1. ----参考 tut04 的第五题（true，false）

Q1

Boolean operators NAND and NOR are defined as follows

NAND

	true	false
true	false	**true**
false	**true**	**true**

NOR

	true	false
true	false	false
false	false	**true**

You are given a boolean expression consisting of a string of the symbols true, false, separated by operators AND, OR, NAND and NOR but without any parentheses. Count the number of ways one can put parentheses in the expression such that it will evaluate to true. (20 pts)

> T & F | T & F & F NAND T

(0, n)

子问题：T(i, j) represents the number of ways to parenthesize the symbols between i and j to get True.

还需要知道：F(i, j) represents the number of ways to parenthesize the symbols between i and j to get False.

BaseCase: T(i, i) F(i, i)

When symb[i] == T, T[i][i]=1, F[i][i] = 0

when symn[i] == F, F[i][i]=1, T[i][i] = 0

Recursion:

T(0, n)

Let **T(i, j)** represents the number of ways to parenthesize the symbols between i and j (both inclusive) such that the subexpression between i and j evaluates to **true**.

Let **F(i, j)** represents the number of ways to parenthesize the symbols between i and j (both inclusive) such that the subexpression between i and j evaluates to **false**.

对 T(i, j) 我们假设我们已经有了所有 T(i, k)和T(k+1, j)子问题，对所有的k属于[i, j)

对 F(i, j) 我们假设我们已经有了所有 F(i, k)和F(k+1, j)子问题，对所有的k属于[i, j)

情况1:

（Exp1(i...k)）& （Exp2(k+1... j)）== (i, j)

使得新expression为真，表达式1和2必须都为真。

已知表达式1为真有T(i, k)种方式，表达式2为真有T(k+1, j)种方式，

因此T(i, j)为真有T(i, k) * T(k+1, j)种方式。

情况2:

（Exp1(i...k)） | （Exp2(k+1... j)） == (i, j)

使得新expression为真，表达式1和表达式2至少有一者为真。

Exp1 = T, Exp2 = T => T(i, k) * T(k+1, j)

Exp1 = T, Exp2 = F => T(i, k) * F(k+1, j)

Exp1 = F, Exp2 = T => F(i, k) * T(k+1, j)

已知表达式1为真有T(i, k)种方式，表达式2为真有T(k+1, j)种方式，

表达式1为假有F(i, k)种方式，表达式2为假有F(k+1, j)种方式，

因此我们的表达式为真时有 T(i, k) * T(k+1, j) + T(i, k) * F(k+1, j) + F(i, k) * T(k+1, j)

情况3:

（Exp1(i...k)） NAND （Exp2(k+1... j)） == (i, j)

F(i, k) * F(k+1, j) + T(i, k) * F(k+1, j) + F(i, k) * T(k+1, j)

情况4:

NOR

T(i, j) = F(i, k) * F(k+1, j)

Total(i, k) = F(i, k) + T(i, k)

(补全T表和F表的情况)

$$T(i,j) = \sum_{k=i}^{j-1} \begin{Bmatrix} T(i,k) * T(k+1,j) & if \quad operator \quad [k] \quad is' \& ' \\ Total(i,k) * Total(k+1,j) - F(i,k) * F(k+1,j) & if \quad operator \quad [k] \quad is' | ' \\ T(i,k) * F(k+1,j) + F(i,k) * T(k+1,j) + F(i,k) * F(k+1,j) & if \quad operator \quad [k] \quad is' NAND' \\ F(i,k) * F(k+1,j) & if \quad operator \quad [k] \quad is' NOR' \end{Bmatrix}$$

$$F(i,j) = \sum_{k=i}^{j-1} \begin{Bmatrix} Total(i,k) * Total(k+1,j) - T(i,k) * T(k+1,j) & if \quad operator[k] \quad is' \& ' \\ F(i,k) * F(k+1,j) & if \quad operator[k] \quad is' | ' \\ T(i,k) * T(k+1,j) & if \quad operator[k] \quad is' NAND' \\ T(i,k) * F(k+1,j) + F(i,k) * T(k+1,j) + T(i,k) * T(k+1,j) & if \quad operator[k] \quad is' NOR' \end{Bmatrix}$$

$$Total(i,j) = T(i,j) + F(i,j)$$

https://www.geeksforgeeks.org/boolean-parenthesization-problem-dp-37/

T & F | T NAND T

1---2---3---4

T(1, 2) = T(1, 1) * T(2,2)

T(2, 3) =T(2, 2) * T(3, 3) + T(3, 3) * F(3, 3) + F(2, 2) * T(3, 3) = 1

T(1, 3) = T(1, 1) * T(2, 3) + T(1, 2) *...

