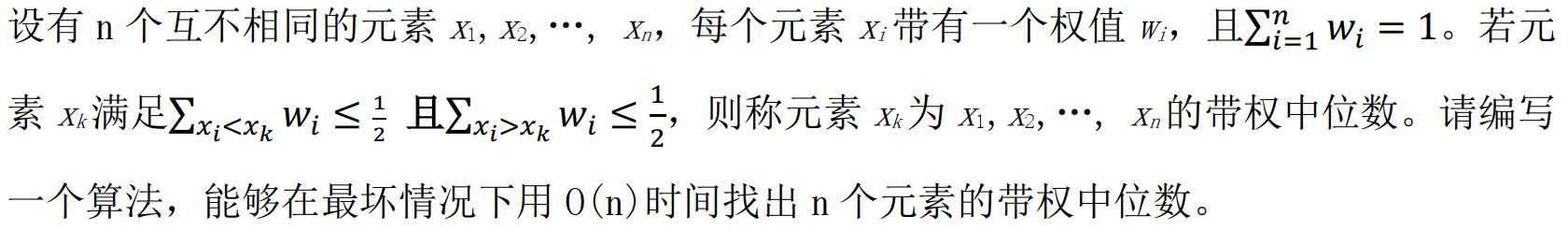
西安交通大学实验报告

课程名称：算法设计与分析 实验名称： 分治与递归

学 院：\_\_\_\_\_\_\_电信学部\_\_\_\_\_\_\_\_\_\_\_\_ 实 验 日 期 2022年4 月 24 日

|  |  |  |
| --- | --- | --- |
| **刘沁宇** | **计算机002** | **2203613019** |

#### 问题描述



#### 问题分析

该题需要求在O(n)的时间复杂度下找出中位数，则不能采取暴力算法，则可基于快速排序原理，利用随机划分，将问题规模缩小，再结合带权中位数的定义即可得到结果。

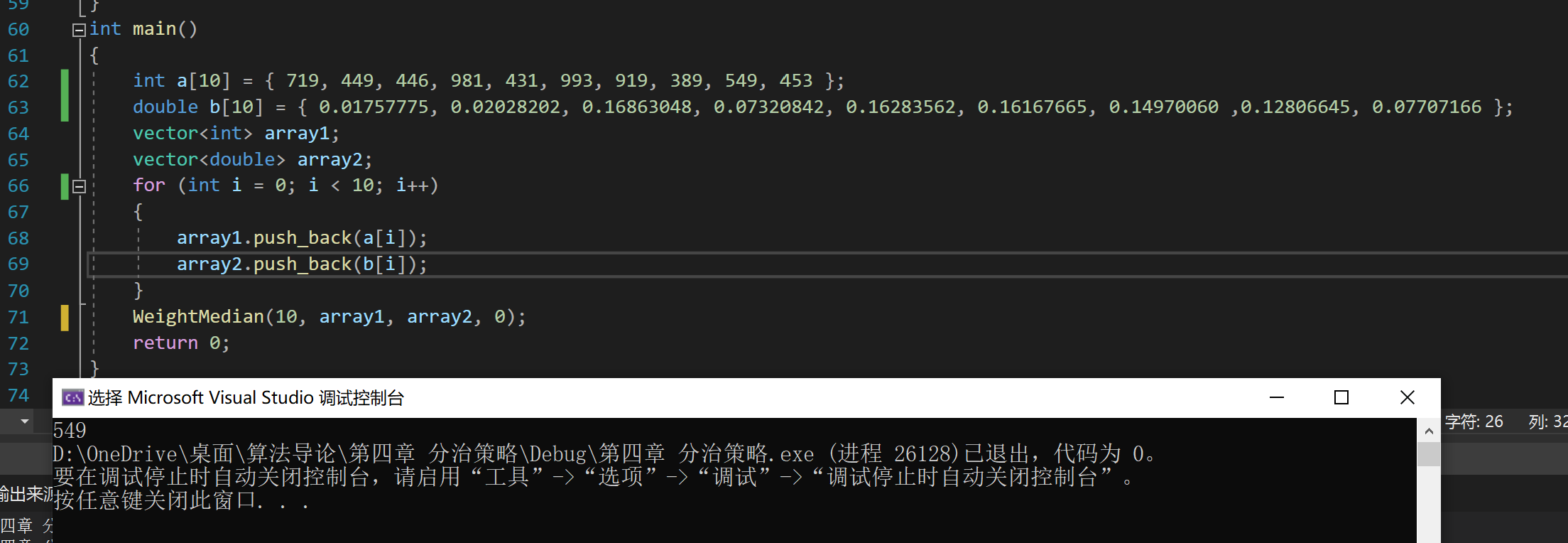
#### 算法设计

利用快速排序中的partition函数得到列表的一个划分和一个划分点pos，若pos左侧列表的权值之和与右侧列表的权值之和均小于 ,则返回pos点对应的值，若左侧权值大于1/2，则在pos左侧列表再利用partition函数继续划分，否则在pos右侧得到一个划分。

#### 算法实现

|  |
| --- |
| 1. #include <iostream> 2. #include <string> 3. #include <cmath> 4. #include <vector> 5. #include <algorithm> 6. **using** **namespace** std; 7. **int** partition(**int** left, **int** right , vector<**int**> &num, vector<**double**> &weight)          //要修改vector的值，因此采用引用传递！！！ 8. { 9. **if** (left == right) 10. **return** left; 11. **int** value = num[right] ; 12. **int** i, j=left; 13. **for** (i = left; i < right; i++) 14. { 15. **if** (num[i] < value) 16. { 17. swap(num[i], num[j]); 18. swap(weight[i], weight[j]); 19. j++; 20. } 21. } 22. swap(num[j], num[right]); 23. swap(weight[j], weight[right]); 24. **return** j; 25. } 27. **double** SumWeight(vector<**double**> weight, **int** left, **int** right)      //这里是返回double型，不是int型 28. { 29. **double** sum = 0; 30. **for** (**int** i = left; i <= right; i++) 31. sum += weight[i]; 32. **return** sum; 33. } 