# CS 211 Homework #3

Note that this is an individual assignment and all work must be your own. Be sure to show your work when appropriate. This assignment is due **in class** on Thursday, December 7, 2017.

1. [3] Show the result of quicksort after one iteration of the quicksort algorithm (until I >= J and pivot is swapped back).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 9 | 12 | 1 | 2 | 15 | 11 | 8 | 10 | 17 | 4 | 6 | 5 | 13 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

2. [3] Perform radix sort on the following numbers:

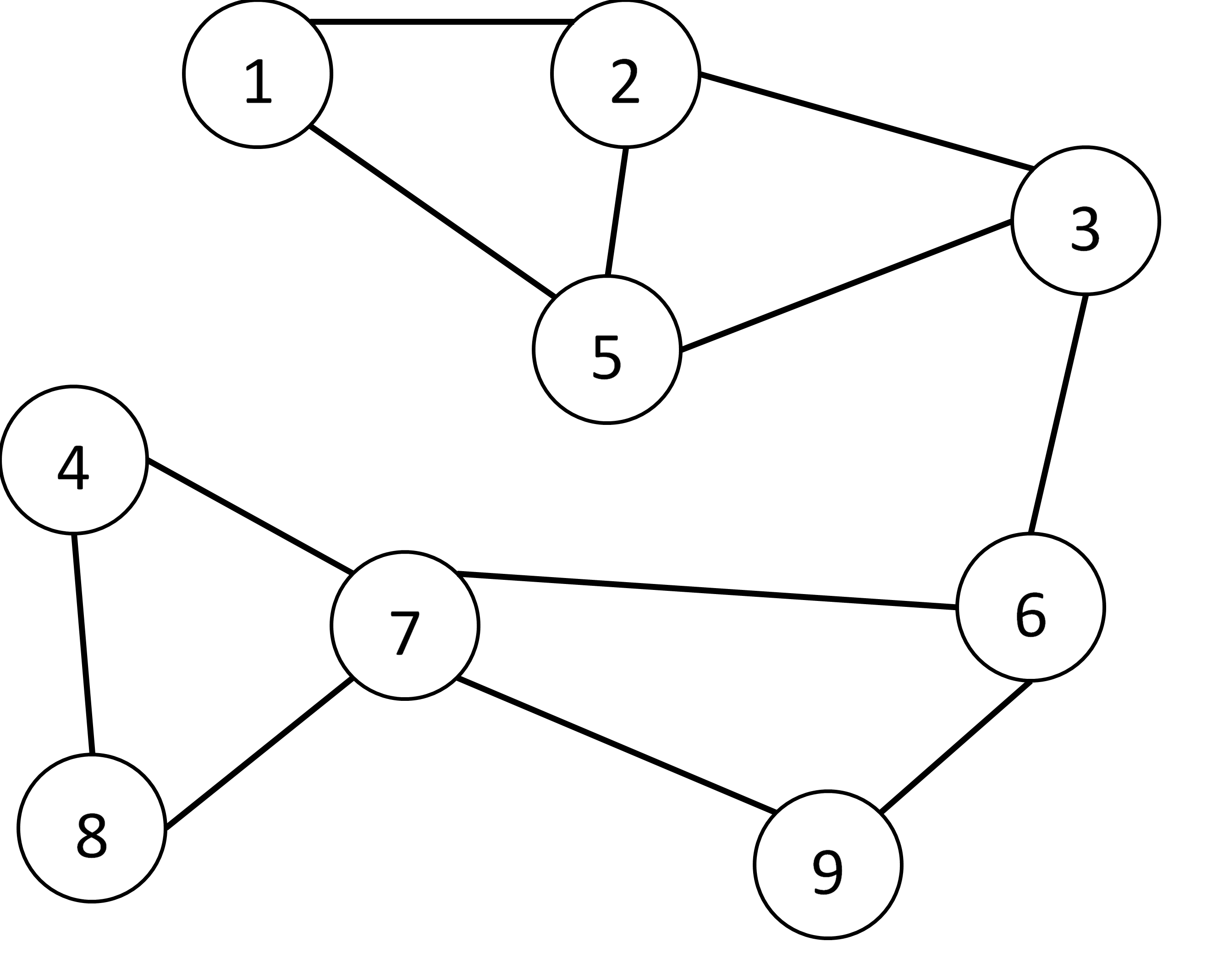
101, 98, 27, 15, 333, 234, 531, 503, 122, 432, 199, 200, 155, 188, 79, 631

|  |  |  |  |
| --- | --- | --- | --- |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |

