Data Structures and Algorithms – (COMP SCI 2C03)
Winter 2021
Tutorial - 6

Feb 22, 2021

1. Suppose we wish to search a linked list of length n, where each element contains a key k along with a hash value $h(k)$. Each key is a long character string. How might we take advantage of the hash values when searching the list for an element with a given key?
   Answer: String comparison takes longer than numerical value comparison. For instance if you have to compare strings $x = x_1x_2 \dots x_n$ and $y = y_1y_2 \dots y_n$, then to check if they are equal, you need to check if $\forall i \in [1..n]$, $x[i] = y[i]$. This takes n time.

   Therefore, when you traverse the linked list, instead of comparing the given string key with the string key at the node (long string comparison), it is better to compute the hash value of the given key and then simply compare it with the hash value of the key of the node. If they match, you have a search hit, and then can compare the string keys; otherwise you move to the next node in the linked list till either you find the node with the input key or the end of list in case of a search miss.

2. Which of the following scenarios leads to expected linear running time for a random search hit in a linear-probing hash table?

   a. All keys hash to the same index.

   b. All keys hash to different indices.

   c. All keys hash to an even-numbered index.

   d. All keys hash to different even-numbered indices.

Answer: A and C.

3. Consider inserting the keys

   `<10, 22, 31, 4, 15, 28, 17, 88, 59 >`

   into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k$. Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \mod (m-1))$.
   Answer:

   - Linear probing (C3P3 slide 16): See Table 1 for the answer. Here is the explanation for it. The hash function for linear probing discussed in class is $h(k, i) = (h'(k) + i) \mod m = (k + i) \mod 11$, for $i = 0, 1, \ldots, 10$. Based on it the keys in the given sequence would be inserted as follows:

     – Key 10 - First slot probed is $h(k, 0) = (10 + 0) \mod 11 = 10$. Since it is empty, key 10 is inserted in it.

     – Key 22 - First slot probed is $h(22, 0) = (22 + 0) \mod 11 = 0$. Since it is empty, key 22 is inserted in it.

     – Key 31 - First slot probed is $h(31, 0) = (31 + 0) \mod 11 = 9$. Since it is empty, key 31 is inserted in it.

     – Key 4 - First slot probed is $h(4, 0) = (4 + 0) \mod 11 = 4$. Since it is empty, key 4 is inserted in it.

     – Key 15 - First slot probed is $h(15, 0) = (15 + 0) \mod 11 = 4$. However since slot 4 is already filled, we fill the key 15 in the next available slot; that is, $h(15 + 1) \mod 11 = 5$.

     – Key 28 - First slot probed is $h(28, 0) = (28 + 0) \mod 11 = 6$. Since it is empty, key 28 is inserted in it.

22	88			4	15	28	17	59	31	10

Table 1: Solution for Q3 Linear probing

- Key 17 - First slot probed is $h(17, 0) = (17+0) \mod 11 = 6$, However since slot 6 is already filled, we fill the key 17 in the next available slot; that is, $h(17 + 1) \mod 11 = 7$.

- Key 88 - First slot probed is $h(88, 0) = (88+0) \mod 11 = 0$. However since slot 0 is already filled, we fill the key 88 in the next available slot; that is, $h(88 + 1) \mod 11 = 1$.

- Key 59 - First slot probed is $h(59, 0) = (59+0) \mod 11 = 4$. However since slot 4 is already filled, we check the next empty slot available at $h(59 + 4) \mod 11 = 8$.

• Quadratic probing with $c_1 = 1$ and $c_2 = 3$ (C3P3 slide 17): See Table 2 for the answer. Here is the explanation for it. The hash function for quadratic probing discussed in class was $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \mod m = (k + i + 3i^2) \mod 11$, for $i = 0, 1, \ldots, 10$. Based on it the keys in the given sequence would be inserted as follows:

- Key 10 - First slot probed is $h(k, 0) = (10+0) \mod 11 = 10$. Since it is empty, key 10 is inserted in it.

