Two Exercises

张海鹏

江南大学

2017-11

目录

Minimum Path Sum

Given a *m* x *n* grid filled with non-negative numbers, find a path from top left to bottom right which *minimizes* the sum of all numbers along its path.

Note: You can only move either down or right at any point in time.

Example 1:[[1,3,1], [1,5,1],[4,2,1]]

Given the above grid map, return 7. Because the path 1→3→1→1→1 minimizes the sum.

Solution

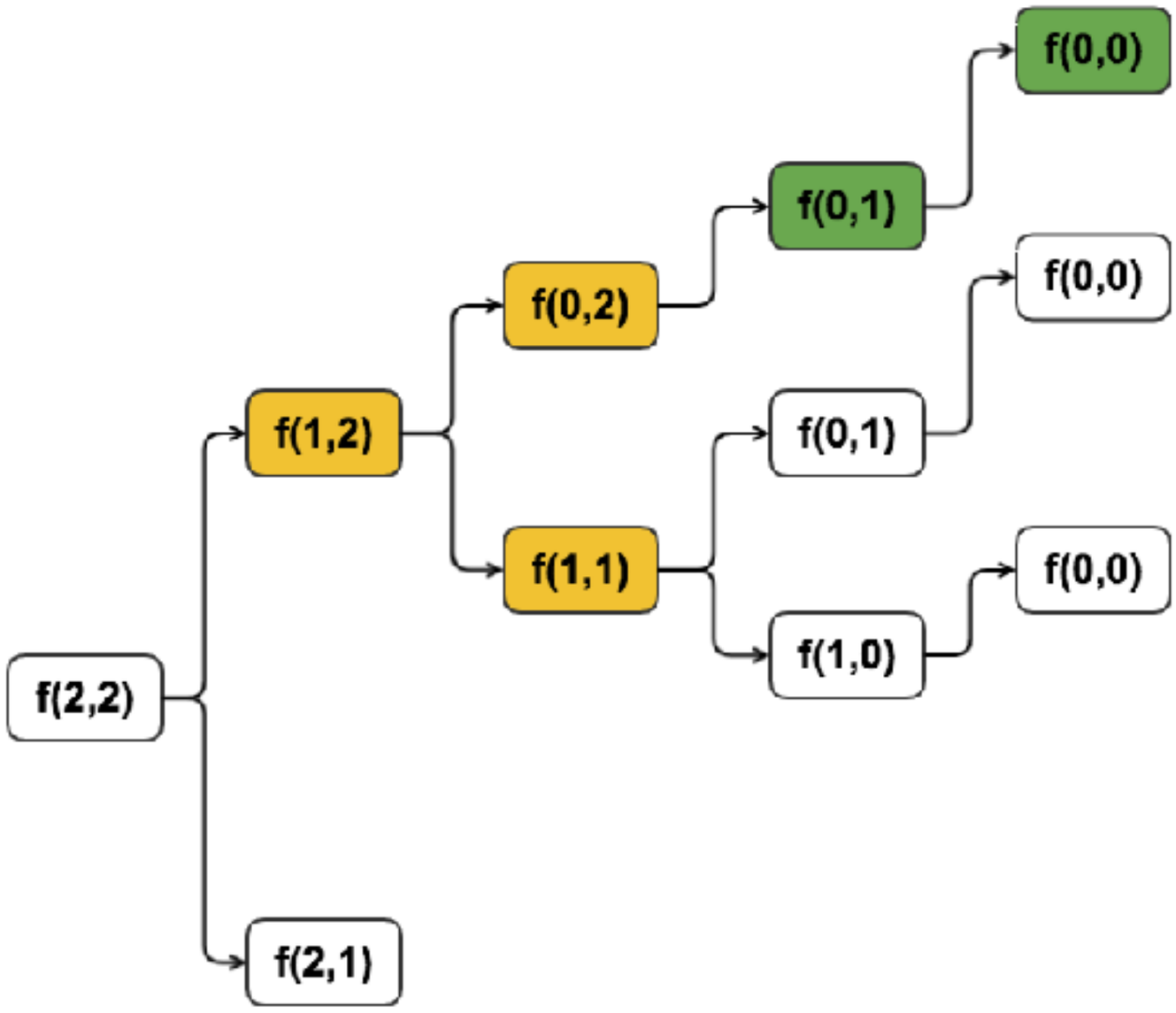
假设到(i,j)的最短路径和为$f(i,j)$

则：

$$f(i,j) = min(f(i-1,j), f(i,j-1)) + grid[i][j]$$

Case Study

1	3	1
1	5	1
4	2	1



Code(recursive)

```
int findPath(vector<vector<int> >& grid, int rowIdx, int colIdx){
    if(rowIdx == 0 && colIdx == 0){return grid[rowIdx][colIdx];}
    if(rowIdx == 0){return findPath(grid, 0, colIdx -1) + grid[0][colIdx];}
    if(colIdx == 0){return findPath(grid, rowIdx - 1, 0) + grid[rowIdx][0];}
    return min(findPath(grid, rowIdx - 1, colIdx), findPath(grid, rowIdx, colIdx - 1)) +
grid[rowIdx][colIdx];
}
int minPathSum1(vector<vector<int> >& grid){
    int rows = grid.size();
    int cols = grid[0].size();
    return findPath(grid, rows-1, cols-1);
}
```

Code(memorize)

```
int findPath1(vector<vector<int> >& grid, int rowIdx, int colIdx, vector<vector<int> >&
results){if(rowIdx == 0 && colIdx == 0){results[0][0] = grid[rowIdx][colIdx];}

    if(rowIdx == 0){results[0][colIdx] = findPath(grid, 0, colIdx -1) + grid[0][colIdx];}

