北　京　邮　电　大　学

# 实　　验　　报　　告

课程名称　算法设计与分析

实验名称　求解最大子段和的算法分析

计算机科学与技术专业　2018211302 班
姓名　2018210074 熊宇

教师　王晓茹　　　　　　成绩______

2020 年 9 月 20 日

# 目录

## 一、 源程序

```
#include <iostream>
#include <ctime>

using namespace std;

int testData_1[10];
int testData_2[100];
int testData_3[1000];
int testData_4[10000];
int testData_5[100000];
int testData_6[1000000];
int testData_7[10000000];
int testData_8[100000000];
clock_t start;
clock_t ending;

void InitialTestData()
{
  for (int i = 0;i < 10;i++)
      testData_1[i] = rand() % 40 - 20;
  for (int i = 0;i < 100;i++)
      testData_2[i] = rand() % 40 - 20;
  for (int i = 0;i < 1000;i++)
      testData_3[i] = rand() % 40 - 20;
  for (int i = 0;i < 10000;i++)
      testData_4[i] = rand() % 40 - 20;
  for (int i = 0;i < 100000;i++)
      testData_5[i] = rand() % 40 - 20;
  for (int i = 0;i < 1000000;i++)
      testData_6[i] = rand() % 40 - 20;
  for (int i = 0;i < 10000000;i++)
      testData_7[i] = rand() % 40 - 20;
  for (int i = 0;i < 100000000;i++)
      testData_8[i] = rand() % 40 - 20;
}

int MaxSubsequenceSum_1(const int A[], int N)
{
  int ThisSum, MaxSum, i, j, k;
  /* 1*/ MaxSum = 0; /* initialize the maximum sum */
  /* 2*/ for (i = 0; i < N; i++) /* start from A[ i ] */
      /* 3*/ for (j = i; j < N; j++) { /* end at A[ j ] */
          /* 4*/ ThisSum = 0;
          /* 5*/ for (k = i; k <= j; k++)
              /* 6*/ ThisSum += A[k]; /* sum from A[ i ] to A[ j ] */
          /* 7*/ if (ThisSum > MaxSum)
```

```
            /* 8*/ MaxSum = ThisSum; /* update max sum */
        } /* end for-j and for-i */
    /* 9*/ return MaxSum;
}

int MaxSubsequenceSum_2(const int A[], int N)
{
    int ThisSum, MaxSum, i, j;
    /* 1*/ MaxSum = 0; /* initialize the maximum sum */
    /* 2*/ for (i = 0; i < N; i++) { /* start from A[ i ] */
        /* 3*/ ThisSum = 0;
        /* 4*/ for (j = i; j < N; j++) { /* end at A[ j ] */
            /* 5*/ ThisSum += A[j]; /* sum from A[ i ] to A[ j ] */
            /* 6*/ if (ThisSum > MaxSum)
                /* 7*/ MaxSum = ThisSum; /* update max sum */
        } /* end for-j */
    } /* end for-i */
    /* 8*/ return MaxSum;
}

int BinarySearch(const int A[], int left, int right)
{
    if (left == right)
        return A[left];
    else
    {
        int center = (left + right) / 2;
        int LeftSum = BinarySearch(A, left, center);
        int RightSum = BinarySearch(A, center+1, right);
        int LeftOfTotalSum = -99999;
        int RightOfTotalSum = -99999;
        int TotalSum = 0;
        int TempLeftSum = 0;
        int TempRightSum = 0;
        for (int i = center;i >= left;i--)
        {
            TempLeftSum += A[i];
            if (TempLeftSum > LeftOfTotalSum)
                LeftOfTotalSum = TempLeftSum;
        }
        for (int i = center + 1;i <= right;i++)
        {
            TempRightSum += A[i];
            if (TempRightSum > RightOfTotalSum)
                RightOfTotalSum = TempRightSum;
        }
        TotalSum = LeftOfTotalSum + RightOfTotalSum;

        if ((RightSum > LeftSum) && (RightSum > TotalSum))
