GeekBand 极客班

互联网人才 + 油站!

GeekBand 极客班

www.geekband.com

6. 动态规划和递归

大纲

1. 动态规划和递归介绍
2. 基本思路
3. 记忆化搜索
4. 经典例题
5. 总结（矩阵，序列动态规划）

递归和动态规划比较

相同

都能分解成若干子问题

不同:

DP存储子问题结果

动态规划介绍

1. Build approach from sub-problems to the final destination

2. Recursion的空间与时间成本

3.Bottom-Up与Top-Down

Fibonacci Number

1, 1, 2, 3, 5…
NON-Dynamic Programming

1) 自顶向下

```
int Fibonacci(int n) {
    if(n == 0)
        return 0;
    if(n == 1)
        return 1;
    return Fibonacci(n-1) + Fibonacci(n-2);
}
```

For fib(8), how many calls to function fib(n)?

DP

avoid repeated calls by remembering function values already calculated

2) 自底向上

```
int array[n] = {0};
array[1] = 1;
for (int i = 2; i < n; i++)
    array[i] = array[i-1] + array[i-2];
```

动态规划的4点要素

1. 状态 (存储小规模问题的结果）
2. 方程 (状态之间的联系，怎么通过小的状态，来算大的状态)
3. 初始化 (最极限的小状态是什么)
4. 答案 (最大的那个状态是什么)

Memorization

Memorize Search *记忆化搜索

Advantage: Easy to think and implement.
Disadvantage: Expensive memory cost.

```
T func(N node, HashTable<N, T>& cache) {
    If (cache.contains(node)) {
        return cache.get(node);
    }
    …
    T sub_res = func(next_node, cache);
    …
    //当前子问题的解，依赖于更小的子问题(s)
    T res = G( sub_res … );
    cache.set(node, res);
    return res;
}
```

算法策略比较

1. Divide and Conquer
2. Dynamic Programming
3. Greedy Algorithm
4. Backtracking

模式识别

1. 用Dynamic Programming (Bottom-Up) 解决收敛结构问题

可以把上述递推关系写在手边，这样做非常有利于理清算法实现的思路。

例如对于斐波那契数列，有递推式：
f(n) = G[f(n-1), f(n-2)] = f(n-1) + f(n-2)

如果出现类似于“所有解”，“所有路径”等关键词，则用Top-down方法更为直接。

Climbing Stairs

Suppose we have a ladder which has n steps. Each time you can either climb 1 or 2 steps. Please write a function to calculate how many distinct ways that can you climb to the top?

CountOfWays(n) = CountOfWays(n–1) + CountOfWays(n-2);

Prime

Compute the n^{th} prime

2, 3, 5, 7…

Prime(n) = G (Prime(n-1), Prime(n-2), Prime(n-3) … Prime(1));
Prime(1) = 2
G( )表示不能整除的关系。

Word Break

Given an input string and a dictionary of words, check if the input string can segment into a space-separated sequence of dictionary words if possible.

For example, if the input string is "applepie" and dictionary contains a standard set of English words, then we would return true since the string "apple pie" is found in dict.

Palindrome Partition

Given a string s, we can partition s such that every segment is a palindrome (e.g, 'abba' is a palindrome, 'a' is a palindrome, 'ab' is not). Please write a function to return the minimum cuts needed for a palindrome partitioning, given string s.

本题可以分为两个部分：判断字串是不是回文*(palindrome)*，如何切割原字符串。

isPalindrome(i , j) = (value(i) == value(j)) AND (isPalindrome(i+1, j-1) OR j – i <= 1) ，

i和j分别表示substring的首坐标和尾坐标

minCut(i) = min(minCut(j+1)+1)，for i <= j < n, and substring(i , j) is palindrome。