- Key 22 - First slot probed is $h(22, 0) = (22+0) \mod 11 = 0$. Since it is empty, key 22 is inserted in it.

- Key 31 - First slot probed is $h(31, 0) = (31+0) \mod 11 = 9$. Since it is empty, key 31 is inserted in it.

- Key 4 - First slot probed is $h(4, 0) = (4 + 0) \mod 11 = 4$. Since it is empty, key 4 is inserted in it.

- Key 15 - First slot probed is $h(15, 0) = (15+0) \mod 11 = 4$. However since slot 4 is already filled, we fill the key 15 in the

22		88	17	4		28	59	15	31	10

Table 2: Solution for Q3 Quadratic probing

next available slot; that is, $h(15, 1) = (15+1+3) \mod 11 = 8$.

- Key 28 - First slot probed is $h(28, 0) = (28 + 0) \mod 11 = 6$. Since it is empty, key 28 is inserted in it.

- Key 17 - First slot probed is $h(17, 0) = (17 + 0) \mod 11 = 6$, However since slot 6 is already filled, we fill the key 17 in the next available slot; that is, $h(17, 3) = h(17 + 3 + 27) \mod 11 = 3$.

- Key 88 - First slot probed is $h(88, 0) = (88 + 0) \mod 11 = 0$. However since slot 0 is already filled, we fill the key 88 in the next available slot; that is, $h(88, 8) = (88 + 8 + 192) \mod 11 = 2$.

- Key 59 - First slot probed is $h(59, 0) = (59 + 0) \mod 11 = 4$. However since slot 4 is already filled, we check the next empty slot available at $h(59, 2) = (59 + 2 + 12) \mod 11 = 7$.

• Double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \mod (m - 1))$ (C3P3 slide 18): See Table 3 for the answer. Here is the explanation for it. The hash function for double hashing discussed in class was $h(k, i) = (h_1(k) + ih_2(k)) \mod m = (k + i(1 + k \mod 10)) \mod 11$, for $i = 0, 1, \ldots, 10$. Based on it the keys in the given sequence would be inserted as follows:

- Key 10 - First slot probed is $h(k, 0) = (10 + 0) \mod 11 = 10$. Since it is empty, key 10 is inserted in it.

- Key 22 - First slot probed is $h(22, 0) = (22 + 0) \mod 11 = 0$. Since it is empty, key 22 is inserted in it.

- Key 31 - First slot probed is $h(31, 0) = (31 + 0) \mod 11 = 9$.

22		59	17	4	15	28	88		31	10

Table 3: Solution for Q3 Double hashing

Since it is empty, key 31 is inserted in it.

- Key 4 - First slot probed is $h(4,0) = (4+0) \mod 11 = 4$. Since it is empty, key 4 is inserted in it.

- Key 15 - First slot probed is $h(15,0) = (15+0) \mod 11 = 4$. However since slot 4 is already filled, we fill the key 15 in the next available slot; that is,

$$h(15,2) = (15 + 2 * (1 + (15 \mod 10))) \mod 11 = 5.$$

- Key 28 - First slot probed is $h(28,0) = (28+0) \mod 11 = 6$. Since it is empty, key 28 is inserted in it.

- Key 17 - First slot probed is $h(17,0) = (17+0) \mod 11 = 6$, However since slot 6 is already filled, we fill the key 17 in the next available slot; that is,

$$h(17,1) = (17 + 1 * (1 + (17 \mod 10))) \mod 11 = 3.$$

- Key 88 - First slot probed is $h(88,0) = (88+0) \mod 11 = 0$. However since slot 0 is already filled, we fill the key 88 in the next available slot; that is,

$$h(88,2) = (88 + 2(1 + (88 \mod 10))) \mod 11 = 7.$$

- Key 59 - First slot probed is $h(59,0) = (59+0) \mod 11 = 4$. However since slot 4 is already filled, we check the next empty slot available at $h(59,2) = (59+2*(1+(59 \mod 10))) \mod 11 = 2$.

