CISC2006: Algorithm Design and Analysis
Homework 3 **Due Date:** 23:59, 29 March 2022

Rules: Discussion of the problems is permitted, but writing the assignment together is not (i.e. you are not allowed to see the actual pages of another student). You can get at most 100 points if attempting all problems. Please make your answers precise and concise.

1. (20 pts) Complete the tables for dynamic programming for the Longest Common Sub-sequence (LCS) Problem on $X = CGATAC$ and $Y = ACGCTAC$.

   Let $c(i,j)$ represent the length of LCS(X_i, Y_j), where $X_i = (x_1, x_2, \ldots, x_i)$ contains the first i letters of X and $Y_j = (y_1, y_2, \ldots, y_j)$ contains the first j letters of Y.

   Fill in the missing values of the entries.

	•	A	C	G	C	T	A	C
•	0	0	0	0	0	0	0	0
C	0	0	1	1	1	1	1	1
G	0	0	1	2	2	2	2	2
A	0	1	1	2	2	2	3	3
T	0	1	1	2	2	3	3	3
A	0	1	1	2	2	3	4	4
C	0	1	2	2	3	3	4	5

   In the following, each entry $b(i,j) \in \{\uparrow, \nwarrow, \leftarrow\}$ represents the direction of recursion for $c(i,j)$. Specifically,$b(i,j) = \nwarrow$ if $x_i = y_j$; otherwise, $b(i,j) = \uparrow$ if $c(i-1,j) \geq c(i,j-1)$; $b(i,j) = \leftarrow$ if $c(i-1,j) < c(i,j-1)$.

   Fill in the missing values of the entries.

	•	A	C	G	C	T	A	C
•	0	0	0	0	0	0	0	0
C	0	↑	↖	←	↖	←	←	↖
G	0	↑	↑	↖	←	←	←	←
A	0	↖	↑	↑	↑	↑	↖	←
T	0	↑	↑	↑	↑	↖	↑	↑
A	0	↖	↑	↑	↑	↑	↖	←
C	0	↑	↖	↑	↖	↑	↑	↖

   What is the LCS of X and Y? **Solution:** CGTAC

2. (20 pts) Complete the tables for dynamic programming for the Knapsack Problem with $C = 12$, on items $\{(0.74, 6), (0.84, 5), (0.37, 4), (0.57, 3), (0.25, 2), (0.10, 1)\}$, where each item i is represented by (value v_i, size s_i).

   Let $f(i, b)$ represent the optimal objective on items $\{1, 2, \ldots, i\}$ and with capacity b. We have the following recursion for computing the value of $f(i, b)$ (suppose $b \geq s_i$):

$$f(i, b) = \max\{f(i-1, \underline{b - s_i}) + \underline{v_i}, f(i-1, \underline{b})\}$$

   Fill in the missing values of the entries.

f	1	2	3	4	5	6	7	8	9	10	11	12
1 (0.74, 6)	0.00	0.00	0.00	0.00	0.00	0.74	0.74	0.74	0.74	0.74	0.74	0.74
2 (0.84, 5)	0.00	0.00	0.00	0.00	0.84	0.84	0.84	0.84	0.84	0.84	1.58	1.58
3 (0.37, 4)	0.00	0.00	0.00	0.37	0.84	0.84	0.84	0.84	1.21	1.21	1.58	1.58
4 (0.57, 3)	0.00	0.00	0.57	0.57	0.84	0.84	0.94	1.41	1.41	1.41	1.58	1.78
5 (0.25, 2)	0.00	0.25	0.57	0.57	0.84	0.84	1.09	1.41	1.41	1.66	1.66	1.78
6 (0.10, 1)	0.10	0.25	0.57	0.67	0.84	0.94	1.09	1.41	1.51	1.66	1.76	1.78

   In the following, each entry $d(i, b) \in \{Y, N\}$ represents the direction of recursion for $f(i, b)$. Specifically, $d(i, b) = Y$ if $f(i, b)$ is obtained by taking item i into the knapsack. Fill in the missing values of the entries.

d	1	2	3	4	5	6	7	8	9	10	11	12
1 (0.74, 6)	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y
2 (0.84, 5)	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y
3 (0.37, 4)	N	N	N	Y	N	N	N	N	Y	Y	N	N
4 (0.57, 3)	N	N	Y	Y	N	N	Y	Y	Y	Y	N	Y
5 (0.25, 2)	N	Y	N	N	N	N	Y	N	N	Y	Y	N
6 (0.10, 1)	Y	N	N	Y	N	Y	N	N	Y	N	Y	N

   What is the optimal solution (set of items) of the problem instance?

   Solution: items $\{2, 3, 4\}$, with total size 12 and total value 1.78.

   What is the solution returned by the Greedy algorithm that

   - picks items with maximum value first: **Solution:** items $\{2, 1, 6\}$, with total size 12 and total value 1.68.
   - picks items with minimum size first: **Solution:** items $\{6, 5, 4, 3\}$, with total size 10 and total value 1.29.
   - picks items with maximum density first: **Solution:** items $\{4, 2, 5, 6\}$, with total size 12 and total value 1.76.

3. (10 pts) Given n intervals $\{I_1, I_2, \dots, I_n\}$, where each interval $I_i = (s_i, f_i)$ has start time s_i and finish time f_i, both of which are integers and $f_i > s_i > 0$. The intervals are sorted in ascending order of their finish times, i.e., we have $f_1 \le f_2 \le \dots \le f_n$. Additionally, each interval is associated with a value $v_i \ge 0$. The objective of the problem is to pick a subset of non-overlapping intervals with maximum total value.