“聚合”性质

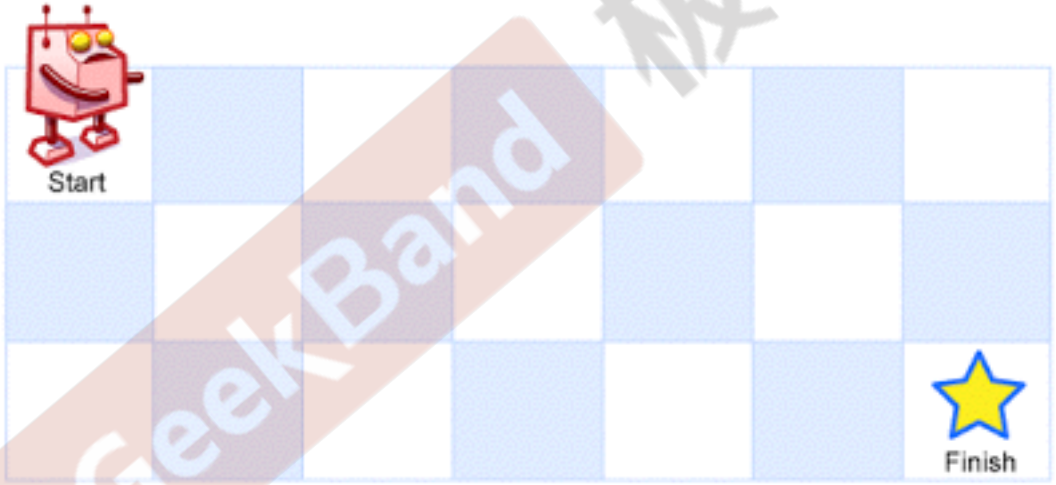
求最值或者求和问题，往往可以进一步优化DP Table的空间。

如果只在乎紧邻的前一个的局部解，而不在乎前几个局部解的问题，就可以接受每次在计算当前解的时候，替换掉那个最优解。

Unique Path

How many paths are there for a robot to go from (0,0) to (x,y), supposing it can only move down and move right.

Matrix DP



res[i][j]=res[i-1][j]+res[i][j-1]; i和j分别表示起点的横纵坐标

Unique Path II

Now consider if some obstacles are added to the grids. How many unique paths would there be?

An obstacle and empty space is marked as 1 and 0 respectively in the grid.

There is one obstacle in the middle of a 3x3 grid as illustrated below.

[
[0,0,0],
[0,1,0],
[0,0,0]
]
The total number of unique paths is 2.

1. 当(i, j)有障碍时dp[i][j] = 0
2. dp[0][j]和dp[i][0]未必为1.
dp[0][j] = obstacleGrid[0][j] ? 0 : dp[0][j-1]
dp[i][0] = obstacleGrid[i][0] ? 0 : dp[i-1][0]
3. 当obstacleGrid [0][0] = 1时，return 0

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.

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

Jump Game

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Determine if you are able to reach the last index.

For example: A = [2,3,1,1,4], return true.

A = [3,2,1,0,4], return false.

Triangle

Given a triangle, find the minimum path sum from top to bottom. Each step you may move to adjacent numbers on the row below.

For example, given the following triangle

```
[
     [2],
    [3,4],
   [6,5,7],
  [4,1,8,3]
]
```

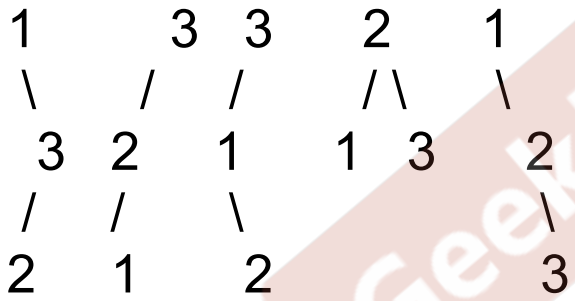
The minimum path sum from top to bottom is 11 (i.e., 2 + 3 + 5 + 1 = 11).

dp[m + 1][n] = min(dp[m][n], dp[m][n - 1]) + triangle[m + 1][n] if n > 0
dp[m + 1][0] = dp[m][0] + triangle[m + 1][0]

Unique Binary Search Trees

Given n, how many structurally unique BST's (binary search trees) that store values 1...n?