35. **int** Get\_weightMidNum(**int** left, **int** right,vector<**int**> num, vector<**double**> weight) 36. { 37. **if** (left == right) 38. **return** num[left]; 39. **if** ((right - left) == 1) 40. { 41. **if** (weight[left] >= weight[right]) 42. **return** num[left]; 43. **else** 44. **return** num[right]; 45. } 46. **int** pivot = partition(left, right, num, weight); 47. **double**  LW = SumWeight(weight, 0, pivot - 1),          //左权值和 48. RW = 1 - LW - weight[pivot];          //右权值之和 49. **if** (LW < 0.5 && RW <= 0.5)                    //当主元的左列表权值和和右列表的权值和均小于0.5时 50. **return** num[pivot]; 51. **else** **if** (LW > 0.5 && RW <= 0.5) 52. **return**(Get\_weightMidNum(left, pivot - 1, num, weight)); 53. **else** 54. **return**(Get\_weightMidNum(pivot + 1, right, num, weight)); 55. } 56. **void** WeightMedian(**int** length, vector<**int**>num, vector<**double**>weight, **int** index) 57. { 58. cout << Get\_weightMidNum(index, length + index - 1, num, weight); 59. } 60. **int** main() 61. { 62. **int** a[10] = { 719, 449, 446, 981, 431, 993, 919, 389, 549, 453 }; 63. **double** b[10] = { 0.01757775, 0.02028202, 0.16863048, 0.07320842, 0.16283562, 0.16167665, 0.14970060 ,0.12806645, 0.07707166 }; 64. vector<**int**> array1; 65. vector<**double**> array2; 66. **for** (**int** i = 0; i < 10; i++) 67. { 68. array1.push\_back(a[i]); 69. array2.push\_back(b[i]); 70. } 71. WeightMedian(10, array1, array2, 0); 72. **return** 0; 73. } |

#### 实验结果



图表 1测试用例为题中用例

