

# 实验报告

## 一.实验内容

1.对规模为 5, 9, 13, 17, 21 的输入, 建立最优搜索二叉树, 输出期望搜索代价和最优搜索二叉树的前序遍历结果。

2.实现最长公共子序列的算法, 给出对应序列的最长公共子序列长度和其中一个最长公共子序列。

## 二.实验环境

采用语言为 C 语言, 编译环境为 Mac 上的 Xcode。

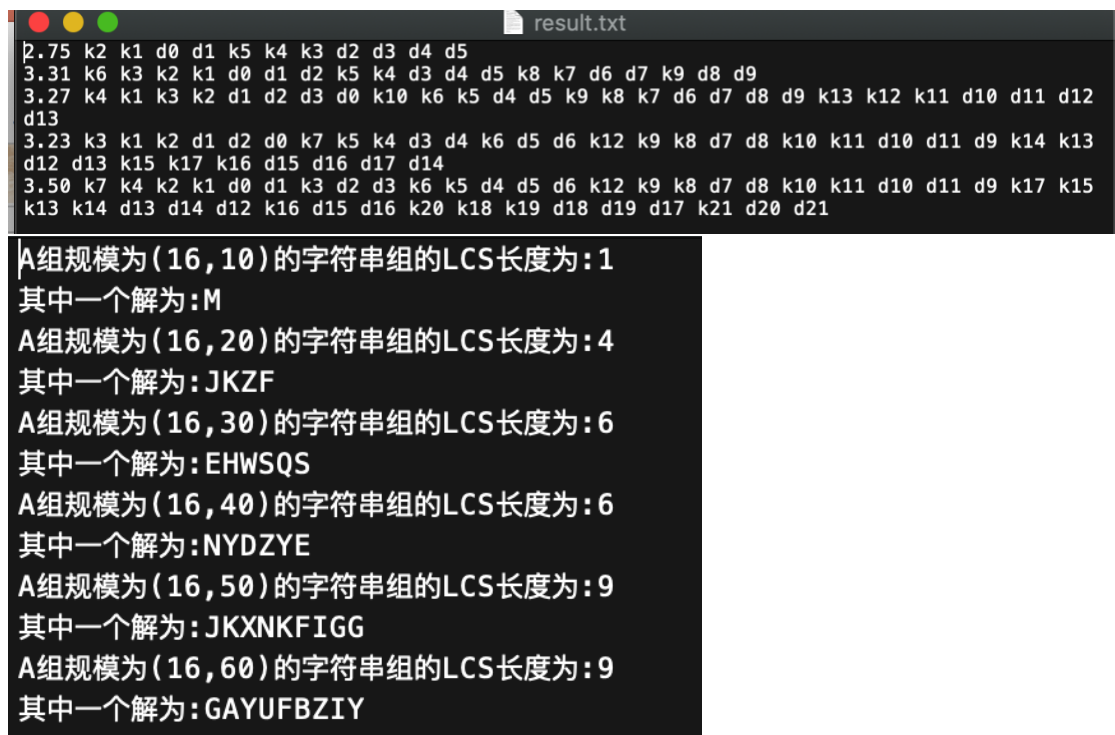
## 三.实验步骤

1.采用给定的 input 文件, 使用 fopen 从文件中输入数据, 放入数组 p 和 q 中, 使用课本上的算法, 构建最优搜索二叉树。将输出结果保存在 output 文件夹中, reuslt.txt 存放结果, time.txt 存放程序运行时间。

2.先通过随机数生成程序, 并对应 26 个字母, 产生随机字符串序列, 分别保存在 input 文件夹中的 inputA.txt 和 inputB.txt 文件中。通过 fopen 从文件中独如数据, 保存在 char 型数组 x 和 y 中, 使用课本上求最大子序列的算法进行处理, 控制台输入每次处理的字符串长度。结果存放在 output 文件中, resulr.txt 存放结果, time.txt 存放程序运行时间。

## 四.实验结果

两个实验的 output.txt 中内容如下:

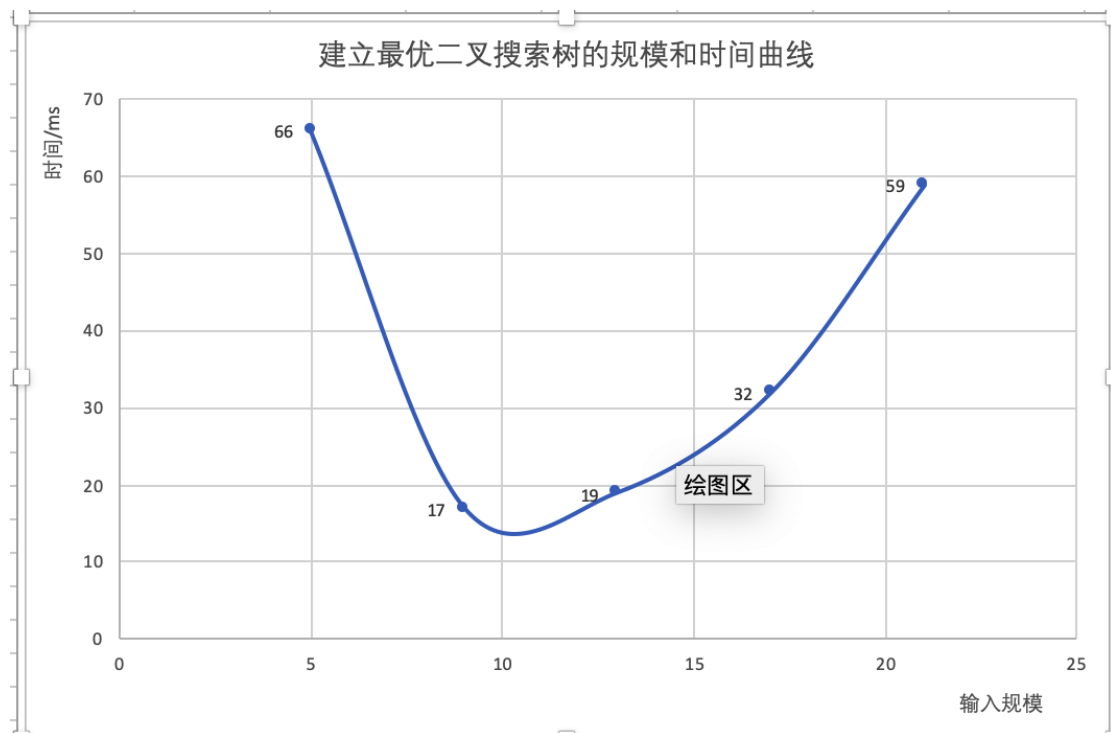


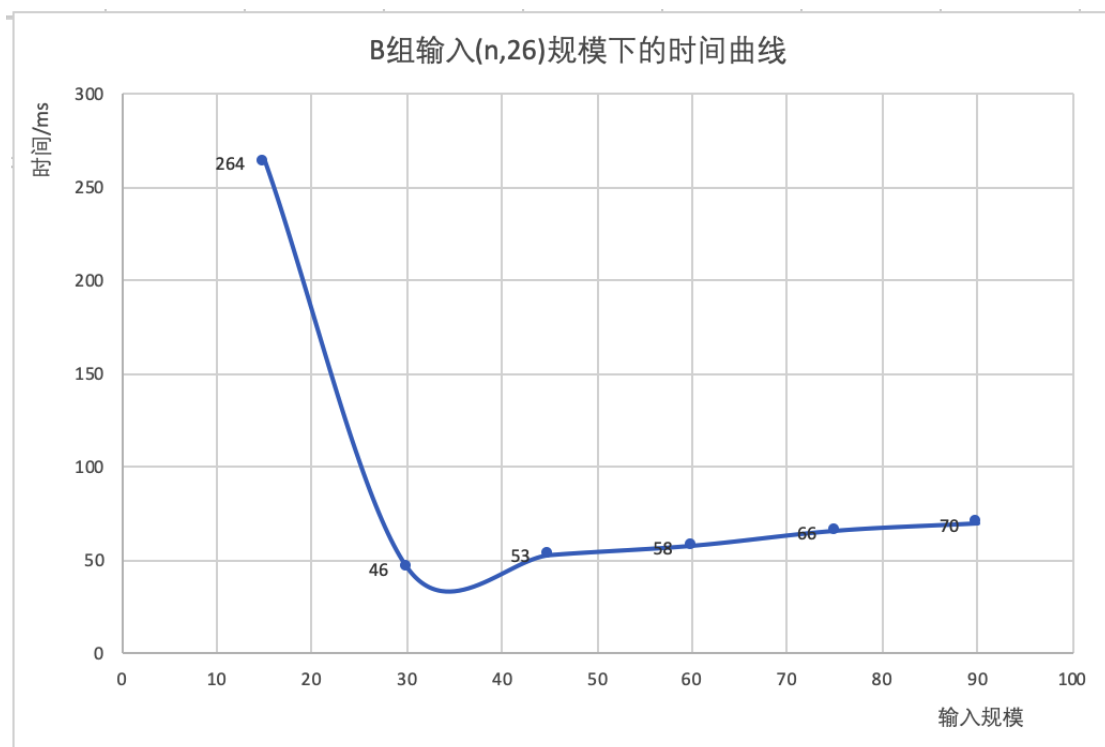
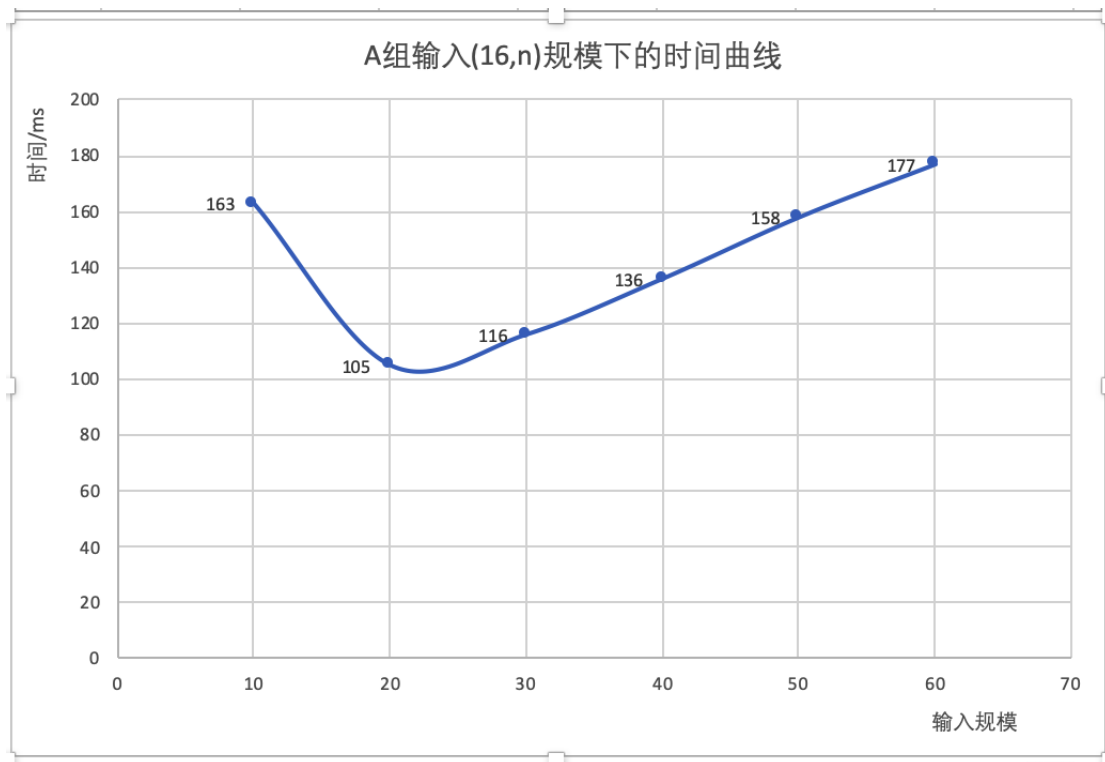
```
result.txt
2.75 k2 k1 d0 d1 k5 k4 k3 d2 d3 d4 d5
3.31 k6 k3 k2 k1 d0 d1 d2 k5 k4 d3 d4 d5 k8 k7 d6 d7 k9 d8 d9
3.27 k4 k1 k3 k2 d1 d2 d3 d0 k10 k6 k5 d4 d5 k9 k8 k7 d6 d7 d8 d9 k13 k12 k11 d10 d11 d12
d13
3.23 k3 k1 k2 d1 d2 d0 k7 k5 k4 d3 d4 k6 d5 d6 k12 k9 k8 d7 d8 k10 k11 d10 d11 d9 k14 k13
d12 d13 k15 k17 k16 d15 d16 d17 d14
3.50 k7 k4 k2 k1 d0 d1 k3 d2 d3 k6 k5 d4 d5 d6 k12 k9 k8 d7 d8 k10 k11 d10 d11 d9 k17 k15
k13 k14 d13 d14 d12 k16 d15 d16 k20 k18 k19 d18 d19 d17 k21 d20 d21

A组规模为(16,10)的字符串组的LCS长度为:1
其中一个解为:M
A组规模为(16,20)的字符串组的LCS长度为:4
其中一个解为:JKZF
A组规模为(16,30)的字符串组的LCS长度为:6
其中一个解为:EHWSQS
A组规模为(16,40)的字符串组的LCS长度为:6
其中一个解为:NYDZYE
A组规模为(16,50)的字符串组的LCS长度为:9
其中一个解为:JKXNKFIGG
A组规模为(16,60)的字符串组的LCS长度为:9
其中一个解为:GAYUFBZIY
```