For example, Given n = 3, there are a total of 5 unique BST's.



$$dp[i] = \sum(dp[k] * dp[i - k - 1]) \quad 0 <= k < i$$

Coin Change

Given a value N, this N means we need to make change for N cents, and we have infinite supply of each of S = { S1, S2, .. , Sm} valued coins, how many ways can we make the change?

$$ways(i, j) = ways(i-s(j), j) + ways(i, j-1);\ \ i \in [0,N],\ j \in [1,m]$$

Coin Change 2

Given a value N, this N means we need to make change for N cents, and we have infinite supply of each of S = { S1, S2, .. , Sm} valued coins. Please implement a function which gets the minimal number of coins for N cents.

$$minNum(i, j) = min(minNum(i-s(j), j) + 1, minNum(i, j-1));\ \ i \in [0,N], j \in [1,m]$$

最长子序列的问题

对于“最长子序列”问题(即有限空间内，满足一定条件的最长顺序子序列)，本身具有很强的聚合性，可以以如下方式解答：用DP Table来记录以当前节点为末节点的序列的解(至少固定问题的一端，因此不是以“当前节点或之前节点”为末节点)的解，并根据递推关系，由问题空间的起点到达问题空间的终点。

Longest Sub Sequence

Find the longest increasing subsequence in an integer array. E.g, for array {1, 3, 2, 4}, return 3.

maxLength(i) = max{ maxLength(k), k = 0~i-1 and array[i] > array[k] } + 1;

Gas Station

Suppose you are traveling along a circular route. On that route, we have N gas stations for you, where the amount of gas at station i is gas[i]. Suppose the size of the gas tank on your car is unlimited. To travel from station i to its next neighbor will cost you cost[i] of gas. Initially, your car has an empty tank, but you can begin your travel at any of the gas stations. Please return the smallest starting gas station's index if you can travel around the circuit once, otherwise return -1.

array[i] = gas[i] – cost[i]。问题转化为，找到一个起始位置*index*，将*array*依此向左*shift*，即*index->0(index*对应新的数组下标*0)*，*index+1->1…*，使得对于任意*0 <= i < n*，满足序列和*subSum(0, i)*大于*0*。

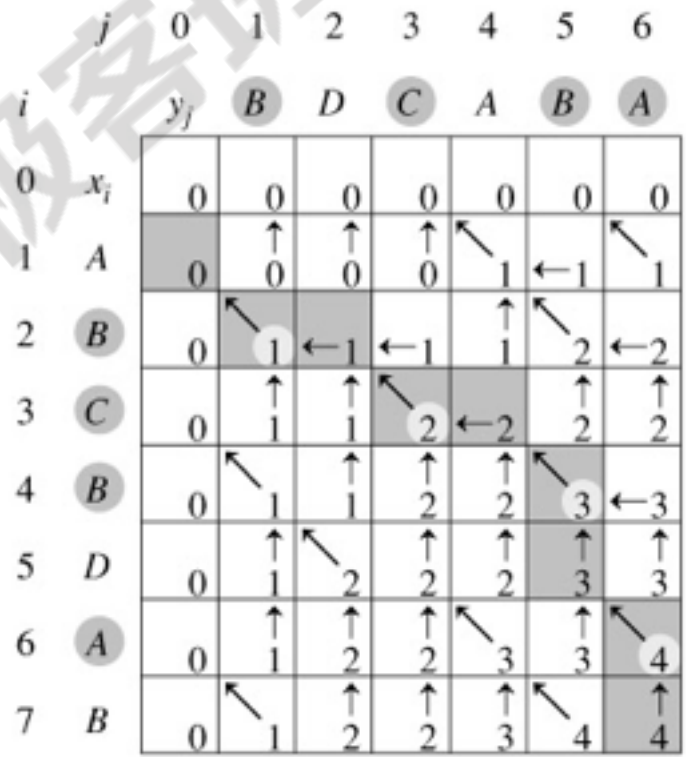
Longest Common Sequence

Please write a function to calculate the Longest Common Subsequence (LCS) given two strings.
LCS for input Sequences “ABCDGH” and “AEDFHR” is “ADH” of length 3.
LCS for input Sequences “AGGTAB” and “GXTXAYB” is “GTAB” of length 4.

