**沈思妤-10235501458-数据科学导论第二次作业**

**复习题：**

1. **列举你认为是数据思维的实例。**

答：蒙特卡罗方法通过随机抽样来估算圆周率。

核心原理是利用点的数量来代表面积：在正方形内随机生成点时，点的数量与面积成正比。在正方形内，每个点的生成都是独立的随机事件，假设我们生成N个点。在正方形的面积为4的情况下，圆的面积为π。通过计算落在圆内的点的数量，我们可以得出圆内点数与总点数的比例，从而估算π。Python中，可以使用简单的循环和随机数生成来实现这一算法。

1. **递归法与分治法背后的思想是怎样的?**

答：递归法与分治法背后的思想是计算思维。

计算思维，也称构造思维，是一种通过设计和实现算法来解决问题的思维方法。它的核心在于将问题构造成计算机能够理解的模型或算法，通过编程描述模型并执行，最终得出解决方案。这一过程类似于“已知、求解”的逻辑推理，强调从实践到理论，从问题到规律的总结与应用。

递归法通过将问题拆解为更小的相似子问题，不断调用自身，直到达到可以直接解决的情况。而分治法则将问题分解成独立的部分，分别求解后再合并结果。这两种方法都通过分解问题、逐步求解，反映了计算思维中系统化解决问题的核心思想。

**践习题：**

1. **给定一个正整数n,找出一个正整数列表,它们的乘积是所有和为 n 的正整数列表中最大的。例如,如果 n 为 4,那么要求的正整数列就是 2,2,因为 2X2 的结果比1X1X1X1、2X1X1 以及3X1 的结果都要大。再比如,如果 n 为 5,那么所求的正整数列就是2,3。**
2. **如果 n=2 001,所求的正整数列是什么?**

答：n=2001时，因为2001除以3得667，所以得到的正整数列是667个3。

1. **解释是如何迈出第一步的。**

答：对于任意两个正整数x和y，当x+y固定时，xy的最大化发生在x和y的值接近时。通过计算以及数学知识可得，当我们将一个数分解为多个因子时，保持因子的数量尽可能多且接近相等的值，会得到最大的乘积。通过观察，可以发现3是最优的因子。如果n除以3的余数为0，使用全3是最好的选择；如果余数为1，最好用一个4（2+2代替1+3）；如果余数为2，可以直接使用2。

**(3)用 Python 实现所设计的算法。**

答：def max(n):

if n < 4:

return [n]

result = []

while n > 4:

result.append(3)

n -= 3

if n == 2:

result.append(2)

elif n == 4:

result.extend([2, 2])

return result

n = int(input())

print(max(n))

1. **用 Python 计算 2的10 次方、20 次方、30 次方、40 次方和 50 次方,观察所得结果是不是增长得很快?(用 2 \*\* 10 的语句可计算出 2 的 10 次方的值)**

答：for i in [10, 20, 30, 40, 50]:

print(f"2\*\*{i} = {2 \*\* i}")

运行得到：

2\*\*10 = 1024

2\*\*20 = 1048576

2\*\*30 = 1073741824

2\*\*40 = 1099511627776

2\*\*50 = 1125899906842624

可以观察到所的结果增长得很快。

1. **用Python 解决本章 2.2.1中的渡河问题,生成所有的可行方案。**

答：name = ["人", "狼", "羊", "菜"]

count = 0

def done(status):

return status[0] and status[1] and status[2] and status[3]

def all\_status(status):

nextlist = []

for i in range(0, 4):

if status[0] != status[i]:

continue

next\_status = [status[0], status[1], status[2], status[3]]

next\_status[0] = not next\_status[0]

next\_status[i] = next\_status[0]

if valid(next\_status):

nextlist.append((next\_status, i))

return nextlist

def valid(status):

if status[1] == status[2] and status[0] != status[1]:

return False

if status[2] == status[3] and status[0] != status[2]:

return False

return True

def search(old\_status, actions):

global count

current\_status = old\_status[-1]

nextlist = all\_status(current\_status)

for next\_status, action in nextlist:

if next\_status in old\_status:

continue

old\_status.append(next\_status)

if action > 0:

actions.append(f"{name[0]}带{name[action]}过去" if action > 0 else f"{name[0]}自己过去")

else:

actions.append(f"{name[0]}自己过去")

if done(next\_status):

count += 1

print(f"方案{count}:")

print("\n".join(actions))

print()

else:

search(old\_status, actions)

old\_status.pop()

actions.pop()

def main():

global count

status = [False, False, False, False]

old\_status = [status]

actions = []

search(old\_status, actions)

print("找到" + str(count) + "个方案")

if \_\_name\_\_ == "\_\_main\_\_":

main()

运行得到：

方案1:

人带羊过去

人自己过去

人带狼过去

人带羊过去

人带菜过去

人自己过去

人带羊过去

方案2:

人带羊过去

人自己过去

人带菜过去

人带羊过去

人带狼过去

人自己过去

人带羊过去

找到2个方案

**4. 用 while 循环实现“笨办法”求解根号2,使得一旦找到所要的 g 就跳出循环。这样可以减少不必要的循环次数。**

答：n = 2.0

g = 1.0

a = 0.001

while True:

g2 = g \* g

if abs(g2 - n) < a:

break

g += a

print("根号2约等于", g)

得到输出根号2约等于 1.4139999999999544。