    if(colIdx == 0){results[rowIdx][0] = findPath(grid, rowIdx - 1, 0) + grid[rowIdx][0];}

    if(results[rowIdx][colIdx] > 0){return results[rowIdx][colIdx];}

    results[rowIdx][colIdx] = min(findPath(grid, rowIdx - 1, colIdx), findPath(grid, rowIdx,
colIdx - 1)) + grid[rowIdx][colIdx];

    return results[rowIdx][colIdx];}

/** recursive with memorize **/

int minPathSum2(vector<vector<int> >& grid){int rows = grid.size();int cols = grid.size();

    vector<vector<int> > results(rows, vector<int>(cols, 0));   findPath1(grid, rows - 1,
cols - 1, results);

    return results[rows - 1][cols - 1];}
```

Case Study

1	3	1
1	5	1
4	2	1

1	4	5
2	7	6
6	8	7

Code(dp)

```
int minPathSum(vector<vector<int> >& grid){
    const int rows = grid.size();
    const int cols = grid[0].size();
    vector<vector<int> > results(grid);
    for(int j = 1;j < cols;j++){results[0][j] = results[0][j-1] + grid[0][j];}
    for(int i = 1;i < rows;i++){results[i][0] = results[i-1][0] + grid[i][0];}
    for(int i = 1;i < rows;i++){
        for(int j = 1;j < cols;j++){
            results[i][j] = min(results[i-1][j], results[i][j-1]) + grid[i][j];
        }
}
```

Code(dp)

```
int main(){
    const int rows = 3, cols = 3;
    int grid[][cols] = {1, 3, 1,
                        1, 5, 1,
                        4, 2, 1};
vector<vector<int> > _grid(3, vector<int>(3, 0));
for(int i = 0;i < rows;i++){
    for(int j = 0;j < cols;j++){
        _grid[i][j] = grid[i][j];
    }
}
```

Q/A

1.边界条件怎样得到?

2.空间能够进一步优化?