Length(i,j) = (str1[i-1] == str2[j-1]) ? Length(i-1, j-1) + 1 : Max { Length(i,j-1), Length(i-1,j) }

LCS

```
LCS(x,y,i,j)
    if x[i] = y[j]
        then C[i,j] ← LCS(x,y,i-1,j-1)+1
    else C[i,j] ← max{LCS(x,y,i-1,j),LCS(x,y,i,j-1)}
    return C[i,j]
```



i	x_i	j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
0	x_i		0	0	0	0	0	0	0
1	A		0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B		0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C		0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B		0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D		0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A		0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B		0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

模式识别

如果当前节点的解，既依赖于前驱问题的解，又依赖于后驱问题的解，但这两部分又互相独立，则可以分别自左开始DP，计算从最左节点到当前节点的结果；自右开始DP，计算从最右节点到当前节点的结果；再用同一个DP Table来合并解。

Product without self

Given an array of integers, write a function to replace each element with the product of all elements other than that element.

假定数组*A={4, 3, 2, 1, 2}*，则*OUTPUT={12, 16, 24, 48, 24}*。

*{ 1, a[0], a[0]*a[1], a[0]*a[1]*a[2], }*
*{ a[1]*a[2]*a[3], a[2]*a[3], a[3], 1, }*

Hold Water

You are given an array of n non-negative integers. Each value means the height of a histogram. Suppose you are pouring water onto them, what is the maximum water it can hold between a left bar and a right bar (no separation)?

当前节点的储水量，等于左侧最高海拔与右侧最高海拔的较小值减去当前节点的海拔。

模式识别

3. 用Memorization Technique(Top-Down)解决收敛结构问题

Memorization是Top-Down形式的Dynamic Programming，也可以用来解决前述的问题(但空间上可能效率不及Bottom-Up形式的DP)。

Memoization的核心在于，在原有递归框架下，储存子问题的计算结果，在重复计算子问题时返回已经计算的值。

Tallest stack of boxes

Given a set of boxes, each one has a square bottom and height of 1. Please write a function to return the tallest stack of these boxes. The constraint is that a box can be put on top only when its square bottom is restrictively smaller.

Word Break II

Given a string and a dictionary of words, please write a function to add space into the string, such that the string can be completely segmented into several words, where every word appears in the given dictionary.

Edit Distance

`Andrew`

`Amdrewz`

1. substitute **m** to **n**
2. delete the **z**

Distance = 2

- **Insertion:** the last entry in the top row is empty, $E(i,j) = E(i,j-1) + 1$.
- **Deletion:** the last entry in the bottom row is empty, $E(i,j) = E(i-1,j) + 1$.
- **Substitution:** both rows have characters in the last column that are different, $E(i,j) = E(i-1,j-1) + 1$.
- **No action:** both rows end in the same character, $E(i,j) = E(i-1,j-1)$.

Let P be the logical proposition $A[i] \neq B[j]$ and denote by $|P|$ its indicator variable: $|P| = 1$ if P is true and $|P| = 0$ if P is false. We can now summarize and for $i,j > 0$ get the edit distance as the smallest of the possibilities:

$$E(i,j) = \min \left\{ \begin{array}{l} E(i,j-1) + 1 \\ E(i-1,j) + 1 \\ E(i-1,j-1) + |P| \end{array} \right\}.$$

Edit Distance

		P	A	R	K
	0	1	2	3	4
S	1	1	2	3	4
P	2	1	2	3	4
A	3	2	1	2	3
K	4	3	2	2	2
E	5	4	3	3	3

Final cost of aligning all of both strings.

