

Dynamic Programming | Set 5 (Edit Distance)

Continuing further on dynamic programming series, *edit distance* is an interesting algorithm.

Problem: Given two strings of size m , n and set of operations replace (R), insert (I) and delete (D) all at equal cost. Find minimum number of edits (operations) required to convert one string into another.

Identifying Recursive Methods:

What will be sub-problem in this case? Consider finding edit distance of part of the strings, say small prefix. Let us denote them as $[1...i]$ and $[1...j]$ for some $1 < i < m$ and $1 < j < n$. Clearly it is solving smaller instance of final problem, denote it as $E(i, j)$. Our goal is finding $E(m, n)$ and minimizing the cost.

In the prefix, we can right align the strings in three ways $(i, -)$, $(-, j)$ and (i, j) . The hyphen symbol $(-)$ representing no character. An example can make it more clear.

Given strings SUNDAY and SATURDAY. We want to convert SUNDAY into SATURDAY with minimum edits. Let us pick $i = 2$ and $j = 4$ i.e. prefix strings are SUN and SATU respectively (assume the strings indices start at 1). The right most characters can be aligned in three different ways.

Case 1: Align characters U and U. They are equal, no edit is required. We still left with the problem of $i = 1$ and $j = 3$, $E(i-1, j-1)$.

Case 2: Align right character from first string and no character from second string. We need a deletion (D) here. We still left with problem of $i = 1$ and $j = 4$, $E(i-1, j)$.

Case 3: Align right character from second string and no character from first string. We need an

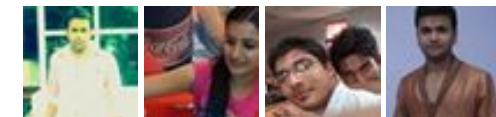
Google™ Custom Search



GeeksforGeeks



53,524 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

insertion (I) here. We still left with problem of $i = 2$ and $j = 3$, $E(i, j-1)$.

Combining all the subproblems minimum cost of aligning prefix strings ending at i and j given by

$$E(i, j) = \min([E(i-1, j) + D], [E(i, j-1) + I], [E(i-1, j-1) + R \text{ if } i, j \text{ characters are not same}])$$

We still not yet done. What will be base case(s)?

When both of the strings are of size 0, the cost is 0. When only one of the string is zero, we need edit operations as that of non-zero length string. Mathematically,

$$E(0, 0) = 0, E(i, 0) = i, E(0, j) = j$$

Now it is easy to complete recursive method. Go through the code for recursive algorithm (edit_distance_recursive).

Dynamic Programming Method:

We can calculate the complexity of recursive expression fairly easily.

$$T(m, n) = T(m-1, n-1) + T(m, n-1) + T(m-1, n) + C$$

The complexity of $T(m, n)$ can be calculated by successive substitution method or solving homogeneous equation of two variables. It will result in an exponential complexity algorithm. It is evident from the recursion tree that it will be solving subproblems again and again. Few strings result in many overlapping subproblems (try the below program with strings *exponential* and *polynomial* and note the delay in recursive method).

We can tabulate the repeating subproblems and look them up when required next time (bottom up). A two dimensional array formed by the strings can keep track of the minimum cost till the current character comparison. The visualization code will help in understanding the construction of matrix.

The time complexity of dynamic programming method is $O(mn)$ as we need to construct the table fully. The space complexity is also $O(mn)$. If we need only the cost of edit, we just need $O(\min(m, n))$ space as it is required only to keep track of the current row and previous row.

Usually the costs D , I and R are not same. In such case the problem can be represented as an acyclic directed graph (DAG) with weights on each edge, and finding shortest path gives edit



Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and Data Packing](#)

[Intersection point of two Linked Lists](#)

[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

distance.

Applications:

There are many practical applications of edit distance algorithm, refer [Lucene](#) API for sample. Another example, display all the words in a dictionary that are near proximity to a given word\incorrectly spelled word.

```
// Dynamic Programming implementation of edit distance
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

// Change these strings to test the program
#define STRING_X "SUNDAY"
#define STRING_Y "SATURDAY"

#define SENTINEL (-1)
#define EDIT_COST (1)

inline
int min(int a, int b) {
    return a < b ? a : b;
}

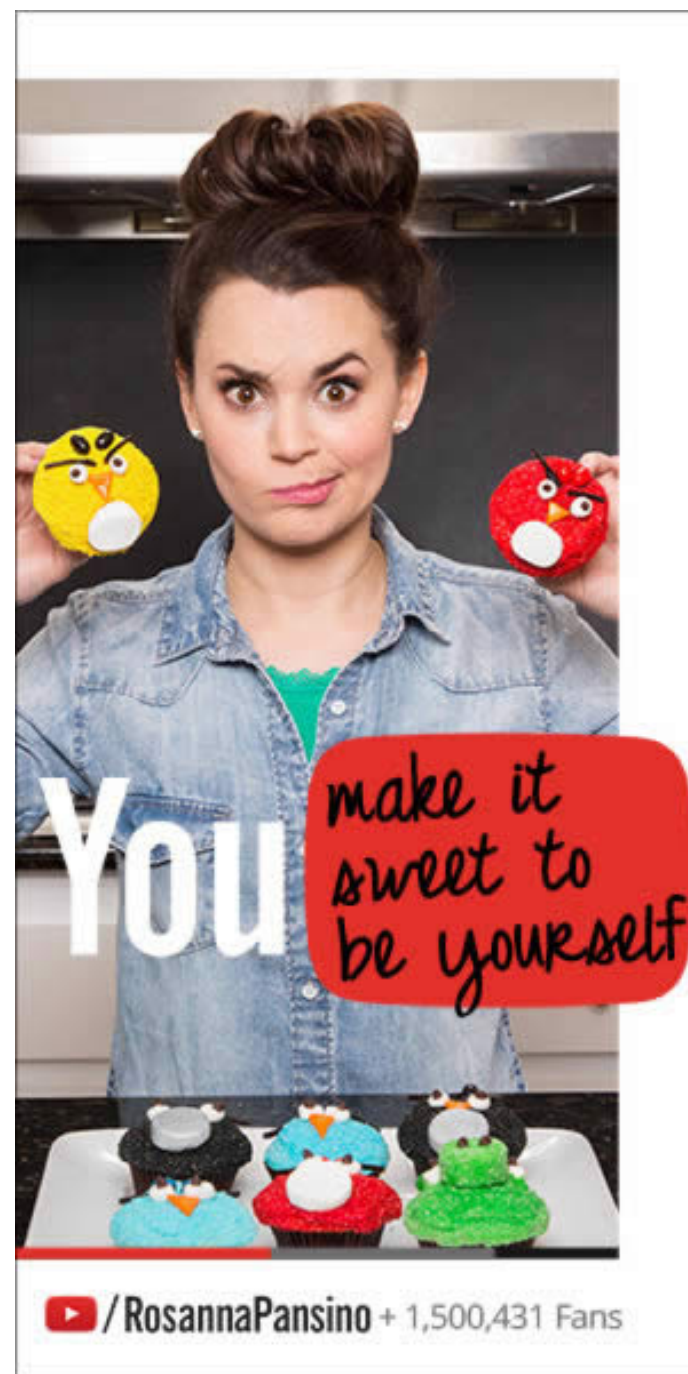
// Returns Minimum among a, b, c
int Minimum(int a, int b, int c)
{
    return min(min(a, b), c);
}

// Strings of size m and n are passed.
// Construct the Table for X[0...m, m+1], Y[0...n, n+1]
int EditDistanceDP(char X[], char Y[])
{
    // Cost of alignment
    int cost = 0;
    int leftCell, topCell, cornerCell;

    int m = strlen(X)+1;
    int n = strlen(Y)+1;

    // T[m][n]
    int *T = (int *)malloc(m * n * sizeof(int));

    // Initialize table
```



Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 38 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 42 minutes ago

Sanjay Agarwal bool

tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...


Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it..

Find subarray with given sum · 1 hour ago

AdChoices 

[► C++ Data String](#)

[► String Java](#)

[► C String](#)

```
for(int i = 0; i < m; i++)
    for(int j = 0; j < n; j++)
        *(T + i * n + j) = SENTINEL;

// Set up base cases
// T[i][0] = i
for(int i = 0; i < m; i++)
    *(T + i * n) = i;

// T[0][j] = j
for(int j = 0; j < n; j++)
    *(T + j) = j;

// Build the T in top-down fashion
for(int i = 1; i < m; i++)
{
    for(int j = 1; j < n; j++)
    {
        // T[i][j-1]
        leftCell = *(T + i*n + j-1);
        leftCell += EDIT_COST; // deletion

        // T[i-1][j]
        topCell = *(T + (i-1)*n + j);
        topCell += EDIT_COST; // insertion

        // Top-left (corner) cell
        // T[i-1][j-1]
        cornerCell = *(T + (i-1)*n + (j-1) );

        // edit[(i-1), (j-1)] = 0 if X[i] == Y[j], 1 otherwise
        cornerCell += (X[i-1] != Y[j-1]); // may be replace

        // Minimum cost of current cell
        // Fill in the next cell T[i][j]
        *(T + (i)*n + (j)) = Minimum(leftCell, topCell, cornerCell)
    }
}

// Cost is in the cell T[m][n]
cost = *(T + m*n - 1);
free(T);
return cost;
}

// Recursive implementation
int EditDistanceRecursion( char *X, char *Y, int m, int n )
```

```

{
    // Base cases
    if( m == 0 && n == 0 )
        return 0;

    if( m == 0 )
        return n;

    if( n == 0 )
        return m;

    // Recurse
    int left = EditDistanceRecursion(X, Y, m-1, n) + 1;
    int right = EditDistanceRecursion(X, Y, m, n-1) + 1;
    int corner = EditDistanceRecursion(X, Y, m-1, n-1) + (X[m-1] != Y[n-1]);

    return Minimum(left, right, corner);
}

int main()
{
    char X[] = STRING_X; // vertical
    char Y[] = STRING_Y; // horizontal

    printf("Minimum edits required to convert %s into %s is %d\n",
           X, Y, EditDistanceDP(X, Y) );
    printf("Minimum edits required to convert %s into %s is %d by recursion\n",
           X, Y, EditDistanceRecursion(X, Y, strlen(X), strlen(Y)));

    return 0;
}


```

AdChoices 

[► Int](#)

[► Convert to Java](#)

[► C# String to Int](#)

AdChoices 

[► C++ Length String](#)

[► C++ Substr String](#)

[► String Strings](#)

— **Venki**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

HIGH-PERFORMANCE COMPUTING ON A UNIVERSITY BUDGET

Define your ideal server

Download
the infographic

SERVERSDIRECT

Related Topics:

- [Backtracking | Set 8 \(Solving Cryptarithmic Puzzles\)](#)
- [Tail Recursion](#)
- [Find if two rectangles overlap](#)
- [Analysis of Algorithm | Set 4 \(Solving Recurrences\)](#)
- [Print all possible paths from top left to bottom right of a mXn matrix](#)
- [Generate all unique partitions of an integer](#)
- [Russian Peasant Multiplication](#)
- [Closest Pair of Points | O\(nlogn\) Implementation](#)



9



Tweet

1



8

Writing code in comment? Please use ideone.com and share the link here.

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team