# 动态规划与记忆化搜索

主讲人：邓哲也

# 记忆化搜索

实现一个函数，用“搜索”的方法实现 DP 的更新。

通常用于解决转移顺序不方便人为确定的 DP。

# POJ 3176

数塔。

5

7

3 8

8 1 0

2 7 4 4

4 5 2 6 5

# POJ 3176

设 f[i][j] 表示走到了第 i 行第 j 列的最大值。

正常 DP：

f[i][j] = max(f[i – 1][j], f[i – 1][j – 1]) + a[i][j]

# POJ 3176

搜索：

```
int dp(int i, int j){
        if (i == 0) return 0;
        return a[i][j] + max(dp(i - 1, j), dp(i - 1, j - 1));
}
```

这样有很多状态其实会重复计算。

对于计算过的 dp(i, j)，我们用 f[i][j] 来存。

f 初始化为 -1

# POJ 3176

记忆化搜索：

```
int dp(int i, int j){
        if (i == 0) return 0;
        if (f[i][j] >= 0) return f[i][j];
        f[i][j] = a[i][j] + max(dp(i - 1, j), dp(i - 1, j - 1));
        return f[i][j];
}
```

# POJ 1088 滑雪

Michael喜欢滑雪百这并不奇怪， 因为滑雪的确很刺激。可是为了获得速度，滑的区域必须向下倾斜，而且当你滑到坡底，你不得不再次走上坡或者等待升降机来载你。Michael想知道载一个区域中最长底滑坡。区域由一个二维数组给出。数组的每个数字代表点的高度。下面是一个例子

1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

一个人可以从某个点滑向上下左右相邻四个点之一，当且仅当高度减小。在上面的例子中，一条可滑行的滑坡为24-17-16-1。当然25-24-23-...-3-2-1更长。事实上，这是最长的一条。

# POJ 1088 滑雪

```
int dp(int i, int j){
        if (f[i][j] > 1) return f[i][j];
        for(int k = 0;k < 4;k ++){
                int x = i + dx[k], y = j + dy[k];
                if (x >= 1 && x <= r && y >= 1 && y <= c)
                        if(a[i][j] > a[x][y])
                                f[i][j] = max(f[i][j], dp(x, y) + 1);
        }
        return f[i][j];
}
```

# POJ 1088 滑雪

```
for(int i = 1;i <= r;i ++)
        for(int j = 1;j <= c;j ++)
                f[i][j] = 1;
int ans = 0;
for(int i = 1;i <= r;i ++)
        for(int j = 1;j <= c;j ++)
                ans = max(dp(i, j), ans);
printf("%d\n", ans);
```

# 下节课再见