# CS F364 Design & Analysis of Algorithms

# **ALGORITHM DESIGN TECHNIQUES**

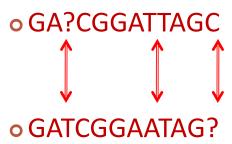
**Dynamic Programming : String / Text Problems: Examples** 

- Approximate String Matching: Edit Distance



# PROBLEM — COMPUTING EDIT DISTANCE

- Consider the problem of <u>finding occurrences of a given</u> word w in a given text T:
  - this would require matching of strings
    - o i.e. matching w with a text window T[i..j]
- Consider a word processor where the text is being edited and spelling errors are likely:
  - i.e. matching has to be approximate
- In such a context it is better for matching to compute a score – referred to as the *edit distance* – that indicates (dis)similarity:
  - for example the following strings would have an edit distance of 3:



### PROBLEM — COMPUTING EDIT DISTANCE

- Problem Definition:
  - Given strings S and T:
    - o compute the number of <u>insertions</u>, <u>deletions</u>, or <u>substitutions</u> (of characters)
    - orequired on S so that it becomes T.

# PROBLEM - COMPUTING EDIT DISTANCE

```
• Recurrence on S[1..i], T[1..j]
• edit(s[1..i],t[1..j])
= min {
    edit(s[1..i],t[1..j-1]) + 1,
    edit(s[1..i-1],t[1..j-1] + ((s[i]==t[j]) ? 0 : 1),
    edit(s[1..i-1],t[1..j]) + 1,
    } if i>=1 and j>=1
= 0 otherwise
```

#### Exercise:

Compare / Contrast this recurrence with the corresponding one for Sequence Alignment