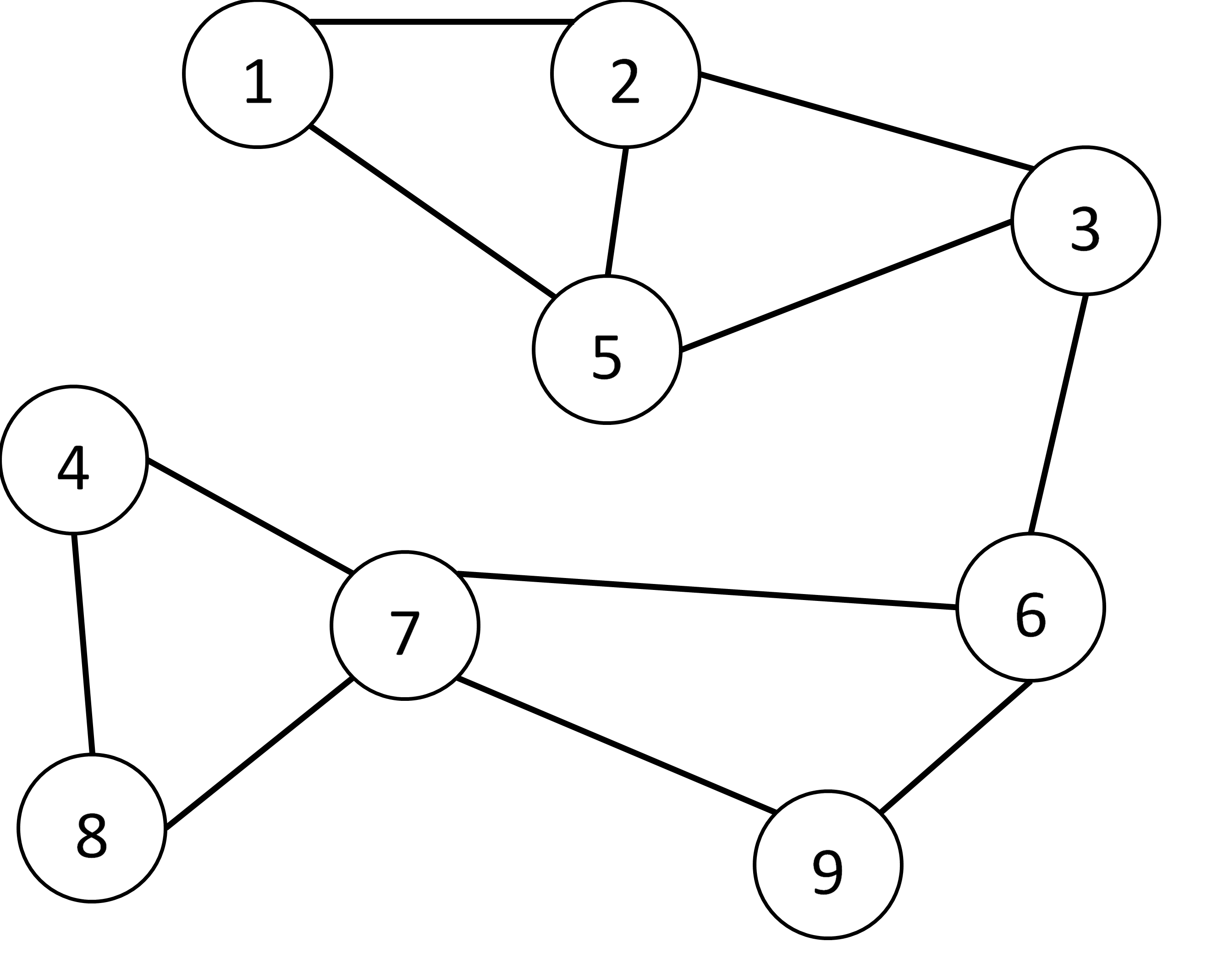
3. [3] Order the following sorting algorithms from worst to best case algorithmic complexity:

Merge Sort, Bubble Sort, Radix Sort, Quick Sort, Shell Sort, Insertion Sort, Heap Sort, Selection Sort, Tree Sort, Shaker Sort

4. [3] Draw the BFS search tree for the following graph starting at vertex 7.



5. [3] Articulation Points. Draw the DFS articulation tree for the following graph starting at vertex 7. Circle any articulation points in your tree.



6. [1] What data structure allow us to perform a DFS on a graph?

7. [1] What data structure allows us to perform a BFS on a graph?

10. [1] Which of the following is a **false** statement about hash tables:

A) Hash tables provide O(1) lookup

B) Hash table entries are ordered based on when they are placed into the data structure (i.e. FIFO, similar to a vector)

C) Hash tables allow for random access (i.e. similar to a vector)

D) Under normal circumstances, hash tables take up more memory than a vector.

11. [3] List the three key factors that affect the runtime performance of a hash table:

14. [3] The item "B" hashes to array index 3. Insert B into the following **hopscotch** hashtable whose max distance is 4.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | C | E | G | D | H | I |  |  |  |
| 1100 | 0100 | 0010 | 1000 | 0000 | 1100 | 0000 |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

RESULT:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

[4] Given a hashing function hash(x) = ((x \* x) + x) % 11, Insert the value 4 into each hash table using the rules specified below. Note that some of the boxes in each hash table are already full.

Linear Probing having probe(i) = (i + 1) % 11  
where i = index location

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 11 | 1 | 9 |  |  | 2 |  |  | 6 |  |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |

Quadratic Probing having probe(i) = ((i^2 + i) + 1) % 11  
where i = index location

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 11 | 1 |  |  |  | 2 | 9 |  | 6 |  |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |