Edit Distance

CS 491 CAP

Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Spring, 2023

Objectives

- Use DP to determine the edit distance between two strings.

The Problem

- Given two strings s and t, how many "edits" does it take to tranform one to another?
  - Edit = insert, delete, or change.
  - Usually each of these "costs" one unit.
- Usually called the Levinstein Distance
- Examples:
  - changing `DATA` to `BETA` needs 2 steps.
  - changing `ETA` to `BETA` needs 1 step.
  - changing `GRETA` to `BETA` needs 2 steps.

The Näive Algorithm

Base Cases

```
1  // Thanks, Wikipedia!
2  int LD(string s, int len_s, string t, int len_t) {
3    int cost;
4
5    /* base case: empty strings */
6    if (len_s == 0) return len_t;
7    if (len_t == 0) return len_s;
8
9    /* test if last characters of the strings match */
10   if (s[len_s-1] == t[len_t-1])
11       cost = 0;
12   else
13       cost = 1;
```

The Näive Algorithm, ctd

Recursive Case

```
15    /* return minimum of delete char from s,
16                          delete char from t,
17                      and delete char from both */
18    return minimum(LD(s, len_s - 1, t, len_t    ) + 1,
19                   LD(s, len_s    , t, len_t - 1) + 1,
20                   LD(s, len_s - 1, t, len_t - 1) + cost);
21  }
```

Dynamic Programming using Memoization

Base Cases

```
1  int LD(const char *s, int len_s, const char *t, int len_t)
2  {
3    vvi dp = vvi(len_s + 1, vi(len_t +1));
4    int cost;
5
6    for(int i=0; i<=len_s; ++i)
7       dp[i][0] = i;
8
9    for(int i=0; i<=len_t; ++i)
10      dp[0][i] = i;
```

Dynamic Programming using Memoization, ctd

Memoized Part

```
11    for(int i=1; i<=len_s; ++i)
12      for(j=1; j<=len_t; ++j) {
13          cost = s[i] == t[j] ? 0 : 1;
14
15          dp[i,j] = minimum(dp[i-1][j] + 1,
16                            dp[i][j-1] + 1,
17                            dp[i-1][j-1] + cost);
18          }
19     return dp[len_s][len_t];
20  }
```