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 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 if i,j 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 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
    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;
}
```

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

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Dynamic Programming

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

Nilesh • 23 days ago

I was thinking on this approach. Compare and delete the strings not required and inset the ones required.Is this right?

• Reply • Share ›

Harut • a month ago

Let us pick i = 2 and j = 4 i.e. prefix strings are SUN and SATU respectively

should be

Let us pick i = 2 and j = 4 i.e. prefix strings are SU and SATU respectively

• Reply • Share ›

Om Prakash Gupta • a month ago

Standford university pdf link for better understanding.

https://web.stanford.edu/class...

• Reply • Share ›

manuag Om Prakash Gupta • 18 days ago

Thanks..Really helpful

• Reply • Share ›

Jerry Goyal • a month ago

I found this extremely easy comparatively:

```
int min(int i, int j, int k)
{
    return i < j ? (i < k ? i : k) : (j < k ? j : k);
}

int editDistance(string str1, string str2)
{
    int m = str1.length();
    int n = str2.length();
    int **E = new int*[m + 1];
    for(int i = 0; i <= m; ++i)
    {
            E[i] = new int[n + 1];
            E[i][0] = i;
    }
```

see more

• Reply • Share ›

Mission Peace • 2 months ago

https://www.youtube.com/watch?... Check out my video on this question.

1 • Reply • Share ›

gratitude • 2 months ago

in the recurrence relation e(i-1,j-1) when the i and j characters are same .it is written exactly opposite

• Reply • Share ›

Cracker • 2 months ago

http://algods-cracker.blogspot...

• Reply • Share ›

Aditya Goel • 3 months ago

Typo
"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."

Here i=2, so only SU should be considered as index starts from 1.

• Reply • Share ›

mantri ➦ Aditya Goel • a month ago

Goel, is that you ? :D
-Mantri

• Reply • Share ›

geekyprateek • 3 months ago

```
class Distance{
        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);
                }

        void findDistance(String X, String Y){

                int m=X.length()+1;
```

```
            int m=X.length()+1;
            int n=Y.length()+1;
            int cost = 0;
        int leftCell, topCell, cornerCell;
```

see more

• Reply • Share ›

Guest • 3 months ago

/* Paste your code here */

• Reply • Share ›

nishant • 3 months ago

this solution will not work when needed because of the size of array n*m

use this solution with little change --
it asks for the value for limit of distance ;;
after just removing 2 lines in code the code is very fast and also memory efficient ...

```
#include <iostream>

#include <stdio.h>

#include <string.h>

#include <algorithm>

#include <cmath>

using namespace std;

char s[100004];
```

see more

• Reply • Share ›

jobanpreet • 3 months ago

Can the same algorithm be used if the three operations had different cost?

• Reply • Share ›

jayasurya_j ➚ jobanpreet • 3 months ago

of course not!

• Reply • Share ›

jobanpreet ➚ jayasurya_j • 3 months ago

What i mean by using same algorithm is that we can just add the weight of each

operation instead of adding just 1 while calculating int left, int right etc.

• Reply • Share ›

zhen wang • 4 months ago

Case 3: Align right character from second string and no character from first string. We need an insertion (I) here. We still left with problem of i = 2 and j = 3, E(i, j-1).

I didn't get it. Can you give me a example.

Why should we get E(i, j - 1) after we operate a insert.

• Reply • Share ›

ash567 • 5 months ago

is my observation right that
converting string a to b or b to a will always take the same number of minimum steps

• Reply • Share ›

jayasurya_j ash567 • 3 months ago

yeah absolutely!

• Reply • Share ›

ash567 • 5 months ago

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)

I guess i = 2 means only SU must be taken

• Reply • Share ›

tekmaverick ash567 • 4 months ago

Yes.

1 • Reply • Share ›

Bhumik Thakkar • 5 months ago

What would be the time and space complexity?
I think time complexity : O(len1 * len2) and the same for space.

• Reply • Share ›

Amogh Margoor • 5 months ago

http://ideone.com/YBH8CB

• Reply • Share ›

abcde Amogh Margoor • 5 months ago

hey its giving 0 for book and took

• Reply • Share ›

Amogh Margoor ➦ abcde • 5 months ago

Corrected.... Thanks.

• Reply • Share ›

Tarun Kundhiya • 6 months ago

Let us pick i = 2 and j = 4 i.e. prefix strings are SUN and SATU respectively (assume the strings indices start at 1). shouldnt it be i = 3 and j=4

8 • Reply • Share ›

fortybucks • 6 months ago

is there any special reason you use pointer dereferencing to access the array? the code looks much cleaner if you access it like an array..

7 • Reply • Share ›

srj_michael • 7 months ago

Code in java

http://ideone.com/cNp7At

• Reply • Share ›

BLANK • 7 months ago

how to form the DAG??????? please explain.....

6 • Reply • Share ›

Gaurav Ramesh ➦ BLANK • 4 months ago

Check out Chap. 6 of Algorithms by Das Gupta.
http://cseweb.ucsd.edu/~dasgup...

• Reply • Share ›

Guest • 7 months ago

