《算法分析基础》上机实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 实验三 分治算法实例 | | | 实验时间 | 2022.3.17 |
| 学生姓名 | 胡恒宇 | 班级 | 计z1912 | 学号 | 19200135221 |
| 指导教师 | 陆悠 | 批阅教师 | 陆悠 | 成绩 |  |
| **一、实验目的**  1. 了解分治策略算法思想及基本原理；  2. 掌握使用分治算法求解问题的一般特征；  3. 掌握分解、治理的方法；  4. 能够针对实际问题，能够正确的分解、治理，设计分治算法；  5. 能够正确分析算法的时间复杂度和空间复杂度。  **二、实验任务**  下列2个问题中任选一个完成  1、棋盘覆盖问题：给定一个2k×2k的棋盘(具体图例见教材)，有一个特殊棋格，拥有一个特殊棋格的棋盘称为特殊棋盘。现要用四种L型骨牌(具体图例见教材)覆盖特殊棋盘上除特殊棋格外的全部棋格，不能重叠，找出覆盖方案。  2、邮局选址问题：对一个以XY轴表达的城市里，n个居民点散乱分布，居民点位置可以用坐标(x,y)表示，任意两点(x1,y1)(x2,y2)间距离可以用| x1-x1|+|y1-y2|度量，居民希望在城市中选择建立邮局的最佳位置，使得n个居民点到邮局的距离综合最小  3\*、扩展内容  （1）请在查询资料基础上，实现递归和分治思路的汉诺塔问题、整数划分问题。  （2）大整数乘法问题：给定两个n位的大整数A、B，求A与B的乘积。  （3）最小值问题：求n个元素的最小值。  （4）幂乘问题：给定实数a和自然数n，求an。  **三、审题结果**  1、棋盘覆盖问题：  已知条件：如图1所示，左侧为k=2的棋盘，盘中黑棋为特殊棋格；右侧为四种可用的L型骨牌。    图1  求解目标：如图2所示，以L骨牌覆盖特殊棋格外所有棋格，不可重叠。    图2  **四、问题建模结果**  当k>0时，将2k\*2k棋盘分割为4个2(k-1)\*2(k-1)子棋盘，如图3所示：    图3    特殊方格必定位于这四个小棋盘中，其余三个子棋盘没有特殊方格，为了将这三个无特殊方格的子棋盘转换为特殊棋盘，我们可以用一个L型骨盘覆盖这三个较小棋盘的会合处，如4图所示：    图4  从图5可以看出，这三个子棋盘上被L型骨牌覆盖的方格就成为该棋盘上的特殊方格，从而将问题分解为4个较小规模的棋盘覆盖问题。递归地使用这种分割方法，直至棋盘简化为1\*1棋盘，就结束递归。    图5  时间复杂度分析：设T(k)为覆盖2k\*2k棋盘的时间，  当k=0时，覆盖时间为O(1);  当k>0时，测试哪个子棋盘特殊以及形成3个特殊子棋盘需要O(1),覆盖4个特殊子棋盘需要四次递归调用，更需4T(k-1);  =>T(k)=O(4^k)  **五、算法实现部分**  1、棋盘覆盖问题：  **def** chess(board, tr, tc, pr, pc, size): *# tr,tc：棋盘左上角的位置，即棋盘位置。pr,pc:特殊方格的位置，size为棋盘大小。* **global** mark  *# global table* **if** size == 1:  **return** mark = mark + 1  count = mark  half = size // 2  **if** pr < tr + half **and** pc < tc + half:  chess(board,tr, tc, pr, pc, half)  **else**:  board[tr + half - 1][tc + half - 1] = count  chess(board, tr, tc, tr + half - 1, tc + half - 1, half)  **if** pr < tr + half **and** pc >= tc + half:  chess(board, tr, tc + half, pr, pc, half)  **else**:  board[tr + half - 1][tc + half] = count  chess(board, tr, tc + half, tr + half - 1, tc + half, half)  **if** pr >= tr + half **and** pc < tc + half:  chess(board, tr + half, tc, pr, pc, half)  **else**:  board[tr + half][tc + half - 1] = count  chess(board, tr + half, tc, tr + half, tc + half - 1, half)  **if** pr >= tr + half **and** pc >= tc + half:  chess(board, tr + half, tc + half, pr, pc, half)  **else**:  board[tr + half][tc + half] = count  chess(board, tr + half, tc + half, tr + half, tc + half, half)  **def** Print(board):  **for** i **in** range(len(board)):  **for** j **in** range(len(board[i])):  print(board[i][j], end = **' '**)  print()  mark = 0 **if** \_\_name\_\_ == **'\_\_main\_\_'**:  k = int(input().strip())  s = 1  **for** i **in** range(k):  s \*= 2  line = input().strip().split()  x = int(line[0]) - 1  y = int(line[1]) - 1  board = [[0 **for** i **in** range(s)] **for** j **in** range(s)]  board[x][y] = -1  chess(board, 0, 0, x, y, len(board))  Print(board)  2、幂乘问题：  **def** power(a,n):  **if** a == 1:  **return** 1  **if** a == 0:  **return** 0  **if** n == 1:  **return** a  **if** n % 2 == 0:  b = power(a,n//2)  **return** b\*b  **else**:  b = power(a,(n-1)//2)  **return** b \* b \* a  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  a=int(input(**"a="**).strip())  n=int(input(**"n="**).strip())  res=power(a,n)  print(**"a^n ="**,res)  **六、实验结果分析**  1、棋盘覆盖问题      2、幂乘问题：        **七、实验总结**  通过本次实验，进一步加深对分治算法的理解，学会将一个规模为n的问题分解为原理相同的k个规模较小的子问题，递归的解决子问题，然后将子问题的解合并到原问题，简单地说，就是将规模为n的问题自顶向下分解，直到子问题分解到足够小，可以容易解决时，再自底向上合并，从而得到原来的解。 | | | | | |