- (5 pts) For each interval i, let $k(i) \le i - 1$ be the maximum index such that $f_{k(i)} \le s_i$. If $f_1 > s_i$ then we define $k(i) = 0$.

  Show that we can compute $k(1), k(2), \dots, k(n)$ in $O(n \log n)$ total time.

  Solution: Since the finish times are sorted from minimum to maximum, for each i we can use binary search to find the maximum f_j satisfying $f_j \le s_i$ in $O(\log n)$ time. Then we set $k(i) \leftarrow j$. Hence we can compute $k(1), k(2), \dots, k(n)$ in $O(n \log n)$ total time.

- (5 pts) Let $A[i]$ be the value of the optimal solution of the problem with intervals $\{1, 2, \dots, i\}$, where $i \in \{0, 1, \dots, n\}$. In other words, $A[i]$ is the maximum total value of a subset of non-overlapping intervals in $\{I_1, \dots, I_i\}$. Derive a recursive formula for $A[i]$, e.g., express $A[i]$ using $A[1], \dots, A[i-1]$ and v_i, and use it to give an $O(n)$ time algorithm to compute the value of the optimal solution.

  You can assume that you already have the values $k(1), k(2), \dots, k(n)$.

  Solution: Initialize $A[0] = 0$. If interval i does not appear in the optimal solution that gives value $A[i]$, then we have $A[i] = A[i-1]$; if interval i appears in the optimal solution that gives value $A[i]$, then we have $A[i] = A[k(i)] + v_i$, because all other intervals in the solution must finish before s_i. Hence we have

  $$A[i] = \max\{A[i-1], A[k(i)] + v_i\}.$$

  The value of the optimal solution is given by $A[n]$, and can be computed in linear time by computing $A[i]$ in the order of $i = 1, \dots, n$.

4. (10 pts) Compute a table representing the failure function in the Knuth-Morris-Pratt (KMP) algorithm, for the pattern string "CGTCGCGTCGTAC".

   Recall that given a string P with m characters, the failure function f is defined as follows: for all $i \in \{0, 1, \ldots, m-1\}$, $f(i) \in \{0, 1 \ldots, i\}$ is the length of the longest prefix (excluding $P[0, i]$ itself) of $P[0, i]$ that is also a suffix of $P[0, i]$.

   Solution:

$P[i]$	C	G	T	C	G	C	G	T	C	G	T	A	C
i	0	1	2	3	4	5	6	7	8	9	10	11	12
$f(i)$	0	0	0	1	2	1	2	3	4	5	3	0	1

5. (10 pts) Let T be a text of length n, and let P be a pattern of length m. Design an algorithm for finding the longest prefix of P that is a sub-string of T and analyze its running time. You get 5 pts if your algorithm runs in $O(nm)$ time; 8 pts if it runs in $O(n \log m)$ time; 10 pts if it runs in $O(n+m)$ time.

Solution:

Using the KMP algorithm we can check whether P is a sub-string of T in $O(n+m)$ time. Since P has $O(m)$ prefixes, by enumerating all of them and run the KMP algorithm on each of them (with T), we get an $O(nm)$ time algorithm. We can improve this algorithm by a binary search on the length of prefix of P to identify the longest prefix of P that is a sub-string of T in $O(n \log m)$ time.

To get an $O(n+m)$ algorithm, we modify the KMP as follows (refer to the pseudo-code of KMP algorithm from the lecture notes).

As in KMP, we first compute the failure function and initialize $i \leftarrow 0$ and $j \leftarrow 0$. Then in each while loop, if we have $T[i] = P[j]$, we record j. Note that this happens only if $P[0,j]$ is a sub-string of T. In other words, we maintain a variable l (which is initialized to 0) and update $l \leftarrow \max\{l, j\}$ whenever $T[i] = P[j]$. The other parts of the algorithms are the same. When the while loop breaks, we output $P[0,l]$, which is longest prefix of P that is a sub-string of T.

Since the algorithm is identical to KMP except that we record the maximum of j throughout the algorithm (which takes $O(n)$ time because every time we update l, i increases by 1), the running time is still $O(n+m)$.

6. (30 pts) **Dynamic Programming Algorithm for the Knapsack Problem**.

   In this problem you need to implement the dynamic programming algorithm for the Knapsack problem.

   In the main function, your algorithm need to read an input instance of the Knapsack problem from a file (each item has an ID i, a value v_i and size s_i). Then a function is called to compute a solution for the instance, and output the solution (the set of items) and its objective value (total value of items in the solution).

   You need to implement each of the following 4 algorithms as a function.

   - Algorithm that greedily picks items with maximum value first.
   - Algorithm that greedily picks items with minimum size first.
   - Algorithm that greedily picks items with maximum density first.
   - The Dynamic Programming algorithm the computes the optimal solution.

   Several test cases of input will be provided.

7. (20 pts) **Implementing the Knuth-Morris-Pratt (KMP) Algorithm**.

   In this problem you need to implement the KMP algorithm for string matching.

   In the main function, your algorithm need to read a text string T (of length n) and a pattern string P (of length m). Then a function is called to decide whether P is a sub-string of T or not. If it is, then output the position i for which $P = T[i, i+m-1]$; otherwise output "P is not a sub-string of T".

   You need to implement the following 2 algorithms.

   - The $O(nm)$ time Brute-Force algorithm.
   - The $O(n+m)$ time KMP algorithm. In the implementation, you need to first compute the failure function in $O(m)$ time, then use the failure function to solve the problem in $O(n+m)$ time.

   Several test cases of input will be provided. For each test case, the main function calls the two algorithms to solve the problem, and reports the running time of the algorithms.