3.做题的选择是怎样的？(递归搜索，记忆化去冗余，动规)

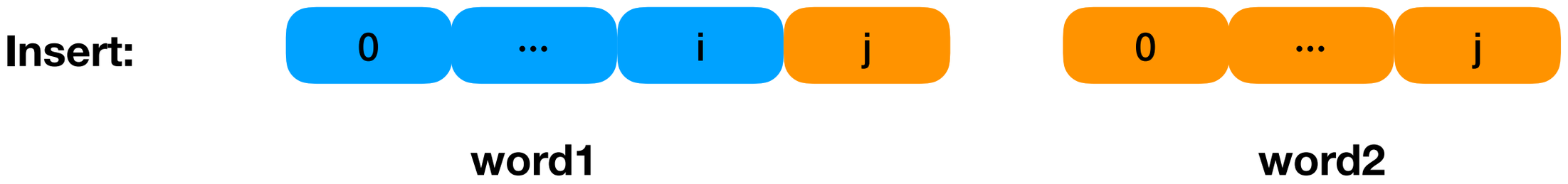
Edit Distance

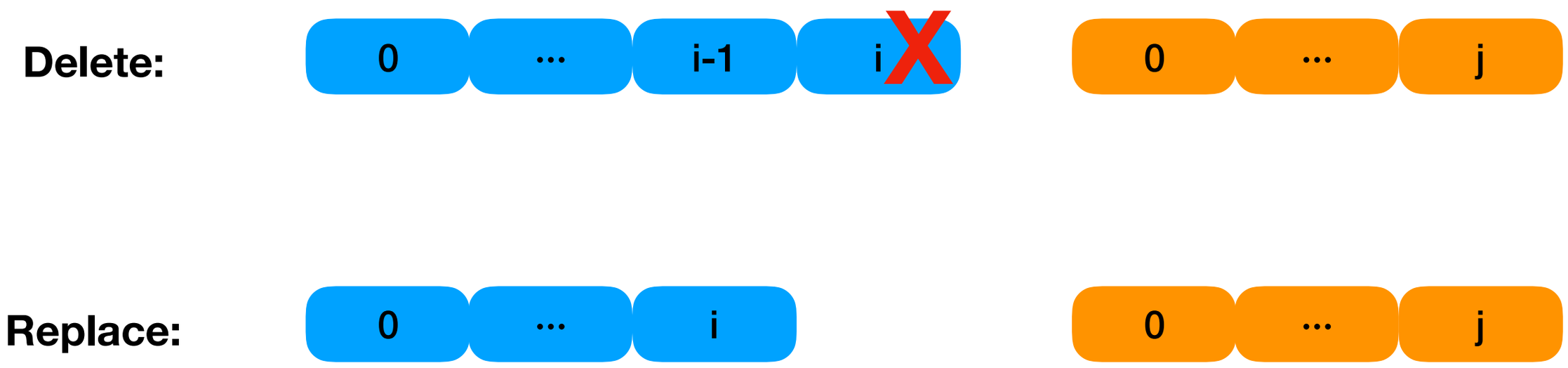
Given two words *word1* and *word2*, find the minimum number of steps required to convert *word1* to *word2*. (each operation is counted as 1 step.)

You have the following 3 operations permitted on a word:

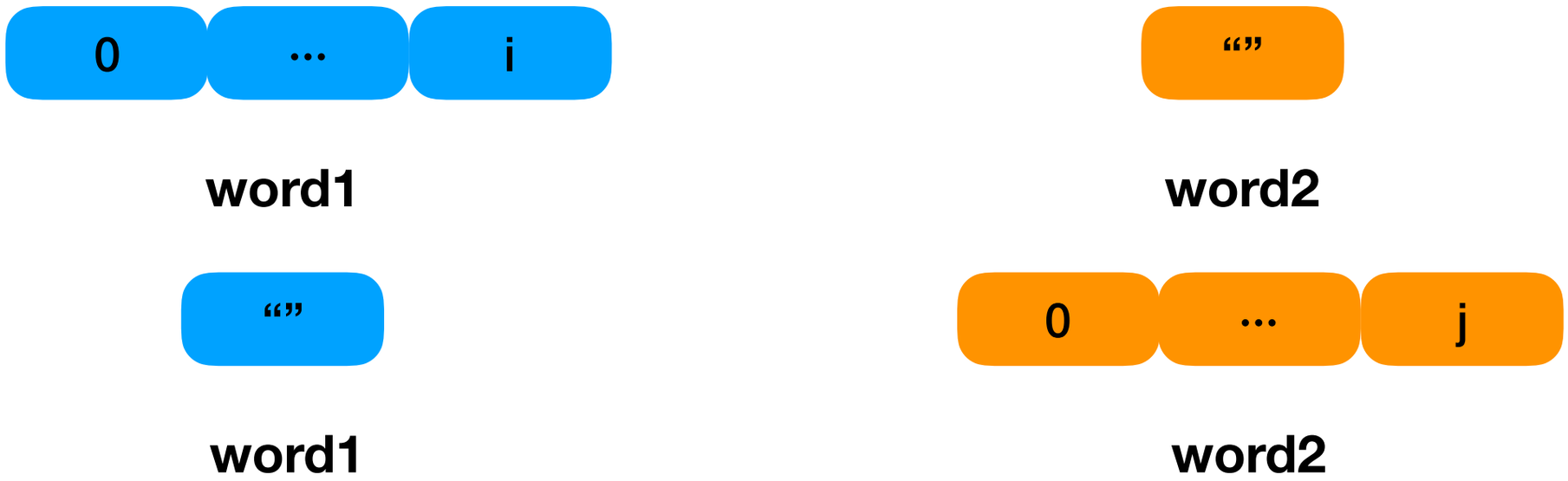
a) Insert a character
b) Delete a character
c) Replace a character

Case Study(word1[i]!=word2[j])





Case Study(word1[i]==word2[j])



corner case

Solution

$f(i,j)$ 从word1的字符 i 变到word2的字符 j 需要的最少的步数

case1: word1[i] = word2[j]

$$f(i,j) = f(i-1,j-1)$$

case2: word1[i] != word2[j]

$$f(i,j) = min\{f(i,j-1), f(i-1,j), f(i-1,j-1)\} + 1$$

Case Study

word1

		word2					
		z	h	p	m	a	t
	0	1	2	3	4	5	6
m	1	1	2	3	3	4	5
a	2	2	2	3	4	3	4
t	3	3	3	3	4	4	3
r	4	4	4	4	4	5	4
i	5	5	5	5	5	5	5
x	6	6	6	6	6	6	6

string word1 = “matrix”

string word2 = “zhpmat”

Code(dp)

```
int minDistance(string& word1, string& word2){
        int word1Len = word1.length();int word2Len = word2.length();
        vector<vector<int> > results(word1Len+1, vector<int>(word2Len+1, 0));
        for(int i = 0;i <= word1Len;i++){results[i][0] = i;}for(int j = 0;j <= word2Len;j++){results[0][j] = j;}
        for(int i = 1;i <= word1Len;i++){
                for(int j = 1;j <= word2Len;j++){
                        if(word1[i-1] == word2[j-1]){results[i][j] = results[i-1][j-1];}
                        else{results[i][j] = min(results[i][j-1],min(results[i-1][j],results[i-1][j-1])) + 1;}
                }
        }
        return results[word1Len][word2Len];
}
```

Q/A

1.rows=word1Len, cols=word2Len的二维数组能不能解决问题？(边界处理)

2.搜索，冗余?

3.能不能找到具体的变换方式(word1->word2)?

4.贪心和动规的区别和联系

5.如果我们把证明片段洒向空中, 如同漫天飞舞的雪花的话, 我们只需要几片就有超过一半的几率判断证明是否正确

Best Time to Buy and Sell Stock(I & II)

Prices:**2, 4, 9, 5, 6**

Diff: **2, 5, -4, 1**

I:f(0)=2,f(1)=7,f(2)=3,f(3)=4 II:maxProfit=2+5+1

Max Subarray $max[f(i)]$

f(i)表示以第i个元素结尾的子数组的最大和，则

$$f(i) = \begin{cases} nums[i] & i = 0 or f(i-1) <= 0 \\ f(i-1) + nums[i] & i! = 0 and f(i-1) > 0 \end{cases}$$

TKS

大家有啥要问的吗?