实验舱
青少年编程
走近科学 走进名校

蛟龙五班

递推

Mas

递推

递推是一种重要的数学方法,在数学的各个领域中都有广泛的运用,也是计算机用于数值计算的一个重要算法

递推求解的问题,往往在已知条件和所求问题之间总存在着某种相互联系的关系

在计算时,若找到前后过程之间的数量关系(即递推式),那么从问题出发逐步即可推到已知条件

递推算法的首要问题是得到相邻的数据项间的关系(即递推关系)

递推算法避开了求通项公式的麻烦,把一个复杂的问题的求解,分解成了连续的若干步简单运算

一般说来,可以将递推算法看成是一种特殊的迭代算法

#1895 移动汉诺塔

题目描述

在世界中心贝拿勒斯（在印度北部）的圣庙里，一块黄铜板上插着三根宝石针。

印度教的主神梵天在创造世界的时候，在其中一根针上从下到上地穿好了由大到小的 64 片金片，这就是所谓的汉诺塔。

不论白天黑夜，总有一个僧侣在按照下面的法则移动这些金片：

> *一次只移动一片，不管在哪根针上，小片必须在大片上面。*

僧侣们预言，当所有的金片都从梵天穿好的那根针上移到另外一根针上时，世界就将在一声霹雳中消灭，而梵塔、庙宇和众生也都将同归于尽。

现在 Mas 开始玩汉诺塔游戏， 他放了 n 片黄金圆盘在第 1 根柱子上，从上到下依次编号为 $1 \sim n$ ， 1 号圆盘最小， n 号圆盘最大。

现在 Mas 想把圆盘全部移动到第 3 根柱子上，移动过程中 Mas 必须准守游戏规则。

现在 Mas 想知道他完成游戏的最小移动次数。

输入格式

输入一行一个整数 $n(1 \leq n \leq 60)$

输出格式

输出 Mas 的最少移动次数

#1895 移动汉诺塔

设f_i表示i个圆盘移动到任意一根柱子(可借助辅助的柱子)的最少移动次数

显然$f_0 = 0$,当$i \geq 1$时:

1. 先将前$i-1$个圆盘移动到辅助的柱子
2. 再将第i个圆盘移动到目标柱子
3. 最后将前$i-1$个圆盘移动到目标柱子

注意第一步和第三步正好和原问题一模一样,只是规模减了一

即

$$f_i = 2 \times f_{i-1} + 1$$

f_i的递推关系如下

$$f_i = \begin{cases} 1, i = 0 \\ 2 \times f_{i-1} + 1, i \geq 1 \end{cases}$$

#1030、爬楼梯

题目描述

树老师爬楼梯,他可以每次走 1 级或者 2 级,输入楼梯的级数,求不同的走法数

例如：楼梯一共有 3 级,他可以每次都走一级,或者第一次走一级,第二次走两级,也可以第一次走两级,第二次走一级,一共 3 种方法

输入格式

输入包含若干行,每行包含一个正整数 N ,代表楼梯级数, $1 \leq N \leq 36$

输出格式

不同的走法数,每一行输入对应一行输出

输入样例

```
5
8
10
```

输出样例

```
8
34
89
```

#1030、爬楼梯

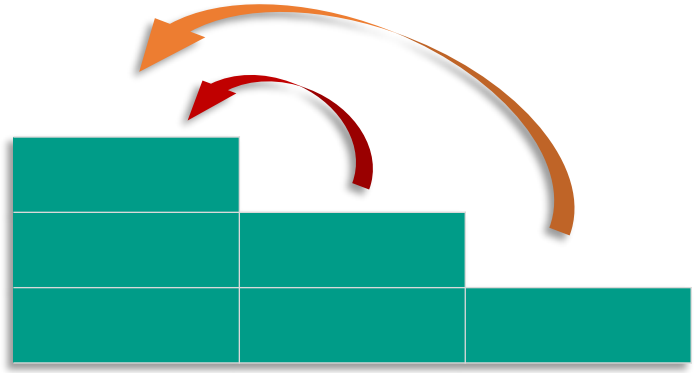
设f_i表示到达第i级台阶的方案数

第0、1级台阶只有1种走法

第$i(i \geq 2)$级台阶

- 可由第$i-1$级走1步到达
- 可由第$i-2$级走2步到达

根据加法原理有$f_i = f_{i-1} + f_{i-2}$



$$f_i = \begin{cases} 1, & i = 0 \\ 1, & i = 1 \\ f_{i-1} + f_{i-2}, & i \geq 2 \end{cases}$$

使用数组存储f_i递推计算即可

#1347、路径计数

题目描述

从一个 n 行 m 列的迷宫的左上角走到右下角,每次只能向下或者向右走一步,一共有多少种走法

一个 $n \times m$ 的网格,你一开始在 $(1,1)$,即左上角

每次只能移动到下方相邻的格子或者右方相邻的格子,问到达 (n,m) ,即右下角有多少种方法

输入格式

输入 2 个整数 $n,m(1 \leq n \leq m \leq 30)$

输出格式

输出方案数

样例输入1

```
3 3
```

样例输出1

```
6
```

设$f_{i,j}$表示到达(i,j)的方案数

第1行/列方格只有1种走法

此外的位置(i,j)

- 可由上方$(i-1,j)$走1步到达
- 可由右方$(i,j-1)$走1步到达

$$f_{i,j}=\begin{cases}1, & i=1\\ 1, & j=1\\ f_{i-1,j}+f_{i,j-1}, & \text{else}\end{cases}$$

#1035、过河卒

题目描述

棋盘上 A 点有一个过河卒,需要走到目标 B 点

卒行走的规则：可以向下、或者向右

同时在棋盘上 C 点有一个对方的马,该马所在的点和所有跳跃一步可达的点称为对方马的控制点,因此称之为**马拦过河卒**

棋盘用坐标表示, A 点 $(0,0)$ 、 B 点 (n,m) (n , m 为不超过 20 的整数),同样马的位置坐标是需要给出的

现在要求你计算出卒从 A 点能够到达 B 点的路径的条数,假设马的位置是固定不动的,并不是卒走一步马走一步

输入格式

一行四个数据,分别表示 B 点坐标和马的坐标。

输出格式

一个数据,表示所有的路径条数

输入样例#1

```
6 6 3 3
```

输出样例#1

```
6
```

提示说明

结果可能很大!

设$f_{i,j}$表示到达(i,j)的方案数

若(i,j)为马的控制点, $f_{i,j}=0$

若(i,j)不为马的控制点,递推关系与上题一致

#1069、骨牌铺方格

题目描述

在 $n \times m$ 的一个长方形方格中,用一个 $1 \times n$ 的骨牌铺满方格,输入 n, m ,输出铺放方案的总数

输入说明

输入两个数整数 n, m

输出格式

输入方案数,结果可能很大输出 $\mod 1000000007$ 的结果

输入样例

```
2 3
```

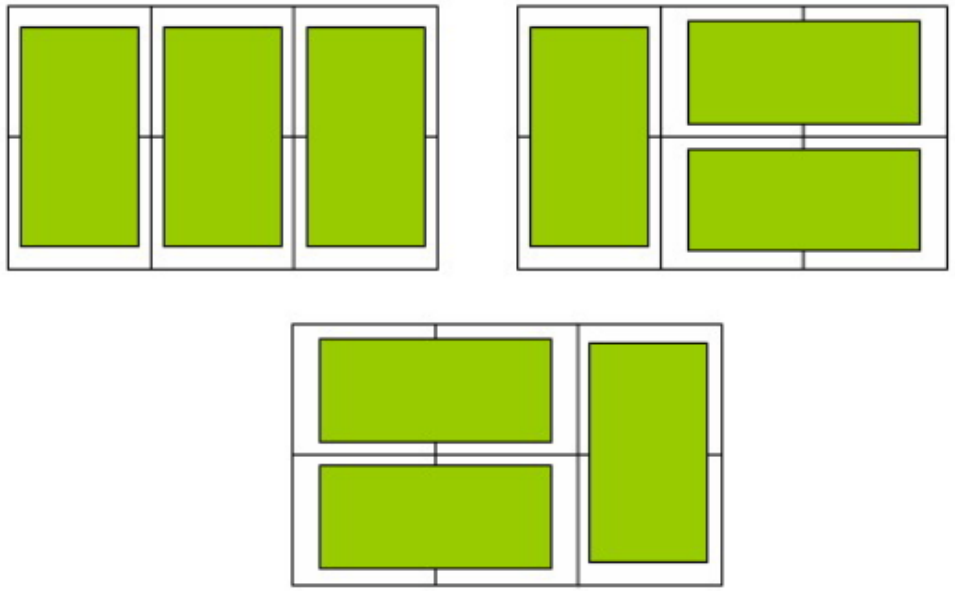
输出样例

```
3
```

数据规模

对于 20% 的数据 $1 \leq n, m \leq 100$
对于 100% 的数据 $1 \leq n, m \leq 10^7$

样例解释



#1069、骨牌铺方格

逐列考虑,设 f_i 表示将n行第$1 \sim i$列全部铺满的方案数

显然$f_0 = 1$

当 $i \geq 1$ 时,如何计算 f_i ?我们对最后一列（第 i 列）的情况分类讨论,只可能是以下两种:

- 正好有一个 $1 \times n$ 的骨牌竖直铺满第 i 列,那么 $f_i \mathrel{+}= f_{i-1}$
- 有 n 个骨牌横向铺着,结束位置正好是第 i 列(需满足$i \geq n$)。那么 $f_i \mathrel{+}= f_{i-n}$。

时间复杂度$O(m)$

#1031、数字金字塔

题目描述

观察下面的数字金字塔,写一个程序查找从最高点到底部任意处结束的路径,使路径经过数字的和最大

每一步可以从当前点走到左下方的点也可以到达右下方的点

在上面的样例中,从 13 到 8 到 26 到 15 到 24 的路径产生了最大的和 86

输入格式

第一个行包含 $R(1 \leq R \leq 1000)$,表示行的数目

后面每行为这个数字金字塔特定行包含的整数

所有的被供应的整数是非负的且不大于 100

输出格式

单独的一行,包含那个可能得到的最大的和

输入样例

```
5
13
11 8
12 7 26
6 14 15 8
12 7 13 24 11
```

输出样例

```
86
```

dfs?

时间复杂度?

#1031、数字金字塔

从$(1,1)$到达第R层的路径,等价于第R层开始每个位置选择两条合法的路径往上走

设$f_{i,j}$表示到达(i,j)时的最大路径和

若$i=R$显然$f_{R,j}=a_{R,j}$

若$1\le i<R$,对于(i,j)只能从$(i+1,j)$或$(i,j+1)$到达,取其两种情况的最大值并累加$a_{i,j}$

$$f_{i,j}=\max\{f_{i+1,j},f_{i+1,j+1}\}+a_{i,j}$$

从第R层到第1层递推,$f_{1,1}$即为答案

时间复杂化度$O(R^2)$

回忆贪心中最优子结构,尝试过证明该递推做法的正确性

记忆化搜索

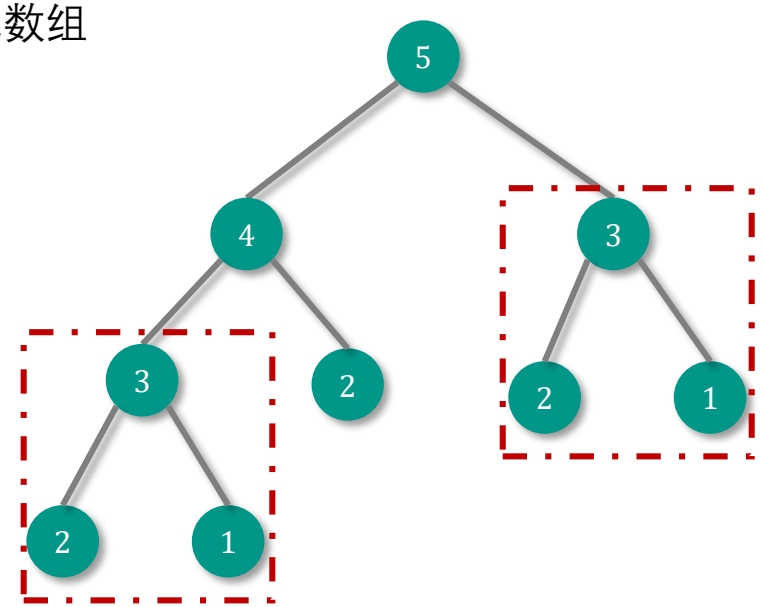
记忆化搜索是一种通过记录已经遍历过的状态的信息(空间换时间)

从而避免对同一状态重复遍历的搜索实现方式

当搜索到某一状态时

- 若该状态的答案在记忆化数组中时,直接返回答案
- 若不在记忆化数组中时,计算出答案后存入记忆化数组

记忆化搜索能够确保了每个状态只搜索一次



fibonacci(n)	调用次数
0	34
1	55
2	34
3	21
4	13
5	8
6	5
7	3
8	2
9	1
10	1

#1613、递归函数

题目描述

对于一个递归函数 $w(a,b,c)$

如果 $a \le 0 \ || \ b \le 0 \ || \ c \le 0$ 返回 1

如果 $a \ge 20 \ || \ b \ge 20 \ || \ c \ge 20$ 返回 $w(20,20,20)$

如果 $a < b$ 且 $b < c$ 就返回 $w(a,b,c-1)+w(a,b-1,c-1)-w(a,b-1,c)$

其它的情况就返回 $w(a-1,b,c)+w(a-1,b-1,c)+w(a-1,b,c-1)-w(a-1,b-1,c-1)$

这是个简单的递归函数,但实现起来可能会有些问题

当 a,b,c 均为 15 时,调用的次数将非常的多,你要想个办法才行

> $w(30,-1,0)$ *既满足条件* 1 *又满足条件* 2
> *这种时候我们就按最上面的条件来算所以答案为* 1

输入格式

输入若干行
并以 $-1,-1,-1$ 结束。
保证输入的数在 $[-2^{63},2^{63}-1]$ 之间,并且是整数

输出格式

输出若干行,每行一个答案

注意递归函数中边界问题

避免记忆化数组出现下标越界

#2285、专业小偷

题目描述

有一个专业的小偷，计划偷窃沿街的 n 个房屋。

每间房内都藏有一定的现金 a_i ，影响你偷窃的唯一制约因素就是相邻的房屋装有相互连通的防盗系统，如果两间相邻的房屋在同一晚上被小偷闯入，系统会自动报警。

给出每个房屋存放金额，请你计算你 **不触动警报装置** 的情况下 ，小偷一夜之内能够偷窃到的最高金额。

输入格式

第一行一个正整数 n 表示房屋的数量

第二行 n 个非负整数，每个整数 a_i 表示房屋内的金额

输出格式

输出一行一个整数,表示能获得的最大金额

数据规模:

对于 20% 的数据 $1 \leq n \leq 20$

对于 40% 的数据 $1 \leq n \leq 1000$

对于 100% 的数据 $1 \leq n \leq 100000$

对于全部的数据 $0 \leq a_i \leq 1000$

#2285、专业小偷

从第n个房间开始搜索

对于第$1 \sim i$个房间

- 若选择当前房间,那么必须跳过第$i-1$个房间,只能考虑前$i-2$个房间
- 若不选择当前房间,只能考虑前$i-1$个房间

第$1 \sim i$个房间的答案为两种策略的最大值

朴素搜索时间复杂度为指数级

不难发现其中有大量重复状态,考虑记忆化搜索

记忆化数组每个状态的未出现时默认值应当选择一个**绝不可能出现的**值

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, a[100005], mem[100005];
int dfs(int cur)
{
  if (cur < 0)
    return 0;
  if (~mem[cur])
    return mem[cur];
  int ll = dfs(cur - 2), l = dfs(cur - 1);
  return mem[cur] = max(ll + a[cur], l);
}

int main()
{
  memset(mem, -1, sizeof mem);
  scanf("%d", &n);
  for (int i = 1; i <= n; i++)
    scanf("%d", a + i);
  printf("%d", dfs(n));
  return 0;
}
```

#2285、专业小偷

看上去也可以直接按顺序动态规划

设 f_i 表示考虑了房屋 $1 \sim i$ 的最优方案。现在决策 i 是否偷:

- 偷,则 $f_i = \max(f_i, f_{i-2} + a[i])$
- 不偷,则 $f_i = \max(f_i, f_{i-1})$

是不是所有记忆化搜索都可以按某种顺序进行动态规划?

那为什么还要记忆化搜索呢?有些时候动态规划的顺序确定起来很麻烦

#1288、滑雪

【题目描述】

小明喜欢滑雪，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，当小明滑到坡底，不得不再次走上坡或等着直升机来载他，小明想知道在一个区域中最长的滑坡。滑坡的长度由滑过点的个数来计算，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，一条可行的滑坡为 $25-24-17-16-1$（从 25 开始到 1 结束），当然 $25-24\ldots2-1$ 更长，事实上这是最长的一条。

【输入】

输入的第一行为表示区域的二维数组的行数 R 和列数 C（$1 \leq R$、$C \leq 100$），下面是 R 行，每行有 C 个数代表高度。

【输出】

输出区域中最长的滑坡长度。

【输入样例】

```
5 5
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
```

【输出样例】

```
25
```

#1288、滑雪

尝试枚举一个点作为起点进行搜索

对于当前位置(x,y)枚举四个方向若能移动则移动,时间复杂度指数级

不难发现(x,y)无法回到到达它的位置(走回头路)

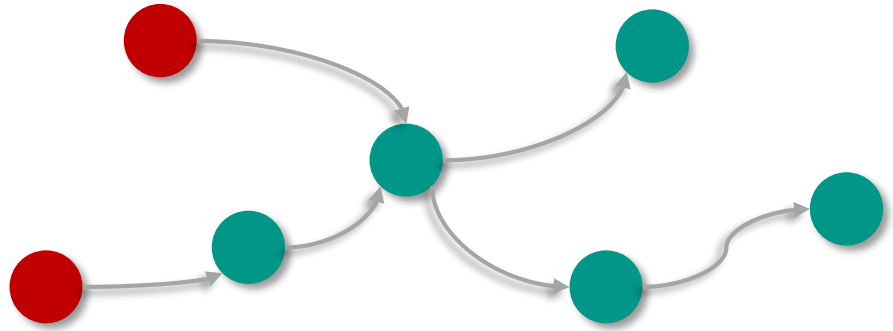
之前的路径选择并不会影响接下来的路径选择

即对于(x,y)的子问题存在重复搜索

记忆化搜索即可

由于每个格子至多被搜索一次,时间复杂度$O(nm)$

强行想按顺序动态规划也可行,但是首先得对高度排序



```
int dfs(int x, int y)
{
  if (mem[x][y])
    return mem[x][y];
  int res = 1;
  for (int i = 0; i < 4; i++)
  {
    int tx = x + dir[i][0], ty = y + dir[i][1];
    if (tx >= 0 && tx < r && ty >= 0 && ty < c && a[x][y] > a[tx][ty])
      res = max(res, dfs(tx, ty) + 1);
  }
  return mem[x][y] = res;
}
```

谢谢观看