COSC581 - Algorithms
Spring 2023
Homework #4 Solutions

1. Data structures review -
    a. Describe the difference between the binary search tree property and the min-heap property.

Binary Search Tree Property: Let x be a node in a binary search tree. If y is a node in the left subtree of x, then y.key <= x.key. If y is a node in the right subtree of x,then y.key >= x.key.

Min-Heap Property: Given an array A, for every node i other than the root, A[parent(i)] <= A[i].

source: textbook pg. 153 and 287

The BST property states that all the nodes in the left subtree are less than (or equal to) their parent and all the nodes in the right subtree are greater than (or equal to) their parent. This is different than the min-heap property which states that a parent must be less than all of its children. With the min-heap property we do not have any guarantees between the subtrees like we do with the BST property. So in a min heap we cannot say that any elements on the left are less than elements on the right and vice-versa.

    b. Can a stack be implemented using a queue? If so, describe how.

To implement a stack using one queue, we need to take advantage of the queue's FIFO policy. When we add an element to the queue, we need to reorder the queue to have the last element first. We can do this by popping off the first element (dequeue) and then putting that element back onto the queue at the end (enqueue). If we do that n-1 times, the last element inserted will be first like a stack. See the pseudo code on the next page.

Running Time:
It will be O(n) for push because we have to iterate through all n elements in order to create a correct stack from the old queue. It will be O(1) for pop because we are just dequeuing the first element in the list which should be correct for a stack.

Pseudo code:

```
class Stack{
        Queue q
        def push(a){
                q.enqueue(a) //put the element into back of queue
                for iter in range(len(q.size()-1)){
                        q.enqueue(q.dequeue()) //circle all elements before it to the back of the
                queue behind new element
                }
        }
        def pop(){
                if q is not empty{
                        return q.dequeue() //pop off first element which should be LRU
                }
                else{
                        return NaN
                }
        }
}
```

c. Can dictionaries/maps be implemented using a hash table? If so, describe why.

Yes, dictionary/map data structures can (and often are) implemented using hash tables. Recall, dictionary/map data structures have {***key***, ***value***} pairs. We can easily hash the ***key*** and use this hash value to insert the ***value*** into the table. For example,

{ID1 : 5}, {ID2 : 6}, {ID4 : 3} → hash(ID1) = 5, hash(ID2) = 1, hash(ID3) = 7

0	1	2	3	4	5	6	7
	6				5		3

2. Dynamic Programming -

DNA sequences are made up of four different amino acids: Adenine, Cytosine, Guanine, and Thymine (denoted A, C, G, T respectively). We can tell the basic genetic differences between two organisms by measuring the distance between their respective DNA strands. One method for measuring this is called global alignment. When we produce a global alignment, we weight our decisions based on the number of matches, mismatches and gaps. For example:

s1 = AGCTTTGAA, s2 = AGGTTGGCAA

AG--CTTTG----AA
AGG--TT--GGCAA

In this example, we weighed match=1, gap=0, and mismatch=-1, for a global alignment score equal to 7 (7 matches - green, 5 gaps - black, 0 mismatches - red).

For your homework, you need to show the full dynamic programming table for aligning the sequences: s1=GCTTACGTGACG, and s2=GCTATCGCGACC. Use **match=2**, **gap=-1**, and **mismatch=-1**. Show all your work.

***Please note that in sequence alignment a λ is prepended to the beginning of each sequence.*

	λ	G	C	T	T	A	C	G	T	G	A	C	G
λ	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
G	-1	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
C	-2	1	4	3	2	1	0	-1	-2	-3	-4	-5	-6
T	-3	0	3	6	5	4	3	2	1	0	-1	-2	-3
A	-4	-1	2	5	5	7	6	5	4	3	2	1	0
T	-5	-2	1	4	7	6	6	5	7	6	5	4	3
C	-6	-3	0	3	6	6	8	7	6	6	5	7	6
G	-7	-4	-1	2	5	5	7	10	9	8	7	6	9
C	-8	-5	-2	1	4	4	7	9	9	8	7	9	8
G	-9	-6	-3	0	3	3	6	9	8	11	10	9	11
A	-10	-7	-4	-1	2	5	5	8	8	10	13	12	11
C	-11	-8	-5	-2	1	4	7	7	7	9	12	15	14
C	-12	-9	-6	-3	0	3	6	6	6	8	11	14	14

GCT--TACGTGACG
GCTAT--CGCGACC
Score = 14