CS F364
Design & Analysis of Algorithms

Algorithm Design Techniques

Dynamic Programming : String / Text Problems: Examples

- Sequence Alignment

Problem – DNA Sequence Alignment

- Consider the following (sample) 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 – DNA Sequence Alignment - Scoring

- 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 a sequence is aligned 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 experts.
  - Then the **similarity score** is the *sum of values of all columns*

Problem – Sequence Alignment

- Given sequences **s** and **t**, we determine similarity 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 :
  i. Align **s[1..i]** with **t[1..j-1]** and match a space with **t[j]**
  ii. Align **s[1..i-1]** with **t[1..j-1]** and match **s[i]** with **t[j]**
  iii. 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 define *similarity scores* between prefixes **s[1..i]** and **t[1..j]**
- using the following recurrence :

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

= d otherwise

Note: **a**,**b**,**c**, and **d** are constants chosen by domain experts. End of Note.

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 it be pruned?

    [Hint: *Inspect the induction(s).* End of Hint.]

- Question: When is pruning acceptable? [**Hint:** *Do you need to recover the space-inserted strings?* **End of Hint.**]