T	1	2	3	4
1	1	0		
2		0	1	
3			1	
4				1

F	1	2	3	4
1	0			
2		1		
3			0	
4				0

Q2

You are given a 2D map consisting of an C × R grid of squares; in each square there is a number representing the elevation of the terrain at that square. Find a path going from square (1, R) which is the top left corner of the map to square (C, 1) in the lower right corner which from every square goes only to the square immediately below or the square immediately to the right so that the number of moves from lower elevation to higher elevation along such a path is as small as possible. (20 pts)

子问题：到达map[i][j]时的最小值。

dp[i][j]记录到达点[i][j]的最小爬坡次数

(Base Step) 初始化dp[0][k], dp[l][0], k [0, R]. l [0, C]

Recursion:

假设我们已经解决了所有k < i, m < j 的子问题dp[k][m], 我们需要找到dp[i][j]的大小。(Recursion)

已知到达map[i][j]必定通过map[i - 1][j]或者map[i][j - 1]，所以

dp[i][j] = min{dp[i-1][j]+1, dp[i][j-1]+1} if map[i][j] > map[i-1][j] and map[i][j] > map[i][j-1]

= min{dp[i-1][j], dp[i][j-1]+1} if map[i][j] <= map[i-1][j] and map[i][j] > map[i][j-1]

= min{dp[i-1][j]+1, dp[i][j-1]} if map[i][j] > map[i-1][j] and map[i][j] <= map[i][j-1]

= min{dp[i-1][j], dp[i][j-1]} if map[i][j] <= map[i-1][j] and map[i][j] <= map[i][j-1]

dp[C][1]

dp[i][j] = min{dp[i-1][j]+1, dp[i][j-1]+1} if map[i][j] > map[i-1][j] and map[i][j] > map[i][j-1]

= min{dp[i-1][j], dp[i][j-1]+1} if map[i][j] <= map[i-1][j] and map[i][j] > map[i][j-1]

= min{dp[i-1][j]+1, dp[i][j-1]} if map[i][j] > map[i-1][j] and map[i][j] <= map[i][j-1]

= min{dp[i-1][j], dp[i][j-1]} if map[i][j] <= map[i-1][j] and map[i][j] <= map[i][j-1]

dp[C][1]

O(C*R)复杂度

Q3

In a pond there is a sequence on n lily pads arranged in a straight line: 1, 2, 3 . . . n . On lily pad i there are fi ≥ 0 flies. On lily pad 1 there is a frog sitting. The frog can only jump forward from a lily pad i to either lily pad i + 3 or lily pad i + 4. Find the largest number of flies that the frog can catch. (20 pts)

青蛙跳荷叶

子问题：假设跳到第i片荷叶能吃到所有的最大苍蝇数量是opt(i)

Base case:

i = 1, opt(1) = f1

i = 2, opt(2) = 0

i = 3, opt(3) = 0

i = 4, opt(4) = f1 + f4

i = 5, opt(5) = f1 + f5

状态转移方程（recursion）：

假设我们已经知道了所有j<i且i>4的荷叶，对于第i片荷叶 opt(i) = max {opt(i - 4) + fi, opt(i - 3) + fi}.

遍历一次，就能知道答案了，O(n)

Q4

You are on vacation for N days at a resort that has three possible activities 1,2 and 3. For each day i, for each activity 1,2 or 3, you've determined how much enjoyment e(i,j) (1 ≤ i ≤ n; 1 ≤ j ≤ 3) you will get out of that activity if you do it on that particular day (the same activity might give you a different amounts of enjoyment at different days). However, you are not allowed to do the same activity two days in a row. Design an algorithm for determining the maximum total enjoyment possible over the entire stay of N days and the sequence of activities you should do at each day. (20 pts)

假设活动为1，2，3**第i天的enjoyment**（只对这一天，一个活动）分别为 e[i][1],e[i][2], e[i][3]， dp[i][1],dp[i][2],dp[i][3]是到第i天做1，2，3活动的**总共的最大的**快乐值

子问题:

max{我在第i天做活动0的快乐值,

我在第i天做活动1的快乐值,

我在第i天做活动2的快乐值}

= 我的最大快乐值

Base case：第一天直接记录

dp[0][1]=e[0][1],

dp[0][2]=e[0][2],

dp[0][3]=e[0][3]

Recursion: 现在是第i天，我们已经解决了所有j < i的子问题

dp[i][1] = max {dp[i-1][2]+ e[i][1], dp[i-1][3]+e[i][1]}

dp[i][2] = max {dp[i-1][1]+ e[i][2], dp[i-1][3]+e[i][2]}

dp[i][3] = max{dp[i-1][0] + e[i][0], dp[i-1][1]+e[i][3]}

第N天的结果= max{dp[N][0],dp[N][1], dp[N][2]}

Q5

Given a weighted directed graph G(V,E), find a path in G (possibly self- intersecting) of length exactly K that has the maximum total weight. . The path can visit a vertex multiple times and can traverse an edge also multiple times. It can also start and end at arbitrary vertices or even start and end at the same vertex.(20 pts)

maximum total weight = longest path

Let dp[S][X][j] be the longest path from node **S** to node **X** using exactly **J** edges in total. Using this, dp[X][J+1] can be calculated as:

S , X ∈vertices set and j ∈ [0, K]

The result for the problem can be computed by following below steps:

Base Step:

J=0 dp[X][X][0]=0

J=1 dp[i][j][1] 储存了每条边的值（weight）

Recursion Step:

```
dp[S][X][J+1] = max(dp[S][Y][J]+weight[{Y, X}])
for all Y which has an edge from Y to X.
```

结果：

查找最大的dp[S][X][K]，S，X ∈ 顶点集

https://www.geeksforgeeks.org/shortest-path-with-exactly-k-edges-in-a-directed-and-weighted-graph-set-2/