```

```
            return RightSum;
        else if (LeftSum > TotalSum)
            return LeftSum;
        else
            return TotalSum;
    }
}

int MaxSubsequenceSum_3(const int A[], int N)
{
    int MaxSum = 0;
    MaxSum = BinarySearch(A, 0, N - 1);
    return MaxSum;
}

int MaxSubsequenceSum_4(const int A[], int N)
{
    int ThisSum, MaxSum, j;
    /* 1*/ ThisSum = MaxSum = 0;
    /* 2*/ for (j = 0; j < N; j++) {
        /* 3*/ ThisSum += A[j];
        /* 4*/ if (ThisSum > MaxSum)
            /* 5*/ MaxSum = ThisSum;
        /* 6*/ else if (ThisSum < 0)
            /* 7*/ ThisSum = 0;
    } /* end for-j */
    /* 8*/ return MaxSum;
}

int main()
{
    int result = 0;
    double time = 0.0;

    InitialTestData();

    cout <<
"*************************************************************************
****" << endl;
    cout << "当数据量为 10 时：" << endl;
    start = clock();
    result = MaxSubsequenceSum_1(testData_1, 10);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法一用时：" << time * 1000 << "ms，测算结果为：" <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_2(testData_1, 10);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法二用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_3(testData_1, 10);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法三用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_4(testData_1, 10);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法四用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;
    cout <<
"************************************************************************
****" << endl;
    cout << endl;

    cout <<
"************************************************************************
****" << endl;
    cout << "当数据量为 100 时: " << endl;
    start = clock();
    result = MaxSubsequenceSum_1(testData_2, 100);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法一用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_2(testData_2, 100);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法二用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_3(testData_2, 100);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法三用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_4(testData_2, 100);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法四用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;
    cout <<
"***********************************************************************
****" << endl;
    cout << endl;

    cout <<
"***********************************************************************
****" << endl;
    cout << "当数据量为 1000 时: " << endl;
    start = clock();
    result = MaxSubsequenceSum_1(testData_3, 1000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法一用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_2(testData_3, 1000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法二用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_3(testData_3, 1000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法三用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_4(testData_3, 1000);
    ending = clock();
```

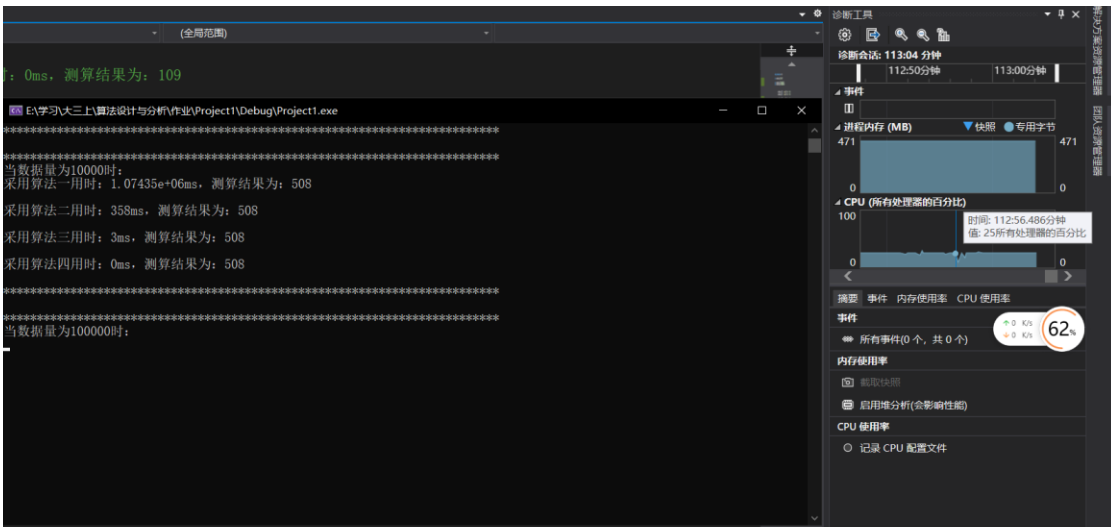
```
   time = (double)(ending - start) / CLOCKS_PER_SEC;
   cout << "采用算法四用时： " << time * 1000 << "ms，测算结果为： " <<
result << endl;
   cout << endl;
   cout <<
"**********************************************************************
****" << endl;
   cout << endl;

   cout <<
"**********************************************************************
****" << endl;
   cout << "当数据量为 10000 时： " << endl;
   start = clock();
   result = MaxSubsequenceSum_1(testData_4, 10000);
   ending = clock();
   time = (double)(ending - start) / CLOCKS_PER_SEC;
   cout << "采用算法一用时： " << time * 1000 << "ms，测算结果为： " <<
result << endl;
   cout << endl;

   start = clock();
   result = MaxSubsequenceSum_2(testData_4, 10000);
   ending = clock();
   time = (double)(ending - start) / CLOCKS_PER_SEC;
   cout << "采用算法二用时： " << time * 1000 << "ms，测算结果为： " <<
result << endl;
   cout << endl;

   start = clock();
   result = MaxSubsequenceSum_3(testData_4, 10000);
   ending = clock();
   time = (double)(ending - start) / CLOCKS_PER_SEC;
   cout << "采用算法三用时： " << time * 1000 << "ms，测算结果为： " <<
result << endl;
   cout << endl;

   start = clock();
   result = MaxSubsequenceSum_4(testData_4, 10000);
   ending = clock();
   time = (double)(ending - start) / CLOCKS_PER_SEC;
   cout << "采用算法四用时： " << time * 1000 << "ms，测算结果为： " <<
result << endl;
   cout << endl;
   cout <<
"**********************************************************************
****" << endl;
   cout << endl;
```

```
    cout <<
"***********************************************************************
****" << endl;
    cout << "当数据量为 100000 时：" << endl;
    start = clock();
    result = MaxSubsequenceSum_1(testData_5, 100000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法一用时：" << time * 1000 << "ms，测算结果为：" <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_2(testData_5, 100000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法二用时：" << time * 1000 << "ms，测算结果为：" <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_3(testData_5, 100000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法三用时：" << time * 1000 << "ms，测算结果为：" <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_4(testData_5, 100000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法四用时：" << time * 1000 << "ms，测算结果为：" <<
result << endl;
    cout << endl;
    cout <<
"***********************************************************************
****" << endl;
    cout << endl;

    cout <<
"***********************************************************************
****" << endl;
    cout << "当数据量为 1000000 时：" << endl;
    start = clock();
    result = MaxSubsequenceSum_1(testData_6, 1000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法一用时: " << time * 1000 << "ms，测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_2(testData_6, 1000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法二用时: " << time * 1000 << "ms，测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_3(testData_6, 1000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法三用时: " << time * 1000 << "ms，测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_4(testData_6, 1000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法四用时: " << time * 1000 << "ms，测算结果为: " <<
result << endl;
    cout << endl;
    cout <<
"*********************************************************************
****" << endl;
    cout << endl;

    cout <<
"*********************************************************************
****" << endl;
    cout << "当数据量为 10000000 时: " << endl;
    start = clock();
    result = MaxSubsequenceSum_1(testData_7, 10000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法一用时: " << time * 1000 << "ms，测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_2(testData_7, 10000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法二用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_3(testData_7, 10000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法三用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_4(testData_7, 10000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法四用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;
    cout <<
"***********************************************************************
****" << endl;
    cout << endl;

    cout <<
"***********************************************************************
****" << endl;
    cout << "当数据量为 100000000 时: " << endl;
    start = clock();
    result = MaxSubsequenceSum_1(testData_8, 100000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法一用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_2(testData_8, 100000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法二用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_3(testData_8, 100000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法三用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;

    start = clock();
    result = MaxSubsequenceSum_4(testData_8, 100000000);
    ending = clock();
    time = (double)(ending - start) / CLOCKS_PER_SEC;
    cout << "采用算法四用时: " << time * 1000 << "ms, 测算结果为: " <<
result << endl;
    cout << endl;
    cout <<
"************************************************************************
****" << endl;
    cout << endl;

    system("pause");
    return 0;
 }
```

## 二、 实验结果和结论

```
 ***********************************************************************
****
 当数据量为 10 时:
 采用算法一用时: 0ms, 测算结果为: 36

 采用算法二用时: 0ms, 测算结果为: 36

 采用算法三用时: 0ms, 测算结果为: 36

 采用算法四用时: 0ms, 测算结果为: 36

 ***********************************************************************
****

 ***********************************************************************
****
 当数据量为 100 时:
 采用算法一用时: 1ms, 测算结果为: 109

 采用算法二用时: 0ms, 测算结果为: 109

 采用算法三用时: 0ms, 测算结果为: 109

 采用算法四用时: 0ms, 测算结果为: 109

 ***********************************************************************
****

 ***********************************************************************
****
 当数据量为 1000 时:
 采用算法一用时: 938ms, 测算结果为: 320

 采用算法二用时: 2ms, 测算结果为: 320

 采用算法三用时: 0ms, 测算结果为: 320

 采用算法四用时: 0ms, 测算结果为: 320

 ***********************************************************************
****

 ***********************************************************************
****
 当数据量为 10000 时:
 采用算法一用时: 1.07435e+06ms, 测算结果为: 508
```

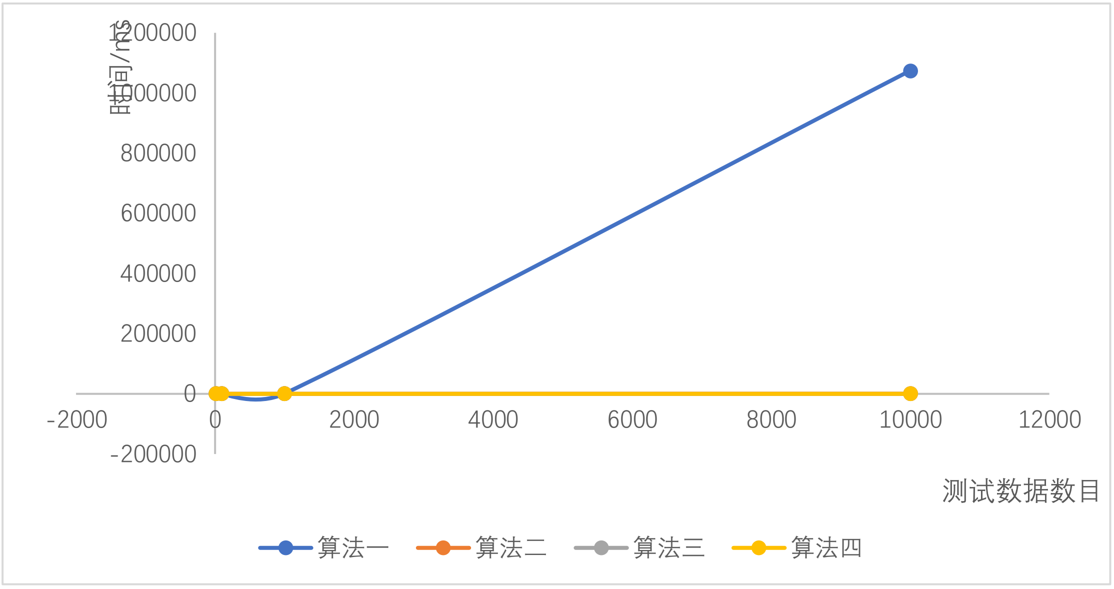
```
采用算法二用时: 358ms, 测算结果为: 508

采用算法三用时: 3ms, 测算结果为: 508

采用算法四用时: 0ms, 测算结果为: 508

 ************************************************************************
****
```



```
: 0ms, 测算结果为: 109
```

```
E:\学习\大三上\算法设计与分析\作业\Project1\Debug\Project1.exe
****************************************************************************

****************************************************************************
当数据量为10000时:
采用算法一用时: 1.07435e+06ms, 测算结果为: 508

采用算法二用时: 358ms, 测算结果为: 508

采用算法三用时: 3ms, 测算结果为: 508

采用算法四用时: 0ms, 测算结果为: 508

****************************************************************************

****************************************************************************
当数据量为100000时:
```

```
诊断工具
诊断会话: 113:04 分钟
112:50分钟    113:00分钟
事件
进程内存 (MB)    快照  专用字节
471    471
0    0
CPU (所有处理器的百分比)
100
时间: 112:56.486分钟
值: 25所有处理器的百分比
0    0
摘要  事件  内存使用率  CPU 使用率
事件
所有事件(0 个, 共 0 个)
内存使用率
截取快照
启用堆分析(会影响性能)
CPU 使用率
记录 CPU 配置文件
```



