COSC 2123/1285 Algorithms and Analysis
Tutorial 1
Introduction to algorithms analysis and fundamental data structures

1 Information

- Additional exercises are available in the book if you need more practice.
- Hints are available for each question at the end of the book.

2 Additional Resources

- Check the discussion boards on blockboard to see if your question has been answered. If not, discuss problems on the discussion boards.
- See the lecturer during consutltation hours.

3 Remainders

For next week:

- Buy the text book (the subject follows the textbook very closely).
- Read chapters 2.1-2.6.

---

4 Questions

1.4.2 If you have to search for an element in a list/collection of n numbers, how can you take advantage of the fact that the list is known to be sorted? Give separate answers for:

a. lists represented as arrays.

b. lists represented as linked lists.

Answer:

1. Use binary search. We will learn about this soon, but we use a divide and conquer approach to take advantage of the sortedness of the array.
2. When searching in a sorted linked list, stop as soon as an element greater than or equal to the search key is encountered.

1.4.3a Show the stack after each operation of the following sequence that starts with the empty stack:

push(a), push(b), pop, push(c), push(d), pop

Answer: Push(x) puts x on the top of the stack, pop deletes the item from the top of the stack. Final state of stack is [c, a], where 'c' is the top of the stack.

1.4.3b Show the queue after each operation of the following sequence that starts with the empty queue:

enqueue(a), enqueue(b), dequeue, enqueue(c), enqueue(d), dequeue

Answer: Enqueue(x) adds x to the rear of the queue, dequeue deletes the item from the front of the queue. Final state of queue is [d, c], where 'c' is the front of that queue.

1.4.4a Let A be the adjacency matrix of an undirected graph. Explain what property of the matrix indicates that

a. the graph is complete, i.e., there is an edge between every pair of distinct vertices.

b. the graph has a loop, i.e., an edge connecting a vertex to itself.

c. the graph has an isolated vertex, i.e., a vertex with no edges incident to it.

Answer:

1. A graph is complete if and only if all the elements of its adjacency matrix except those on the main diagonal are equal to 1, i.e., $A[i, j] = 1$ for every $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$
2. A graph has a loop if and only if its adjacency matrix has an element equal to 1 on its main diagonal, i.e., $A[i, i] = 1$ for some $1 \leq i \leq n$.
3. An (undirected, without loops) graph has an isolated vertex if and only if its adjacency matrix has an all-zero row.

1.4.4b Answer the same questions as in a. for the adjacency list representation.

Answer:

1. A graph is complete if and only if each of its linked lists contains all the other vertices of the graph.
2. A graph has a loop if and only if one of its adjacency lists contains the vertex defining the list.
3. An (undirected, without loops) graph has an isolated vertex if and only if one of its adjacency lists is empty.

1.4.9 For each of the following applications, indicate the most appropriate data structure:

a. answering telephone calls in the order of their known priorities.

b. sending backlog orders to customers in the order they have been received.

c. implementing a calculator for computing simple arithmetical expressions.

Answer:

1. A priority queue
2. A queue
3. Reverse Polish notation (or just RPN) is a mathematical notation wherein every operator follows all of its operands. It is also known as Postfix notation. Example: 3 4 + equals 3 + 4.

   A stack (and reverse Polish notation). $3 + 4 \rightarrow$ push(3); push(4); push(+)

1.4.10 Design an algorithm for checking whether two given words are anagrams, i.e., whether one word can be obtained by permuting the letters of the other. For example, the words *tea* and *eat* are anagrams.

Answer: The most straightforward solution is to search for each successive letter of the first word in the second one. If the search is successful, delete the first occurrence of the letter in the second word, stop otherwise.

Another solution is to sort the letters of each word and then compare them in a simple parallel scan.

We can also generate and compare "letter vectors" of the given words: $V_w[i]$ = the number of occurrences of the alphabet's ith letter in the word w. Such a vector can be generated by initializing all its components to 0 and then scanning the word and incrementing appropriate letter counts in the vector.

