COL 351 : Analysis & Design of Algorithms

LECTURE 28

Quiz 3

Oct 16, 2024 | Rohit Vaish

Problem 1

Problem 1 [12 points]

Let X and Y be two strings of length n and m, respectively. How many distinct alignments between X and Y are there? Justify your answer.

Note: Your calculation should exclude alignments that match two gaps with each other. Additionally, if $X =$ `AB` and $Y =$ `CD`, then the alignments `(AB_,C_D)` and `(A_B,CD_)` are considered equivalent and should be counted as a single alignment.

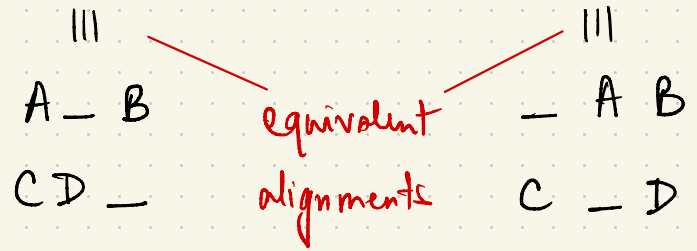
The answer is ${}^{n+m}C_n$.

To show this, let us construct a bijection between alignments and merged strings.

For example, let $X = AB$ and $Y = CD$.

Then, there are six possible alignments given by:

A B	A B _	A B _	A _ B	_ A B	A B _ _
C D	C _ D	_ C D	_ C D	C D _	_ _ C D



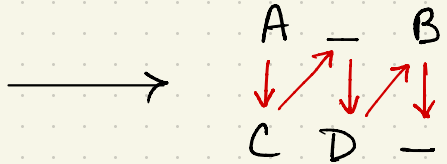
Convention: Among equivalent alignments, we will use those where a character from string Y that is matched to a gap is positioned as far left as possible while maintaining the relative ordering of the characters of both strings.

e.g., between A B _ and A _ B , we will use the
C _ D C D _

the latter alignment because, in the left alignment, "D" (and its match) can be moved left without changing the relative ordering of the characters.

With each alignment (as per our convention), we can associate a unique merged string as follows:

A _ B
C D _



A C D B (ignoring gaps)

Conversely, with any merged string, we can associate a unique alignment (as per our convention) as follows:

A C D B ⟶ A C D B ⟶

A _ B
C D _

Any adjacent pair of characters (X followed by Y) must be a match as per our convention

Thus, no. of distinct alignments = no. of merged strings

= no. of ways of picking n ordered slots out of (m+n) slots

= $^{n+m}C_n$.

Problem 2

Problem 2 [12 points]

Given an integer n and nonnegative numbers $p_1, \ldots, p_n \in [0, 1]$, you want to determine the probability of obtaining strictly more heads than tails when n biased coins are tossed independently at random, where p_i is the probability that the i^{th} coin comes up heads. Give an $\mathcal{O}(n^2)$ algorithm for this task. Assume you can multiply and add two numbers in $[0, 1]$ in $\mathcal{O}(1)$ time. Justify the correctness and running time of your algorithm.

We will design a dynamic programming algorithm.

For any $i \in \{0, 1, \ldots, n\}$ and $j \in \{1, 2, \ldots, n\}$, let $T_{i,j}$ denote the probability of obtaining exactly i heads from the first j coins (assuming $i \leq j$).

Then, the desired answer is given by $\sum_{i \in \lfloor \frac{n}{2} \rfloor + 1}^{n} T_{i,n}$.

Recurrence : For $i \in \{0, 1, \ldots, n\}$ and $j \in \{1, 2, \ldots, n\}$

$$T_{i,j} := \begin{cases} (1-p_1)(1-p_2)\cdots(1-p_j) & \text{if } i = 0 \\ 0 & \text{if } i > j \\ p_1 & \text{if } i = 1 \text{ and } j = 1 \\ p_j \cdot T_{i-1,j-1} + (1-p_j) \cdot T_{i,j-1} & \text{otherwise} \end{cases}$$

- p_j : heads for j^{th} coin
- $T_{i-1,j-1}$: $(i-1)$ heads from previous $(j-1)$ coins
- $(1-p_j)$: tails for j^{th} coin
- $T_{i,j-1}$: i heads from previous $(j-1)$ coins

Algorithm :

input : n nonnegative numbers $p_1, p_2, \cdots, p_n$

output : a nonnegative number p // probability of strictly more heads than tails

* initialize 2D array T of size $(n+1) \times n$

// base cases

* $T(0, j) = \prod_{k \le j} (1 - p_k)$ for all $j \in \{1, 2, \cdots, n\}$
* $T(1, 1) = p_1$
* $T(i, j) = 0$ for all $i > j$

* for all $j \in \{2, 3, \dots n\}$

  for all $i \in \{1, \dots, j\}$

  $T(i,j) = p_j \cdot T(i-1, j-1) + (1-p_j) \cdot T(i, j-1)$

* Return $\sum_{i \in \lfloor \frac{n}{2} \rfloor + 1}^{n} T_{i,n}$

Correctness : Follows from induction and law of conditional probability

Running time: Base case $T(0,j)$ takes $O(n^2)$ time since we are given that multiplication is $O(1)$.

T is of size $O(n^2)$.

Computing $T(i,j)$ inside the nested for-loop takes $O(1)$ time per entry.

Thus, the algorithm takes $O(n^2)$ time overall.