B组规模为(15,26)的字符串组的LCS长度为:7  
其中一个解为:HFSBHBV  
B组规模为(30,26)的字符串组的LCS长度为:8  
其中一个解为:CUKMUAOK  
B组规模为(45,26)的字符串组的LCS长度为:6  
其中一个解为:SBPGDD  
B组规模为(60,26)的字符串组的LCS长度为:9  
其中一个解为:EIWPPCJBD  
B组规模为(75,26)的字符串组的LCS长度为:13  
其中一个解为:MVLBWZRLQSVJT  
B组规模为(90,26)的字符串组的LCS长度为:14  
其中一个解为:PPEYPMPQJVDZQZ

在两个实验的运行时间上,在第一次程序执行的过程中,时间会异常高,之后的程序运行时间按规模增加,应该恢复了正常。





从上面的时间曲线来看，在正常情况下应为线性增长。对于第一个数据的异常情况，猜测可能为在分配内存时，第一次分配的 cache 丢失率比较高，之后的命中率较高，所以第一个消耗了较多时间，是否是正确原因不能确定。

## 五.代码说明

1.在输出最优搜索二叉树时，采用了如下的代码：

```

void Print_Tree(int root[n+1][n+1],int i,int j){
    int r;
    r=root[i][j];
    fprintf(fp2,"k%d ",r);
    if(i!=j&&r>i){
        Print_Tree(root,i,r-1);
    }
    if(i!=j&&r<j){
        Print_Tree(root,r+1,j);
    }
    if(i==r){
        fprintf(fp2,"d%d ",r-1);
    }
    if(r==j){
        fprintf(fp2,"d%d ",r);
    }
}
}

```

该代码采用了递归的前序遍历。先输出左子树 k 的节点，然后进入递归，最后输出叶子节点 d，这样保证了从 k 到 d 的输出顺序。

2.在输入处理方面，一次只能处理一个文件的输入。

```

fp1=fopen
("/Users/wuyuxuan/Documents/xcode/PB17030797-吴钰轩-project2/ex2/input/inputB
.txt","r");
fp2=fopen("result.txt","a");
fp3=fopen("time.txt","a");
int count;
int i,j;
int begin,end;
for(count=0;count<6;count++){
    begin=clock();
    scanf("%d",&m);
    scanf("%d",&n);
    int c[m+1][n+1],b[m+1][n+1];
    char x[m+2],y[n+2];
}

```

若需要处理文件 inputA.txt 则需要讲 fp1 打开的文件名作修改。同时，每次处理需要的字符串长度也需要从控制台输入，文件中有几组字符串，就需要输入几次。其余代码和课本给定代码相同。