D(i,j) = score of **best** alignment from *s1..si* to *t1..tj*

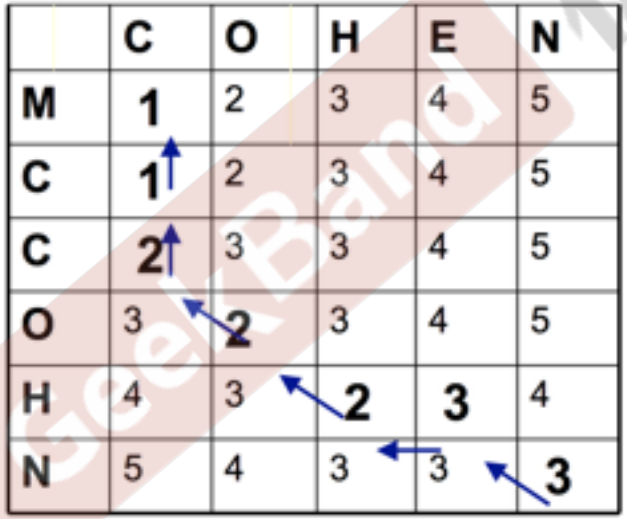
$$= \min \begin{cases} D(i-1,j-1)+d(si,tj) & \textit{//substitute} \\ D(i-1,j)+1 & \textit{//insert} \\ D(i,j-1)+1 & \textit{//delete} \end{cases}$$

```
def stredit (s1,s2):
    "Calculate Levenstein edit distance for strings s1 and s2."
    len1 = len(s1) # vertically
    len2 = len(s2) # horizontally
    # Allocate the table
    table = [None]*(len2+1)
    for i in range(len2+1): table[i] = [0]*(len1+1)
    # Initialize the table
    for i in range(1, len2+1): table[i][0] = i
    for i in range(1, len1+1): table[0][i] = i
    # Do dynamic programming
    for i in range(1,len2+1):
        for j in range(1,len1+1):
            if s1[j-1] == s2[i-1]:
                d = 0
            else:
                d = 1
            table[i][j] = min(table[i-1][j-1] + d,
                              table[i-1][j]+1,
                              table[i][j-1]+1)
```

Edit Distance

Trace

A *trace* indicates where the min value came from, and can be used to find edit operations and/or a best *alignment* (may be more than 1)



	C	O	H	E	N
M	1	2	3	4	5
C	1	2	3	4	5
C	2	3	3	4	5
O	3	2	3	4	5
H	4	3	2	3	4
N	5	4	3	3	3

Summary

Sequence DP -
1. Climbing Stairs
2. Jump game
3. Palindrome Partitioning ii
4. Word Break
5. Triangle
6. Longest Increasing Subsequence
7. Max Subarray Sum

Two Sequences DP -
1. Coin Change
2. Edit Distance
3. LCS

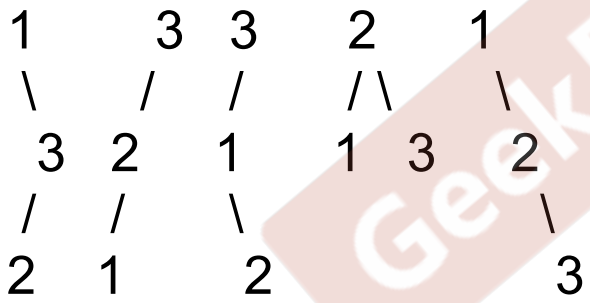
Matrix DP -
1. Minimum Path Sum
2. Triangle
3. Unique Path I, II

Homework

Unique Binary Search Trees II

Given n, generate all structurally unique BST's (binary search trees) that store values 1...n.

For example, Given n = 3, your program should return all 5 unique BST's shown below.



Jump Game II

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Your goal is to reach the last index in the minimum number of jumps.

For example: Given array A = [2,3,1,1,4]

The minimum number of jumps to reach the last index is 2. (Jump 1 step from index 0 to 1, then 3 steps to the last index.)