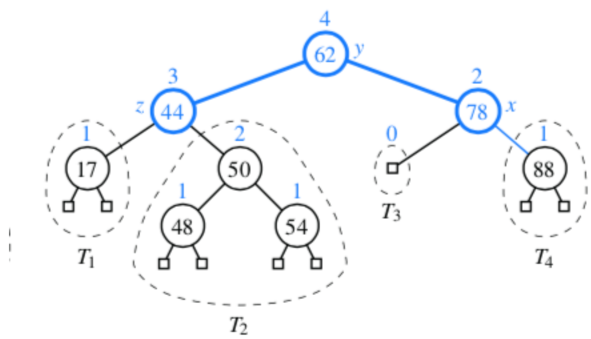
CS206
Final set of review questions

Hashtables and Maps

- Which of the hash table collision-handling schemes could tolerate a load factor above 1 and which could not?
- Draw the 11-entry hash table that results from using the hash function, $h(i) = (3i+5) \bmod 11$, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining.
- What is the result of the previous exercise, assuming collisions are handled by linear probing?
- Show the result of Q2 assuming collisions are handled by quadratic probing, up to the point where the method fails.
- What is the result of Exercise R-10.6 when collisions are handled by double hashing using the secondary hash function $h'(k) = 7 - (k \bmod 7)$?
- What is the worst-case time for putting n entries in an initially empty hash table, with collisions resolved by chaining? What is the best case?
- Explain why a hash table is not suited to implement a sorted map.
- Describe how to perform a removal from a hash table that uses linear probing to resolve collisions where we do not use a special marker to represent deleted elements. That is, we must rearrange the contents so that it appears that the removed entry was never inserted in the first place.

Binary Search Trees and AVL Trees

- If we insert the entries $(1,A)$, $(2,B)$, $(3,C)$, $(4,D)$, and $(5,E)$, in this order, into an initially empty binary search tree, what will it look like?
- Insert, into an empty binary search tree, entries with keys 30, 40, 24, 58, 48, 26, 11, 13 (in this order). Draw the tree after each insertion.
- How many different binary search trees can store the keys{1,2,3}?
- Dr. Amongus claims that the order in which a fixed set of entries is inserted into a binary search tree does not matter—the same tree results every time. Give a small example that proves he is wrong.
- Dr. Amongus claims that the order in which a fixed set of entries is inserted into an AVL tree does not matter—the same AVL tree results every time. Give a small example that proves he is wrong.

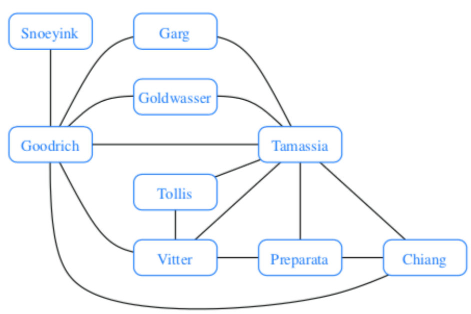


- Draw the AVL tree resulting from the insertion of an entry with key 52 into the AVL tree of the above figure.
- Draw the AVL tree resulting from the removal of the entry with key 62 from the AVL tree of Figure 11.13b.
- Explain why you would get the same output in an inorder listing of the entries in a binary search tree, T , independent of whether T is maintained to be an AVL tree, or not.
- Explain how to use an AVL tree or a to sort n comparable elements in $O(n \log n)$ time in the worst case.

Graphs

- Would you use the adjacency matrix structure or the adjacency list structure in each of the following cases? Justify your choice.
  - The graph has 10,000 vertices and 20,000 edges, and it is important to use as little space as possible.
  - The graph has 10,000 vertices and 20,000,000 edges, and it is important to use as little space as possible.
  - You need to answer the query getEdge(u, v) as fast as possible, no matter how much space you use.

- Explain why the DFS traversal runs in $O(n^2)$ time on an n-vertex simple graph that is represented with the adjacency matrix structure.
- Computer networks should avoid single points of failure, that is, network vertices that can disconnect the network if they fail. We say an undirected, connected graph G is *biconnected* if it contains no vertex whose removal would divide G into two or more connected components. Give an algorithm for adding at most n edges to a connected graph G, with $n \geq 3$ vertices and $m \geq n-1$ edges, to guarantee that G is biconnected. Your algorithm should run in $O(n + m)$ time.



-

- Draw an adjacency matrix representation of the undirected graph shown above. You may add labels to links if you find it convenient
- Draw an adjacency list representation of the undirected graph shown above. You may add labels to links if you find it convenient
- Suppose we represent a graph G having n vertices and m edges with the edge list structure. Why, in this case, does the insertVertex method run in $O(1)$ time while the removeVertex method runs in $O(m)$ time
- Let G be an undirected graph with n vertices and m edges. Write a method traversing each edge of G exactly once in each direction. The method should run in O(m+n) time. You may use choose a graph representation. How would this algorithm be changed for a directed graph?

- Let G be an undirected graph whose vertices are the integers 1 through 8 , and let the adjacent vertices of each vertex be given by the table below:

  vertex

  - 1 (2, 3, 4)
  - 2 (1, 3, 4)
  - 3 (1, 2, 4)
  - 4 (1, 2, 3, 6)
  - 5 (6, 7, 8)
  - 6 (4, 5, 7)
  - 7 (5, 6, 8)
  - 8 (5, 7)

Assume that, in a traversal of G, the adjacent vertices of a given vertex are returned in the same order as they are listed in the table above.

  - Draw G.
  - Give the sequence of vertices of G visited using a DFS traversal starting at vertex 1.
  - Give the sequence of vertices visited using a BFS traversal starting at vertex 1.

- The time delay of a long-distance call is determined by the number of communication links on the telephone network between the caller and callee. The engineers of RT&T want to compute the maximum possible time delay that may be experienced in a long-distance call. Write method that computes the **maximum** number of links (when cycles are disallowed, between some caller A and every other caller.

- For the implementation of Graph, Link and Node add a method

  public void removeLink(Link<E> link)

  in an appropriate place. Where is that "appropriate" place?

- Reimplement Dijkstra's shortest path method (in the Graph class on the class website) using a Stack rather than a Priority Queue. Doing so saves some time in that stacks have O(1) time for push and pop whereas PriorityQueues have O(lg n) time for the equivalent operations. How do the other adjustments you needed to make to the function to ensure that you got the shortest path affect its runtime?

- Write a method to do a breadth first traversal for a graph (starting at a node). Why might a breadth first traversal be preferred to a depth first traversal?