FIT2004: Lab questions for week 5

Objectives: This prac provides a platform for you to learn the formal concepts introduced during lectures in weeks 3& 4. Primarily, these concepts include partitioning problems, recursive sorting, divide and conquer and dynamic programming strategies. NOTE: This prac is **NOT** assessed. These programs are to be written in python programming language.

1. Write a program implementing the basic Edit Distance algorithm from the lecture in Week 4. Your program should accept 2 strings as input and produce as output their edit distance and **alignment**.[1]

2. Given a sequence of numbers, the longest increasing subsequence problem is to find the longest subsequence in the list which is sorted in ascending order. For example, if a list is $\{0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15\}$ then the longest increasing subsequence is $\{0, 2, 6, 9, 11, 15\}$ because there is no subsequence in this list that is in ascending order and is longer than this. Write a dynamic programming algorithm to find the length of the longest increasing subsequence. E.g., the answer for the above example is 6.

3. Suppose you have N jobs where each job has a start time, a finish time and a profit value associated with it. A valid subset of the jobs is the subset of jobs such that no two jobs overlap. Your goal is to find the valid subset of jobs with maximum profit value. Below is an example.

   Suppose you have following jobs where the first two values in each job represent start and finish time and the third value represents the profit of the job.

   Job1 = [1, 10, 50]

   Job2 = [8, 12, 100]

   Job3 = [10, 45, 100]

   Job4 = [18, 55, 150]

   Job5 = [50, 60, 50]

   The optimal schedule is Job2 and Job4 with total profit of 250. Note that $Job1, Job3, Job5$ is a valid schedule but its total profit is 200. Your goal is to write a dynamic program to find the total profit of the optimal schedule.

4. Modify the above program to print the jobs in the optimal schedule using backtracking.

[1] An alignment between two string is derived by backtracking from the sink to the source of the dynamic programmig history matrix.

5. Implement an array-based heap data structure with `upHeap` and `downHeap` operations. Use this to write a Heap Sort program and, as always, test its behaviour and performance carefully.

6. Use the implementation of downHeap and implement the Heapify function with $O(N)$ time complexity as explained in the lecture slides.

```
-=o0o=-
  END
-=o0o=-
```