4. What does the BFS tree tell us about the distance from v to w when neither is at the root?
Answer: BFS computes the shortest distance from source s to all vertices connected to s. (Page 541, proposition B) If s is connected to v and w then the distance between v and w is less than or equal to the sum of shortest distances between s, v and s, w.

5. Suppose you use a stack instead of a queue when running breadth-first search. Does it still compute shortest paths?
Answer: No, it does not compute the shortest path. Counterexample in page 533.

6. Draw the output tree for the graph given in Figure 1 when

   - DFS is called on the source vertex 0.
   - BFS is called on the source vertex 0.

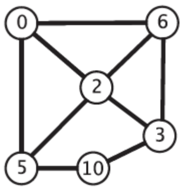

Figure 1: Question 6

Answer: textbook page 533-540

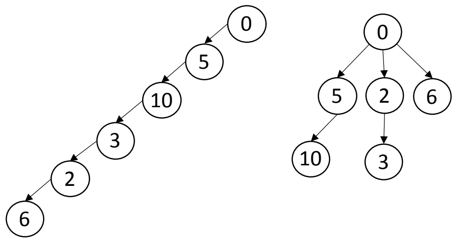

Figure 2: Answer of question 6

DFS. 0-5-10-3-2-6
BFS. 0-5-2-6-10-3

7. True or false: The reverse postorder of a digraph's reverse is the same as the postorder of the graph.
Answer: False.

   Postorder: 3-2-1 Reverse postorder of graph's reverse: 1-3-2

8. Let G be the graph shown in Figure 4.

   a. What is the preorder vertex ordering of G.

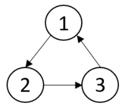

Figure 3: Counterexample for question 7

b. What is the postorder vertex ordering of G.

c. What is the reverse postorder vertex ordering of G.

d. What is the topological sort of G.

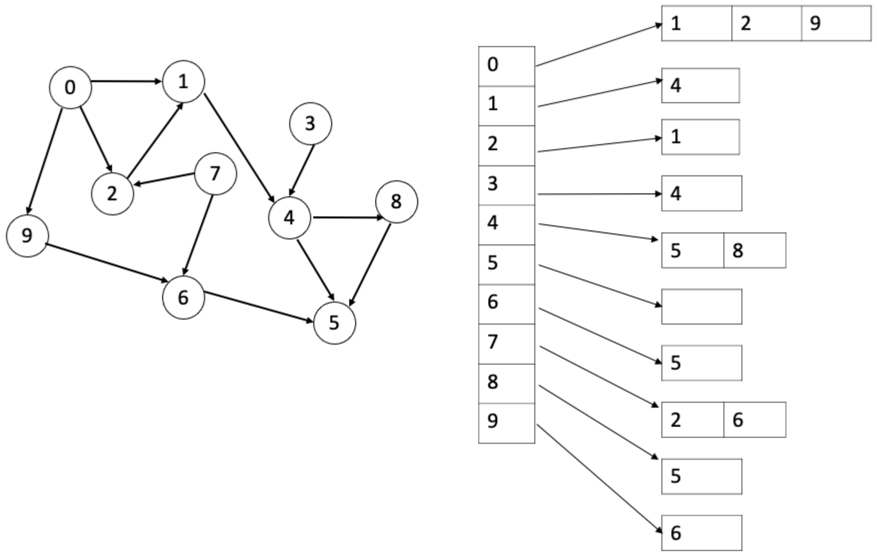

Figure 4: Question 8

Answer: textbook page 579-581
a. 0-1-4-5-8-2-9-6-3-7
b. 5-8-4-1-2-6-9-0-3-7
c. 7-3-0-9-6-2-1-4-8-5
d. 7-3-0-9-6-2-1-4-8-5

9. What are the strong components of a DAG? What happens if you run Kosaraju's algorithm on a DAG?

   Answer: textbook page 584-586
   Strong components are the maximal subsets of vertices that are strongly

connected to one another. Since a DAG has no loop, every vertex in the DAG is a strongly connected component. So Kosaraju's algorithm will return each vertex of the DAG as a strongly connected component.