# 实验一 分治法基本应用

项目编号： 001

实验学时：2课时

实验类型：学习与设计

**一、实验目的**

了解分治法的基本思想。

1. **实验内容**

归并排序算法分析及实现。

用分治法实现寻找最大值。

**四、实验用主要仪器设备、消耗品**

|  |  |
| --- | --- |
| 仪器设备名称 | 规格 |
| 微机 | **HP** |
| 操作系统 | **win10** |
| 软件 | vscode |

**五、实验步骤**

**1.归并排序**

#include<stdio.h>

#include<stdlib.h>

typedef int ElemType;

// Lpos = left's start, Rpos = right's start, RightEnd = right's end; 后面的变量意思同此处

void Merge(ElemType A[], ElemType TmpArray[], int Lpos, int Rpos, int RightEnd)

{

    int  LeftEnd, numElements, TmpPos;

    LeftEnd = Rpos - 1;

    TmpPos = Lpos;

    numElements = RightEnd - Lpos + 1;//此次合并的数组长度

    while (Lpos <= LeftEnd && Rpos <= RightEnd)

    {

        if(A[Lpos] <= A[Rpos])

            TmpArray[TmpPos++] = A[Lpos++];

        else

            TmpArray[TmpPos++] = A[Rpos++];

    }

    while(Lpos <= LeftEnd)

        TmpArray[TmpPos++] = A[Lpos++];

    while(Rpos <= RightEnd)

        TmpArray[TmpPos++] = A[Rpos++];

    for(int i = 0; i < numElements; i++ ,RightEnd--)//恢复到原数组中

        A[RightEnd] = TmpArray[RightEnd];

}

void Msort(ElemType A[], ElemType TmpArray[], int Left, int Right)

{

    int Center;

    if(Left < Right)

    {

        //递归分解问题

        {

            Center = (Left + Right) / 2;

            Msort(A, TmpArray, Left, Center);

            Msort(A, TmpArray, Center + 1, Right);

            Merge(A, TmpArray, Left, Center + 1, Right);

        }

    }

}

void Mergesort(ElemType A[], int N)

{

    ElemType \*TmpArray = NULL;//用于合并

        TmpArray = malloc(N \* sizeof(ElemType));

    if(TmpArray != NULL)

    {

        Msort(A, TmpArray, 0, N - 1);

        for (int i = 0; i < N; i++)

        {

            printf("%d ",A[i]);

        }

        free(TmpArray);

    }

    else

    {

            printf("error");

    }

}

int main()

{

    int a[6] = {6,3,7,0,9,1};

    Mergesort(a,6);

    return 0;

}

**2.寻找最大值**

#include<stdio.h>

typedef int ElemType;

#define MAX(x,y) (((x)>(y))?(x):(y))

ElemType FindMax(ElemType A[], int Left, int Right, int N)

{

    if(N == 1)

        return A[Left];

    int Center = (Left + Right) / 2;

    return MAX(FindMax(A, Left, Center, Center - Left + 1),FindMax(A, Center + 1, Right, Right - Center));

}

int main()

{

    int a[6] = {6,3,7,0,9,1};

    printf("%d", FindMax(a, 0, 5, 6));

}

**六、实验注意事项及对学生的要求**

1. 模块化编程；注意程序的健壮性。
2. 分治法的应用。

**六、思考题**

1. 递归算法的时间效率怎么分析？

（1）分析程序，找出递推方程。

（2）运用迭代法等方法进行计算求解。