## 三、 算法分析

**算法一**

本算法使用了三层循环。第一层来进行子段的起点遍历，第二层进行子段的终点遍历，第三层从前往后进行起点与终点之间的每个数的逐次相加来求得最大子段和。三层循环非常耗费时间，根据分析我们其实可以得知，第三层循环和第二层循环其实可以变成一次循环，因此进阶到算法二。

**算法二**

算法二是在算法一的基础上发展来的，实现原理等同于算法一，这里不再赘述。

**算法三**

算法三是一种分治思想，具体表现为二分查找。先将母段二分再二分，然后在每一个被二分的子段中继续二分，然后将无法进行二分的子段求出最大子段和，再回归到上一个直接母段。对于一个母段而言，他的最大子段和只有可能三种情况：左侧子段的最大子段和、右侧子段的最大子段和及其整体的最大子段和。利用这样的二分思想，可以提高算法效率、节省求解时间。

**算法四**

算法四继续推陈出新。我们知道，最大子段和的所谓子段，必须是在母段上顺序读取若干个数得到的，因此事实上，我们只要从前往后遍历一次母段，将每一个结点的数加上，记录这个过程中曾形成的最大子段和，然后判断加上这个数前后，对于我们记录的最大子段和的影响。如果加上他后，最大子段和可以更大，我们就进行更新；如果变小了，我们先不将当前子段和清零，因为我们并不知道后续会不会出现一个很大的数、在当前子段和的基础上刷新了最大子段和，因此只要当前字段和不为负，我们就不将当前子段和清零。当当前子段和为负数时，我们应该明白需要抛弃当前的所有结点，因为显然我们直接从后续结点开始计算得到的值必定大于等于我们从当前结点发展到后续结点所得到的值。此外，我们也不需要考虑当前子段的中间部分，因为我们是每加一个结点就进行一次判断，所以不可能出现从当前子段的中间某个部分开始计算会得到更大子段和的情况发生。

这样的算法使得一次遍历 O(n)就可以得到我们想要的结果。

## 四、 心得体会

在本次实验过程中，我尝试了四种不同的算法对最大子段和进行了求解，使用了 10-10000 的数据量，本来想尝试 100000-100000000 数据量，但跑了两个小时都没反映出来算法一，自己估算了一下时间，实在是太耗时间了，遂放弃。

本次实验让我认识到了算法的时间复杂度在实际应用场景中对于效率的影响是多么的巨大，直观表现上就是数据量增大后，大量的时间耗费在算法一上，算法一跑完后，立马就可以输出算法二、三、四的运算结果和用时。相同的数据量、相同的正确的运算结果，算法一表现最差、算法四表现最好。

算法的优良性在较小数据量时尚未拥有肉眼可见的表现效果，但数据量一大就表现得非常突出。而在实际的应用场景中，我们需要处理的数据量必定是巨大、丰富的，这对我们的算法优良性就提出了巨大的要求和挑战，同时，一双手、一副头脑就可以创造出表现迥然不同的效果，在我看来是非常神奇的，在认识到算法重要性的同时，我也对算法充满了兴趣和信心。我相信我一定可以学好算法设计与分析这门课。