```
#include<iostream>
#include<string.h>
using namespace std;

int t[1000][1000];

int editDist(char s1[], char s2[], int i, int j, int n1, int n2)
{
if(i >= n1)
return ((n2-j));
if(j >= n2)
return ((n1-i));
```

```
if(t[i][j] == 0)
{
int minC = 1000;
int t1 = editDist(s1, s2, i+1, j+1, n1, n2);
if(t1 < minC)
minC = t1;
```

see more

1 • Reply • Share ›

Guest • 7 months ago

// Solution with DP

```
#include<iostream>
#include<string.h>
using namespace std;

int t[1000][1000];

int editDist(char s1[], char s2[], int i, int j, int n1, int n2)
{
if(i >= n1)
return ((n2-j));
if(j >= n2)
return ((n1-i));
if(t[i][j] == 0)
{
int minC = 1000;
int t1 = editDist(s1, s2, i+1, j+1, n1, n2);
if(t1 < minC)
```

see more

• Reply • Share ›

Abc ➚ Guest • 6 months ago

Ishani/Saurabh,
Could you please explain the logic of the above code? :)

1 • Reply • Share ›

santhoshvai • 8 months ago

A very good detailed explanation is present at https://web.stanford.edu/class...

I have implemented this in python : http://ideone.com/DQbGhf

7 • Reply • Share ›

sukanya • 8 months ago

http://ideone.com/yVsop9

• Reply • Share ›

Jeyanthinathan sukanya • 7 months ago

for SZSS and SZZS, its giving required modification 0.

• Reply • Share ›

Twister • 8 months ago

Can't we just use hashing here for each character and solve the number of edits.i.e. where in hash[i] =1 for non repeating characters(i.e. character not present in either) and
Then just calculate the number of edits by arranging them in non-decreasing order.
1.no of insert =|strlen(string1)-strlen(string2)|;
2.No. of replaces = no. of non matching characters in hash
Similarly for other operations
This would give same result also if each operations are arranges and added in ascending order having differnet costs.
->time O(n)
Another method is also searching.->O(m(logm) m=longest string
PS:Correct me if I am Wrong.

• Reply • Share ›

Siddhanjay Godre Twister • 7 months ago

The hashing method would work provided the ordering of the string need not be retained.Consider the case ,"ccccb" and "cbccc".
The hashing method would give the answer 0 while the correct answer is 2.
You could try this problem here:

http://www.codechef.com/ACMKGP...

• Reply • Share ›

Kim Jong-il • 8 months ago

Typo: **@GeeksforGeeks**
"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."

here i should be equal to 3. Because it has assumed that indices start from 1.

6 • Reply • Share ›

Shrinivas • 8 months ago

"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."

Here, first prefix string should be SU.

3 ▲ | ▼ • Reply • Share ›

Saurabh • 8 months ago

Please refer to this link :

https://secweb.cs.odu.edu/~zei...

Here is the code implementing the above:

http://ideone.com/oyMyGw

2 ▲ | ▼ • Reply • Share ›

guest • 8 months ago

Best Explaination Here:- https://secweb.cs.odu.edu/~zei...

3 ▲ | ▼ • Reply • Share ›

Saurabh ➚ guest • 8 months ago

Thanks for sharing the link :) Nice Explanation.

▲ | ▼ • Reply • Share ›

Ritesh • 9 months ago

The first cell represents the Edit Distance between 2 empty strings, which is 0 .. Hence its value is 0 ..

As you said matching the first characters, it occurs T[1][1] .. Have a closer look and you'll understand ..

▲ | ▼ • Reply • Share ›

Paparao Veeragandham • 9 months ago

```
int RecursiveMethod(char str1[], char str2[], int n , int m)
{
if( n ==0 && m ==0 ) return 0;
if( n ==0) return m;
if(m==0) return n;

if( str1[i-1] ==str2[j-1])
return RecursiveMethod(str1,str2, n-1,m-1);
else
return min(RecurisiveMethod(str1,str2,n-1,m), RecurisiveMethod(str1,str2,n,m-1),
RecursiveMethod(str1,str2, n-1,m-1) ) +1;
}
```

T.c = O(n3)

▲ | ▼ • Reply • Share ›

ms • 9 months ago

cant we just find the lcs between two and subtract it from the longer of two strings?
eg: for SATURDAY & SUNDAY
LCS= 5
therfore ans=8-5=3 (since length of SATURDAY is 8)

1 • Reply • Share ›

santhoshvai ms • 8 months ago

This method indeed works if cost for R,I and D is the same.
But since both LCS and the edit distance is O(nm), i guess they use this method as substitution is cost is higher than insertion and deletion

• Reply • Share ›

santhoshvai ms • 8 months ago

From wikipedia, LCS is related to edit distance - "The edit distance when only insertion and deletion is allowed (no substitution), or when the cost of the substitution is the double of the cost of an insertion or deletion .." http://en.wikipedia.org/wiki/L...

• Reply • Share ›

SHIVAM DIXIT ms • 8 months ago

i dont think its correct....

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

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

Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 1 hour ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

My compile-time solution, which agrees with the...

Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago