

## 第十次作业

### P1255 数楼梯

```
N=int(input())

dp=[0]*(N+1)
dp[0]=1
for i in range(1,N+1):
    if i==1:
        dp[i]=1
    else:
        dp[i]=dp[i-1]+dp[i-2]

print(dp[N])
```



所属题目

P1255 数楼梯

评测状态

Accepted

评测分数

100

提交时间

2024-11-27 15:03:05

### OJ27528: 跳台阶

```
N=int(input())

dp=[0]*(N+1)
dp[0]=1
for i in range(1,N+1):
    if i==1:
        dp[i]=1
    else:
        dp[i]=sum(dp)

print(dp[N])
```

比赛	题目	结果	内存	时间	代码长度	语言	提交时间
 题库 (包括计概、数算题目)	27528: 跳台阶	Accepted	3628kB	35ms	134 B	Python3	刚刚

CF474D: flowers

```
t,k=map(int,input().split())
```

```
MAXN=100001
```

```
dp=[0]*MAXN
```

```
dp[0]=1
```

```
sum_flower=[0]*MAXN
```

```
MOD = 10**9+7
```

```
for i in range(1,MAXN):
```

```
    if i<k:
```

```
        dp[i]=dp[i-1]
```

```
    else:
```


```
        dp[i]=(dp[i-1]+dp[i-k])% MOD
```

```
    sum_flower[i]=(sum_flower[i-1]+dp[i])% MOD
```

```
for _ in range(t):
```

```
    a, b = map(int, input().split())
```

```
    print((sum_flower[b]-sum_flower[a-1]+MOD)% MOD)
```

Contest status 							
#	When	Who	Problem	Lang	Verdict	Time	Memory
293585449	Nov/28/2024 04:07 UTC	liquorice1liquorice	<a href="#">474D - Flowers</a>	Python 3	Accepted	671 ms	7400 KB

最初始的代码是下面这样的，后来因为 test 3 超时，问了 AI，AI 提供了解决方式如下。同时也学会了 MOD 的用法。

```
t,k=map(int,input().split())
```

```
for _ in range(t):
```

```
    a,b=map(int,input().split())
```

```
    dp=[0]*(b+1)
```

```
    dp[0]=1
```

```
    for i in range(1,b+1):
```

```
        if i<k:
```

```
            dp[i]=1
```

```
        else:
```

```
            dp[i]=dp[i-1]+i//k
```

```
    print(sum(dp[a:b+1]))
```

(以下为找规律时写的)

# $i < k$  时,  $dp[i] == 1$ ;

# $i == k$  时,  $dp[i] == 1 + 1 = 2$ ;

# $k < i < 2k$  时,  $dp[i] = 1 + (1 + (i - k))$

# $i == 2k$  时,  $dp[i] == 1 + 1 + (1 + (i - k))$

# $2k < i < 3k$  时,  $dp[i] = 1 + (1 + (i - 2k)) + (1 + (i - k))$

这段代码的性能瓶颈主要在于两个方面:

1. 每次查询都重新计算 dp 数组:

dp 数组的生成过程是按顺序从 1 到 b 计算, 每次都重复计算, 特别是当 t 很大或者 b 值较大时这种重复会导致超时。

2. 求和操作重复执行:

`sum(dp[a:b+1])` 每次需要从 dp 中切片并累加, 同样增加了计算复杂度。

优化方法:

可以通过以下方式改进:

1. 预计算 dp 数组并复用:

- 首先在代码开头预先计算出足够大的 dp 数组, 只需计算到所有查询中最大的 b 即可。
- 这样在处理每次查询时, 直接用已经计算好的 dp 数组即可, 避免重复计算。

2. 构造前缀和数组:

- 使用一个前缀和数组 prefix, 使得 `prefix[i] = dp[0] + dp[1] + ... + dp[i]`。
- 对于区间和 `sum(dp[a:b+1])`, 可以通过 `prefix[b] - prefix[a-1]` 快速获得, 时间复杂度降为  $O(1)$ 。

## LeetCode5.最长回文子串

class Solution:

```
def longestPalindrome(self, s: str) -> str:
```

```
    n=len(s)
```

```
    if n==0:
```

```
        return ""
```

```
    dp=[[False]*n for _ in range(n)]
```

```
    start,max_length=0,1
```

```
    for i in range(n):
```

```
        dp[i][i]=True
```

```
    for i in range(n-1):
```

```
        if s[i]==s[i+1]:
```

```
            dp[i][i+1]=True
```

```
            start=i
```

```
            max_length=2
```

```
    for length in range(3,n-1):
```

```

        for i in range(n-length+1):
            j=i+length-1
            if s[i]==s[j] and dp[i+1][j-1]==True:
                dp[i][j]=True
                start=i
                max_length=length

```

```

    return s[start:start+max_length]

```

☑ 测试用例 | > 测试结果

通过 执行用时: 0 ms

• Case 1 • Case 2

输入

```

s =
"babad"

```

输出

```

"aba"

```

预期结果

```

"bab"

```

OJ12029: 水淹七军

```
import sys
```

```
sys.setrecursionlimit(300000)
```

```
input = sys.stdin.read
```

```

def is_valid(x,y,m,n):
    return 0<=x<m and 0<=y<n

```

```

def dfs(x, y, water_height_value, m, n, h, water_height):
    directions=[(1,0),(-1,0),(0,1),(0,-1)]
    for dx,dy in directions:
        nx=x+dx
        ny=y+dy
        if is_valid(nx,ny,m,n) and h[nx][ny]<water_height_value:
            if water_height[nx][ny]<water_height_value:
                water_height[x][y]=water_height_value
                dfs(nx, ny, water_height_value, m, n, h, water_height)

```

```

def main():
    data = input().split() # 快速读取所有输入数据
    idx = 0

```

```

k = int(data[idx])
idx += 1
results = []

for _ in range(k):
    m, n = map(int, data[idx:idx + 2])
    idx += 2
    h = []
    for i in range(m):
        h.append(list(map(int, data[idx:idx + n])))
        idx += n
    water_height = [[0] * n for _ in range(m)]

    i, j = map(int, data[idx:idx + 2])
    idx += 2
    i, j = i - 1, j - 1

    p = int(data[idx])
    idx += 1

    for _ in range(p):
        x, y = map(int, data[idx:idx + 2])
        idx += 2
        x, y = x - 1, y - 1
        if h[x][y] <= h[i][j]:
            continue

        dfs(x, y, h[x][y], m, n, h, water_height)
        #water_height_value 最初是由主函数中 h[x][y] 的值来决定的，特别是在第一次调用 dfs 时。
        #当 dfs 被递归调用时，water_height_value 会一直沿着水流扩展的路径传递，并且它保持不变，直到递归到一个新的点。如果水可以流到该点，它会更新该点的水位，并继续尝试扩展。
        #每一次 water_height_value 都是递归开始的那个点的高度，并且它在递归过程中不会改变，直到递归进入新的点时才会由新的点的高度决定。
        results.append("Yes" if water_height[i][j] > 0 else "No")
    sys.stdout.write("\n".join(results) + "\n")

if __name__ == "__main__":
    main()

```

比赛	题目	结果	内存	时间	代码长度	语言	提交时间
 题库 (包括计概、数算题目)	12029: 水淹七军	Accepted	14352kB	269ms	2016 B	Python3	刚刚

这个题太难了不会做，是看着答案一点一点思考的，但是让我学会了一次性输入所有数据的逻辑

OJ02802: 小游戏

```
import heapq
num1=1
while True:
    w,h=map(int,input().split())
    if w==0 and h==0:
        break
    print(f"Board #{num1}:")
    martix=[[" "]*(w+2)]+[[[" "]+list(input())+[" "] for _ in range(h)]+[[[" "]*(w+2)]]
    dir=[(0,1),(0,-1),(1,0),(-1,0)]
    num2=1
    while True:
        x1,y1,x2,y2=map(int,input().split())
        if x1==0 and x2==0 and y1==0 and y2==0:
            break
        queue,flag=[],False
        vis=set()
        heapq.heappush(queue,(0,x1,y1,-1))
        martix[y2][x2]=" "
        vis.add((-1,x1,y1))
        while queue:
            step,x,y,dirs=heapq.heappop(queue)
            if x==x2 and y==y2:
                flag=True
                break
            for i,(dx,dy) in enumerate(dir):
                px,py=x+dx,y+dy
                if 0<=px<=w+1 and 0<=py<=h+1 and (i,px,py) not in vis and
martix[py][px]!="X":
                    vis.add((i,px,py))
                    heapq.heappush(queue,(step+(dirs!=i),px,py,i))
        if flag:
            print(f"Pair {num2}: {step} segments.")
        else:
            print(f"Pair {num2}: impossible.")
        martix[y2][x2]="X"
```

```
num2+=1
print()
num1+=1
```

比赛	题目	结果	内存	时间	代码长度	语言	提交时间
 题库 (包括计概、数算题目)	02802: 小游戏	Accepted	4808kB	78ms	1155 B	Python3	刚刚

确实难，根据标答又复习了一下 `heapq` 的用法。

学习感悟：前两道题简单，但是让我把 `dp` 的最基本的思路和逻辑又复习了一遍，我感觉非常好。`Flower` 那个题也是自己想出来的，只不过可能花的时间比较多 `and` 不懂如何节约时间，所以又学到了可以构造前缀和数组。之后的每道题虽然都很难是看着标答做的，但每道题也都有学到一些东西！