9

8

7

6

5

4

3

2

9

6

7

1989

8

4199

Hash table Data element(s) (double hashing using h(I,k))=h1(k)+(i)H2(k) where i is the number of hashes attempted and k is the data)

1989

4371

1

0

9

8

7

6

5

4

3

2

1323

9679

6173

4344

4199

Since 1989 hashes to index 9 and is odd and 7-1989 mod7 =6 we have 9+6+6++6…+i6 we know that the addition of two even numbers will always be even and an even and an odd will be odd therefore 9(odd)+i6(even)%10 is odd and we can only get 1 3 5 7 and 9 this table must rehashed to accommodate for this error perhaps we restrict h(i) with i<tablesize /m then rehash to a bigger table after this otherwise this will be infinite.

**Exercise 2)**

**1. Adjacency matrix**

A B C D E F

A 0 1 0 1 0 0

B 1 0 0 1 0 0

C 0 0 0 0 0 1

D 0 0 1 0 1 0

E 0 0 1 0 0 1

F 0 0 0 1 0 0

2. adjacency list

**A** (B,D)

**B** (A,D)

**C** (F)

**D** (E)

**E** (C,F)

**F** (D)

Exercise 3)

1. BFS
2. Deque A and enque(B,D) **1**. A has 2 adjacent nodes B,D (note the first enque of A was skipped)
3. Que(C,E) **1.** B has no adjacent nodes that are unmarked so we mark it and go to D. D is the second node that is dequed and it has 2 adjacent nodes that are enqued (C,E)
4. Que(F) **1.** C is dequed and has 1 adjacent node F that is enqued
5. E is the first node we deque and it only has 2 already marked adjacent nodes F is the second node and it is the final node with only 1 already marked node so we are done
6. DFS(Discovery time 1-6,finish time 1-6)

**Exercise 4)**

6

So the topological sort s,g,h,d,a,e,i,f,b,c,t

**Exercise 5)** Algorithm x is a binary integer represented as a base 2 binary value where m is prime and >=2, h(x) = xmod(m)

We assume that for some binary integers x and y where y==!x for only one bit, xmod(m) == ymod(m)

For any value of x y differs by 2^I where I>=0 and represents the bit #

So xmod(m)=(x(+-)2^i)mod(m) there are cases (+-)

Case 1: xmod(m)=xmod(m)+x+(2^i)mod(m) but 0=(2^i)mod(m) where i=1 and m=2.

cannot be contradicted so for some case where m>=2 and the integers differ by one bit the same hash value is possible.

**Exercise 6)**

No, to see this we need only one case where a graph does not find the shortest path using BFS

CASE:

Stack: A

Next we pop a and push its adjacent edges and mark it as distance 0

Stack: x,y,z

We know look at x it has not been marked we pop it and push its adjacent edges and mark it as distance 1

Stack: z,y,z

We then pop z and it has not been marked so we push its adjacent edges and mark it as distance 2

Stack: y,z

We pop y and it has not been marked so we mark it and push its adjacent edges and mark it as distance 1

now we pop z it has already been marked so we do nothing but its distance is actually 1 but we have stored a longer path in it already and its distance is 2 which is longer than the shortest path of 1.

1. 7 hrs
2. 5
3. The homework was good but the double hash didn’t work out well so we need some definitions for rehashing to finish it, the last question couldn’t be proven because there is a case where keys that differ by one bit can have the same hash not sure if I misunderstood it.