CS F364
Design & Analysis of Algorithms

Algorithm Design Techniques

Dynamic Programming : String / Text Problems: Examples

- Sequence Alignment

PROBLEM – SEQUENCE ALIGNMENT

- Consider the following DNA sequences:
  - GACGGATTAG and GATCGGAATAG
- They are very similar:
  - GA?CGGATTAG

  - GATCGGAATAG
    - ? denotes a missing element
- In this example,
  - The two sequences differ in 2 positions
- Problem:
  - Given two sequences determine the best alignment (i.e. the alignment with minimal difference)
    - Sub-problem: Compute a similarity score

PROBLEM – SEQUENCE ALIGNMENT

- What is an alignment?
  - An alignment is defined as the insertion of spaces in arbitrary locations in either sequence,
    - so that they end up with the same size
    - but no space in one sequence should align with a space in the other
- Given an alignment a similarity score can be assigned:
  - Each column receives a certain value:
    - Identical characters: **a** (match)
    - Different characters: **b** (mismatch)
    - (One) Space in the column: **c**

    where **a**, **b**, and **c** are constant values chosen by domain expert.
  - Then the **similarity score** is the *sum of values of all columns*

Problem – Sequence Alignment

- Given sequences s and t, we determine alignment scores between arbitrary prefixes:
  - i.e. between s[1..i] and t[1..j]
- Aligning between s[1..i] and t[1..j] requires one of the following :
  1. Align s[1..i] with t[1..j-1] and match a space with t[j]
  2. Align s[1..i-1] with t[1..j-1] and match s[i] with t[j]
  3. Align s[1..i-1] with t[1..j] and match s[i] with a space

- Exercise:
  - Argue that these cases are exhaustive: i.e.
    - *one need not consider any other case.*

Problem – Sequence Alignment

- Given sequences s and t, we determine *alignment scores* between arbitrary prefixes:
  - i.e. between s[1..i] and t[1..j]
- So, the recurrence for *similarity score* is:

```
sim(s[1..i],t[1..j])
   = max  {
      sim(s[1..i],t[1..j-1]) + c,
      sim(s[1..i-1],t[1..j-1]  + ((s[i]==t[j]) ? a : b),
      sim(s[1..i-1],t[1..j]) + c ,
       }    if i>=1 and j>=1
  = 0 otherwise
```

PROBLEM – SEQUENCE ALIGNMENT

- Exercise: Write the DP algorithm for computing the similarity score (based on the recurrence given in the previous slide).

- Time Complexity: $\Theta(|s|*|t|)$
- Space Complexity: $\Theta(|s|*|t|)$
  - Can be pruned to $\Theta(|t|)$ or $\Theta(|s|)$ depending on the order of computations.

- Question: When is pruning acceptable?
  - i.e. when do you need to recover the space insertions or the space-inserted strings?