EECE 7205-Assignment 3

Sreejith Sreekumar: 001277209

October 30, 2018

1 Qn1. Order Statistics

1.1 Code

```
#include <iostream>
#include <cstdlib>
#include <algorithm>
#include <iostream>
#include <cstdlib>
#include <bits/stdc++.h>
#include <time.h>

using namespace std;


/**
 *  Rand Select and helper functions
 */

int randomPartition(int* data, int begin, int end){

  srand(time(NULL));

  int pivot_index = begin + rand() % (end-begin+1);
  int pivot = data[pivot_index];
  swap(data[pivot_index], data[end]);

  pivot_index = end;
  int i = begin - 1;

  for(int j=begin; j<=end-1; j++)
    {
      if(data[j] <= pivot)
        {
          i = i+1;
          swap(data[i], data[j]);
        }
```

```
    }

  swap(data[i+1], data[pivot_index]);
  return i+1;

}


int randomized_select(int* data, int p, int r, int i){

  if (p == r){
    return data[p];
  }

  int q = randomPartition(data, p, r);
  int k = q - p + 1;

  if(i == k){
    return data[q];
  } else if (i < k){
    return randomized_select(data, p, q-1, i);
  } else {
    return randomized_select(data, q+1, r, i-k);
  }

}


/**
 *  Select with linear worst case running time
 *         and helper functions
 */

int findMedian(int data[], int limit) {
    sort(data, data+limit);
    return data[limit/2];
}


void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}


int partition(int data[], int left, int right, int x) {
```

```
  int i;
  for (i=left; i<right; i++)
    if (data[i] == x)
      break;
  swap(&data[i], &data[right]);

  i = left;
  for (int j = left; j <= right - 1; j++)
    {
      if (data[j] <= x)
        {
          swap(&data[i], &data[j]);
          i++;
        }
    }
  swap(&data[i], &data[right]);
  return i;

}

int linear_select(int data[], int left, int right, int k) {

    if (k > 0 && k <= right - left + 1) {

        int n = right - left + 1 ;
        int idx, median[(n+4)/5];

        for (idx=0; idx<n/5; idx++) {
            median[idx] = findMedian(data+left+idx*5,
                                     5);
        }

        if (idx*5 < n) {
            median[idx] = findMedian(data+left+idx*5,
                                     n%5);
            idx++;
        }

        int medOfMed = (idx == 1)? median[idx-1] : linear_select(median, 0, idx-1, idx/2);

        int pos = partition(data,
                            left,
                            right,
                            medOfMed);

        if (pos-left == k-1)
            return data[pos];
```

```
        if (pos-left > k-1)
            return linear_select(data,
                                 left,
                                 pos-1,
                                 k);
        return linear_select(data,
                             pos+1,
                             right,
                             k-pos+left-1);
    }

    return INT_MAX;
}


int main()
{

  int size = 100;

  int input[size];

  for(int i=0;i<100;i++){
    input[i] = i+1;
  }

  /*
   * Create a random permutation of the numbers in the input array
   */
  random_shuffle(&input[0], &input[size-1]);

  int i;
  cout<<"Input i for finding the ith smallest element from the random partition: ";
  cin>>i;

  cout<<"Finding ith smallest number from the array using random select...\n";

  /**
   * Call for finding ith smallest (i+1th since we start from 0)
   * number using randomized select
   */
  int rand_select_op = randomized_select(input, 0, 100, i+1);
  cout<<i<<"th smallest element from rand_select is: "<<rand_select_op<<"\n";

  /**
   * Call for finding ith smallest (i+1th since we start from 0)
   * Call for finding ith smallest number in worst case linear time
   */
```

```
  int linear_select_op = linear_select(input, 0, 99, i+1);
  cout<<i<<"th smallest element from linear_time_select is: "<<linear_select_op<<"\n";
}
```

1.2 Output

```
Input i for finding the ith smallest element from the random partition: 24
Finding ith smallest number from the array using random select...
24th smallest element from rand_select is: 24
24th smallest element from linear_time_select is: 24
```

2 Qn2. Longest Common Subsequence

2.1 Code

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <time.h>
#include <iomanip>
#include <math.h>
#include <string>

using namespace std;

void printLCS(vector<vector<char>> _direction, string _seq1, int i, int j){

  if(i == 0 || j == 0){
    return;
  }

  if(_direction[i][j] == '@'){        //diagonal arrow, match sequence
    printLCS(_direction, _seq1, i - 1, j - 1);
    cout << _seq1[i];
  }

  else if(_direction[i][j] == '^'){  //up arrow
    printLCS(_direction, _seq1, i - 1, j);
  }

  else{         //left arrow
    printLCS(_direction, _seq1, i, j - 1);
  }

}


int longest_common_subsequence(string seq1, string seq2){
```

```
  int m = seq1.length();
  int n = seq2.length();

  //B matrix - modeled as vector of vector B[m+1][n+1];
  vector<vector<char>> direction(m+1, vector<char>(n+1, 0));

  //C matrix - modeled as vector of vector  C[m+1][n+1];
  vector<vector<char>> magnitude(m+1, vector<char>(n+1, 0));

  for(int i = 0; i <= m; i++){

    for(int j = 0; j <= n; j++){

      if(i == 0 || j == 0) {        //default fill
        magnitude[i][j] = 0;
        direction[i][j] = '/';
      }

      else if(seq1[i] == seq2[j]){
        magnitude[i][j] = magnitude[i-1][j-1] + 1;
        direction[i][j] = '@'; //using @ to represent diagonal(up/left) arrow.
      }

      else if(magnitude[i-1][j] >= magnitude[i][j-1]){
        magnitude[i][j] = magnitude[i-1][j];
        direction[i][j] = '^'; //using # to represent the up arrow.
      }

      else{
        magnitude[i][j] = magnitude[i][j-1];
        direction[i][j] = '<'; //using ! to represent the left arrow.
      }
    }
  }

  printLCS(direction, seq1, seq1.length(), seq2.length());

  return magnitude[m][n];
}



int main(){

  string sequence1;
  string sequence2;
```

```
  cout << "Enter sequence 1: ";
  cin >> sequence1;

  cout << "Enter sequence 2: ";
  cin >> sequence2;
  // sequence1 = "abcdeeffdd";
  // sequence2 = "ded";

  string space = " ";
  sequence1.insert(0, space);
  sequence2.insert(0, space);

  cout<<"\n";
  int sequenceLength = longest_common_subsequence(sequence1, sequence2);
  cout << endl << "Length of longest subsequence: "<<sequenceLength - 1 << endl;
}
```

2.2 Output

```
Enter sequence 1: abcbdab
Enter sequence 2: bdcaba

bcba
Length of longest subsequence: 4
```