**《算法设计与分析》实验报告**

实验名称 ： 实验1 递归与分治 1

实验日期 ： 　 2025.3.4

姓 名 ： 　　高心阳

学 号 ： 　 084623237

班 级 ： 　 计算机232

成 绩 ：

**人工智能与信息技术学院**

**南京中医药大学**

|  |
| --- |
| **实验目的：** |
| 1. 熟悉递归与分治的概念 2. 掌握递归与分治的使用 |
| **实验内容和要求** |
| 1、设计一个程序执行快速排序法。  快速排序是最经常使用的一种排序方式，对于给定的n个数组成的一个数组，请使用快速排序对其进行排序。对于这道题，你只需要实现一个用于排序的函数qsort，函数的头部如下：  　　void qsort(int array[], int size)  其中array是待排序的数组，size是它的大小，注意数组中的元素是从0开始编号的，也就是说你要排序的元素是array[0..size-1]。  可以加入其他的辅助函数，在main函数中对qsort函数加以调用验证。  输入样例  65 70 75 80 85 60 55 50 45  输出样例  45 50 55 60 65 70 75 80 85  2、士兵站队问题  在一个划分成网格的操场上，n个士兵散乱地站在网格上，网格点由整数坐标(x,y)表示，士兵们可以沿网格边上、下、左、右移动，移动一格算一步。按照军官的命令，士兵们要整齐的列成一格水平队列，即排成(x,y),(x+1,y),....(x+n-1,y)。  问题是：如何选择x和y的值使得士兵们以最少的移动步数排成一列。  问题分析：  士兵有多种移动方式：通过适当的方法使得同一时刻不会有两名士兵站在同一点。  题目要求最少移动步数：转化为求士兵站立的"最终位置"，即如何取各个士兵站立的行列使得士兵移动的步数最少。  解题思路：  将X轴和Y轴分开来看：  考虑Y轴方向：  （1）设Y轴目标坐标为M，即n个士兵最终需要移动到Y轴的坐标为M。  设n个士兵初始位置的Y轴坐标分别为：Y0,Y1,Y2...Yn-1，则最优步数为S=|Y0-M|+|Y1-M|+...+|Yn-1-M|。  M取中间点的值可以使得S达到最优，也就是处于中间点位置士兵的Y轴坐标就是"最终位置"的Y轴坐标。  可以对所有Y轴最表进行排序，然后取"中间"点的Y轴坐标作为最佳位置M的值，最后通过公式求出Y方向上移动的最优步数。  考虑X轴方向：  设n个士兵初始位置的X轴坐标分别为：X0,X1,X2...Xn-1，n个士兵最终需要移动到X轴的坐标分别为N,N+1，N+2...N+（n-1）。则最优步数为S=|X0-N|+|X1-N-1|+...+|Xn-1-N-n+1|。  X轴方向与Y轴方向考虑一致，同样是n个已知数分别减去一个待定数后取绝对值，然后求和。因此还是采用取中位数的方法求解最优解。  算法实现：  用快速排序算法对X轴和Y轴坐标进行排序。  分别找出X轴和Y轴坐标的第(n+1)/2个元素，即中位数。  分别计算X轴方向和Y轴方向移动的步数，然后将这两个数值相加就是所求的最小步数。  样例结果：  请输入士兵数目：  5  请输入5个士兵的坐标位置：  1 2  2 2  1 3  3 -2  3 3  使士兵排成一行所要移动的最少步数为：  8  \*3、题目描述  阴天傍晚车窗外  未来有一个人在等待  向左向右向前看  爱要拐几个弯才来  我遇见谁会有怎样的对白  我等的人他在多远的未来  我听见风来自地铁和人海  我排着队拿着爱的号码牌  ——「遇见」孙燕姿  城市中人们总是拿着号码牌，不停寻找，不断匹配，可是谁也不知道自己等的那个人是谁。  可是燕姿不一样，燕姿知道自己等的人是谁，因为燕姿数学学得好！燕姿发现了一个神奇的算法：假设自己的号码牌上写着数字S，那么自己等的人手上的号码牌数字的所有正约数之和必定等于S。  所以燕姿总是拿着号码牌在地铁和人海找数字（喂！这样真的靠谱吗）可是她忙着唱《绿光》，想拜托你写一个程序能够快速地找到所有自己等的人。 输入描述: 输入包含k组数据。  对于每组数据，输入包含一个号码牌S。 输出描述: 对于每组数据，输出有两行，第一行包含一个整数m，表示有m个等的人。第二行包含相应的m个数，表示所有等的人的号码牌。  注意：你输出的号码牌必须按照升序排列。  示例1  输入  1  42  输出  3  20 26 41  备注: k<=100 且 S<=2\*109 |
| **运行结果（写清题号）** |
| 1.45 50 55 60 65 70 75 80 85  #include <iostream>  using namespace std;  void qsort(int \*array, int size) {  if (size <= 1)return;  int head = 0, tail = size - 1;  int mid = array[size / 2];  while (head <= tail) {  while (array[head] < mid)head++;  while (array[tail] > mid)tail--;  if (head <= tail) {  int temp = array[head];  array[head] = array[tail];  array[tail] = temp;  head++;  tail--;  }  }  qsort(array,tail+1);  qsort(array+head,size-head);  }  int main() {  int arr[9] = {65, 70, 75, 80, 85, 60, 55, 50, 45};  qsort(arr, 9);  for (int i = 0; i < 9; ++i)  cout << arr[i] << " ";  return 0;  }  2.8  #include <iostream>  #include <cmath>  #include <vector>  using namespace std;  void solve() {  int n;  cin >> n;  vector<int> x(n);  vector<int> y(n);  for (int i = 0; i < n; i++)  cin >> x[i] >> y[i];  sort(x.begin(), x.end());  sort(y.begin(), y.end());  for (int i = 0; i < n; i++)  x[i] = x[i] - i;  sort(x.begin(), x.end());  int xmid = x[n / 2];  int ymid = y[n / 2];  int xsum = 0,ysum = 0;  for (int i = 0; i < n; i++) {  xsum += abs(x[i] - xmid);  ysum += abs(y[i] - ymid);  }  cout << xsum + ysum << endl;  }  int main() {  solve();  return 0;  }  3.3  20 26 41  #include <iostream>  #include <cmath>  #include <vector>  using namespace std;  int Num;  int sum(int num) {  int sum = 0;  for (int i = 1; i < sqrt(num); ++i)  {  if (num % i == 0)sum += i + num / i;  if (sum > Num)return -1;  }  return sum;  }  void solve() {  std::vector<int> v;  cin >> Num;  for (int i = 1; i < Num; i++) {  if (sum(i) == Num)  v.push\_back(i);  }  cout << v.size() << endl;  for (int i = 0; i < v.size(); i++)  {  cout << v[i] << " ";  }  }  int main() {  int T;  cin >> T;  while (T--) {  solve();  }  return 0;  } |
| **实验的体会与建议** |
|  |