Given the BST to the right, provide the following (Each is independent of the others) (8pts)

* 1. Show the result of inserting 17
  2. Show the result of removing 27

30

* 1. Show the result of removing 50
  2. Show the result of removing 20

23

27

45

40

25

15

20

50

1. For the graph to the right, provide the following (each is independent of the others) (15pts)
   1. Show the node order and spanning tree that results from BFS(5)
   2. Show the node order that results from BFS(4)
   3. Show the node order that results from the DFS(5)
   4. Show the node order that results from the DFS(4)
   5. Construct an adjacency matrix representation for this graph
   6. Construct an adjacency list representation for this graph
   7. Find the shortest path from 1 to all other nodes

1

2

3

5

4

6

1. Below is the adjacency matrix for a directed weighted graph. Draw the visual representation of the graph it represents (6pts)

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

1. What big O family would you classify the following code fragment as? Why? (6pts)

while(val > 0)

{

sum += val;

val /= 2;

}

1. The below code fragments can be used to implement a list with LIFO behavior(6pts)

N++ | val = L[n];

L[n] = val | n--;

This code is simple however your boss believes you need to go through all the extra pieces of work and fully implement a stack and follow stack naming conventions. Why would that be preferable to the easy to use and implement code above?

There is no safety. All this code does is increment and decrement a count to keep track of where to pull and push information, the pulled information is never used, so if the program were to lose its spot (the value of n) the data stored in object L would be of no value. In addition, a significant amount of space could be wasted unintentionally if, for example, L is an array and n is an integer and let’s say the program pushed 100 integers onto the “stack” then popped them off and then never, through the rest of the program through pushing and popping, incremented n by more than some small number, like 5. Enough space to store 95 integers (or some other data type) would be wasted.

1. Now for a little bit of a thinker. It is possible that we can see a O(n^2) algorithm perform a task faster then a O(n) algorithm. Explain how that may be possible and do these sorts of cases show O() is not useful. (6pts)

Absolutely. O() has no bearing on performance. If each iteration of a linear algorithm takes some amount of time to execute and each iteration of an exponential algorithm takes some arbitrarily significantly smaller amount of time to execute, it would very probably run faster in most cases. This however, does not mean that O() is not valuable, it just means that when comparing algorithm’s run-time a number of factors other than their complexity should be taken into consideration. For example, you could use bubble sort on a small number of integers on a fast computer and possibly complete your task faster than using merge sort on a complex datatype on a slow computer. This doesn’t mean that knowing the complexities of these sorts isn’t useful.

1. Show the state of each listed data structure after the insertion of the following data

5, 2, 7, 13, 4

Then show me the state of the structure after 1 REMOVE operation. Diagrams are preferable, ensure whatever you choose is very clearly marked. (14pts)

Stack

Max heap

Binary search tree

Hashtable(5 buckets, mod number by 5 for hash algorithm, use chaining for collisions)

1. Linked list operations. Write a code fragment for each (code fragment, not entire program!)(8pts)

* p, q, and r are pointers into the list
* Each node in the list has a val and next field
* Assume p, q, and r are initialized properly
  1. Print the larger of the 2 values of p and q
  2. Insert r between p & q in the list, where p & q are consecutive items
  3. Assume p and q point to items in a linked list and that p is to the left of q (p is before q). Write a loop that forces r to visit each node between p and q
  4. p points to a node in the middle of the list(not beginning or end). Delete the node after p.

1. In class, we discussed both Prim’s & Kruskal’s algorithms and how they may result in the same or different MST. Provide a graph where they will provide a different MST. (6pts)
2. Consider the QuickSort algorithm. Show a reasonable state of the list after the first round of choosing a pivot point and partitioning the list(6pts)

List: 17, 45, 11, 30, 47, 12, 33

1. When studying O() we always end up with terms like O(N), O(N^2), O(N^3) even though there are other statements in the code that make up the overall efficiency of the program. For instance take the code below

for(i = 0; i < x; i++)

{

for(z = 0; z < x; z++)

{

Val \*= 2;

Tmp += 1;

}

}

while(x != 0)

{

Val –= x;

x--;

}

Return 0;

Would become O(2n^2 + 2n + 1). Why does it make sense to drop all of the other numbers and just take the highest-powered term? In this case O(n^2). (6pts)

1. Given the graph below, find the solutions to the requested operations and show the process you took to reach those solutions in as much detail as possible(13pts)

1

2

3

5

4

6

5

2

3

4

4

3

1

2

* 1. Show the steps in solving the shortest path problem according to Dijkstra’s algorithm when starting at node 6.
  2. Construct the MST for the graph according to Prim’s algorithm starting at vertex 1
  3. Construct the MST according to Kruskal’s algorithm