**To compute a segmentation of maximum total quality for a string**

**Abstract:**

For a provided string of letters y = y1, y2….yn we have to do the segmentation of y such that at the end we get the string of maximum qualtity for this we are following two appproaches 1. Bute force 2. Dynamic programming. By using Dynamic programming approach we can optimize the time complexity of this segmentaion problem.

1. Problem Statement

As some of you know well, and others of you may be interested to learn, a number of languages (including Chinese and Japanese) are written without spaces between the words. Consequently, software that works with text written in these languages must address the word segmentation problem--inferring likely boundaries between consecutive words in the text. If English were written without spaces, the analogous problem would consist of taking a string like “meetateight" and deciding that the best segmentation is "meet at eight" (and not "me et at eight," or "meet ateight," or any of a huge number of even less plausible alternatives). How could we automate this process?

A simple approach that is at least reasonably effective is to find a segmentation that simply maximizes the cumulative "quality" of its individual constituent words. Thus, suppose you are given a black box that, for any string of letters x = x1x2… xk, will return a number quality(x). This number can be either positive or negative; larger numbers correspond to more plausible English words. (So quality ("me") would be positive, while **quality (**"ght") would be negative.) Given a long string of letters y = y1y2…yn, a segmentation of y is a partition of its letters into contiguous blocks of letters; each block corresponds to a word in the segmentation. The total quality of a segmentation is determined by adding up the qualities of each of its blocks. (So we would get the right answer above provided that quality("meet") + quality("at") +quality(" eight") was greater than the total quality of any other segmentation of the string.)

Give an efficient algorithm that takes a string y and computes a segmentation of maximum total quality. (You can treat a single call to the black box computing quality(x) as a single computational step.)

(A final note, not necessary for solving the problem: To achieve better performance, word segmentation software in practice works with a more complex formulation of the problem-for example, incorporating the notion that solutions should not only be reasonable at the word level, but also form coherent phrases and sentences. If we consider the example "theyouthevent," there are at least three valid ways to segment this into common English words, but one constitutes a much more coherent phrase than the other two. If we think of this in the terminology of formal languages, this broader problem is like searching for a segmentation that also can be parsed well according to a grammar for the underlying language. But even with these additional criteria and constraints, dynamic programming approaches lie at the heart of a number of successful segmentation systems.)

2 The Brute Force approach

In the Brute approach, we consider all the possible combination of segmentation in string for each passible segmentation we are checking it with dictionary if found the quality of that segmented part is 1 else -1 .In this fashion we are doing this procedure for all the possible segmentation of string at the end. We will consider such a combination of segmented string which return the maximum quality.

**2.1 Example & Analysis**

* Input : goto
* Dictionary ={ go , to, too, and}

1. goto (quality = -1)

2. g\_oto (quality = -2)

3. **go\_to (quality = 2)**

4. got\_o (quality = -2)

5. g\_o\_to (quality = -1)

6. g\_ot \_o (quality = -3)

7. go\_t\_o (quality = -1)

8. g\_o\_t\_o (quality = -4)

Time complexity

As we can see in this example there are 2^n-1 possilbles segmentaion of string out of which one will give the maximum quality so in the worst case we need to check approximately 2^n to get the maximum quality . Time complexity is O(2^n)

**3 Dynamic Programming approach**

3.1 Dynamic programming Strategy

As Brute approach takes O (2n) time complexity in worst case so we can optimize it to O (n2\*s) time complexity by using Dynamic programming technique where s is length of dictionary maintaining. In this strategy, we maintaining two matrix 1. Quality Matrix 2. Trace Matrix. We compute the quality of segmented string if it is already computed we take that value if not computed we computed it and store it for future reference. While doing so we maintaining entery in tradce matrix . At the end from the trace matrix we will trace down the maximum quality segmented string.

3.2 Algorithm

1. Calculate quality of each single letter of a string

If letter found in dictionary then quality 1 else -1 and store it in QM

Store position of letter in TM

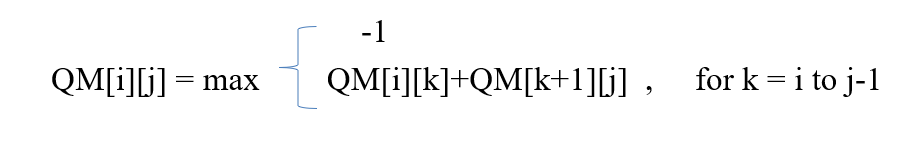
2. Calculate remaining possibilities of string

Take substring from i to j

3. for i<j,

If substring present in dictionary then quality 1

TM[i][j] = j



TM[i][j] = k for which QM[i][j] is max

* **Tracing String from traceMatrix:**

traceString ( TM, str, start, en

**{**

**int split = TM[start][end];**

if(split == end)

{

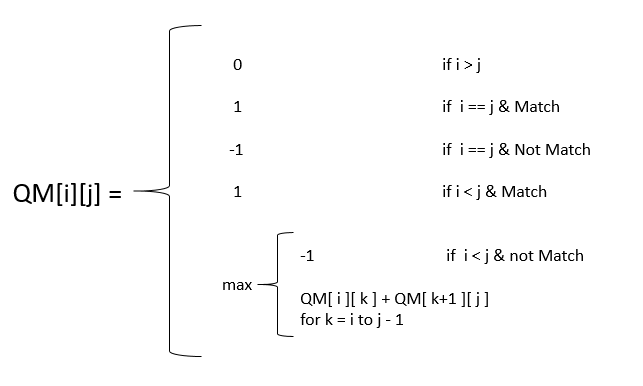
print string from start to end

return;

} traceString(TM, str, start, split); traceString (TM, str, split+1,end); }

3.3 Proof and analysis

Consider a string a stirng “goto”

In dynamic programming approach, first we check quality of each single letter in string and make the entry in the quality matrix and trace matrix. After that we find quality of remaining possible segmentation o string for all this we use the following condition. 

Make entry for ‘j’ in trace matrix if value is max

else

Make entry for ‘k’ in trace matrix if k value is max

By using above conditions we ge matrix

**For “goto” :**

**Quality Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **-1** | **1** | **0** | **2** |
| **0** | **-1** | **-1** | **0** |
| **0** | **0** | **-1** | **1** |
| **0** | **0** | **0** | **-1** |

**Trace matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **1** | **1** | **1** |
| **0** | **1** | **2** | **1** |
| **0** | **0** | **2** | **3** |
| **0** | **0** | **0** | **3** |

**Tracing of trace matrix**

traceString ( TM, str, start, end)

{

int split = TM[start][end];

if(split == end)

{

// print string from start to end

return ;

}

traceString(TM, str, start, split);

traceString (TM, str, split+1,end);

}

By using above algorithm we can get maximum quality string “go to” having quality 2

To calculate this 2 matrix we require n^2 time and for each element we check in dicrionary to get qualtioy (say s= length of dictionary )

Therefore,

**Time complexity = (n^2 \*s)**

**4 Performance evaluation**

|  |  |  |
| --- | --- | --- |
| **Input String** | **Brute Force**  **(time in microsec)** | **Dynamic Programming**  **(time in microsec)** |
| Iam | 3.03 | 3.03 |
| Iamace | 3.28 | 3.28 |
| howareu | 3.33 | 3.33 |
| Meetmeat | 3.42 | 3.42 |

Conclusion

* Brute Force approach tries all possible paths which may take exponential time.
* In Dynamic programming approach, it checks if the subproblem is already computed or not. If computed then it will not compute again.