河北工程大学

信息与电气工程学院

**实 验 报 告**

|  |  |
| --- | --- |
| 学期 | 2024/2025学年 第1学期 |
| 课程编号及名称 | F095042131 算法分析与设计 |
| 学生学号及姓名 | 222030615郭文钊 |
| 任课教师 | 池静 |
| 提交时间 | 2024.11 |

分治法求最大子段和

1. 相关知识点

对于一个规模为n的问题：若问题规模足够小则可以直接解决，否则将其分解为k个规模较小的子问题，这些子问题互相独立且与原问题形式相同，分别求解这些子问题（用递归或循环），然后将各子问题的解合并得到原问题的解。这种算法设计策略叫做分治法。

1. 实验目的

本实验的目的是让学生巩固下列知识点：

分治法算法的基本思想；

分治法求解问题的一般步骤。

1. 实验内容及要求

①采用简单算法(时间复杂性为O(n3))，实现求最大子段和，并统计运行时间。

②采用改进算法(时间复杂性为O(n2))，实现求最大子段和，并统计运行时间。

③采用分治算法(时间复杂性为O(nlogn))，实现求最大子段和，并统计运行时间。

编写程序，分别采用上述三种算法实现求一个整数序列的最大子段和。要求程序能正确运行，可读性好，具有较好交互性和容错性。

1. 程序及运行结果：

程序1：

int maxCrossingSum\_n3(int *arr*[], int *length*) {

    int res = 0;

    for (int i = 0; i < *length*; i++) {

        for (int m = i; m < *length*; m++) {

            int sum = 0;

            for (int n = i; n <= m; n++) {

                sum += *arr*[n];

            }

            res = res > sum ? res : sum;

        }

    }

    return res;

}



程序2：

int maxCrossingSum\_n2(int *arr*[], int *length*) {

    int res = 0;

    for (int i = 0; i < *length*; i++) {

        int sum = 0;

        for (int j = i; j < *length*; j++) {

            sum += *arr*[j];

            res = res > sum ? res : sum;

        }

    }

    return res;

}



程序3：

int maxCrossingSum(int *arr*[], int *left*, int *mid*, int *right*) {

    int left\_sum = INT\_MIN;

    int sum = 0;

    for (int i = *mid*; i >= *left*; i--) {

        sum += *arr*[i];

        if (sum > left\_sum)

            left\_sum = sum;

    }

    int right\_sum = INT\_MIN;

    sum = 0;

    for (int i = *mid* + 1; i <= *right*; i++) {

        sum += *arr*[i];

        if (sum > right\_sum)

            right\_sum = sum;

    }

    return left\_sum + right\_sum;

}

int maxCrossingSum\_nlogn(int *arr*[], int *left*, int *right*) {

    if (*left* == *right*)

        return *arr*[*left*];

    int mid = (*left* + *right*) / 2;

    int left\_max = maxCrossingSum\_nlogn(*arr*, *left*, mid);

    int right\_max = maxCrossingSum\_nlogn(*arr*, mid + 1, *right*);

    int cross\_max = maxCrossingSum(*arr*, *left*, mid, *right*);

    return max(max(left\_max,right\_max),cross\_max);

}

