华东师范大学数据科学与工程学院上机实践报告

课程名称：算法设计与分析	**年级**：21 级	**上机实践成绩**：
指导教师：金澈清	**姓名**：杨茜雅	
上机实践名称：元素查找	**学号**：10215501435	**上机实践日期**：
上机实践编号：No.10	**组号**：1-435	

一、目的

1．熟悉算法设计的基本思想

2．掌握计数排序（count sort）的方法

二、内容与设计思想

有一个公司想开发一个关于花卉的百科全书，用户只要输入花卉的名称，就能够输出花卉的详细信息。花卉包括：牡丹、芍药、茶花、菊花、梅花、兰花、月季、杜鹃花、郁金香、茉莉花、海棠、荷花、栀子花、莲花、百合、康乃馨、玫瑰、格桑花（编程的时候可以用花名，也可以直接用 1-18 的编号来代替）。公司也在试运行阶段发现这些花的访问频率不一，有些花经常性被访问，有些被访问的次数就少很多了。这 18 种花中，第 1 种的访问频率是 6，第 2-3 种的访问频率是 5，第 4-6 种的访问频率是 4，第 7-10 种的访问频率是 3，第 11-15 种的访问频率是 2，第 16-18 种的访问频率是 1。

这个公司想提升花卉检索效率，所以对比了三种方法。

1. 通过动态规划构建优化的二叉搜索树（optimal BST），进行搜索。
2. 将
3. 构建哈希表（链表法）来存储并检索数据。

请实现这三种方法，并且通过实验来比较这三种方法的优劣。

你是否还能够想出其他高效的方法?

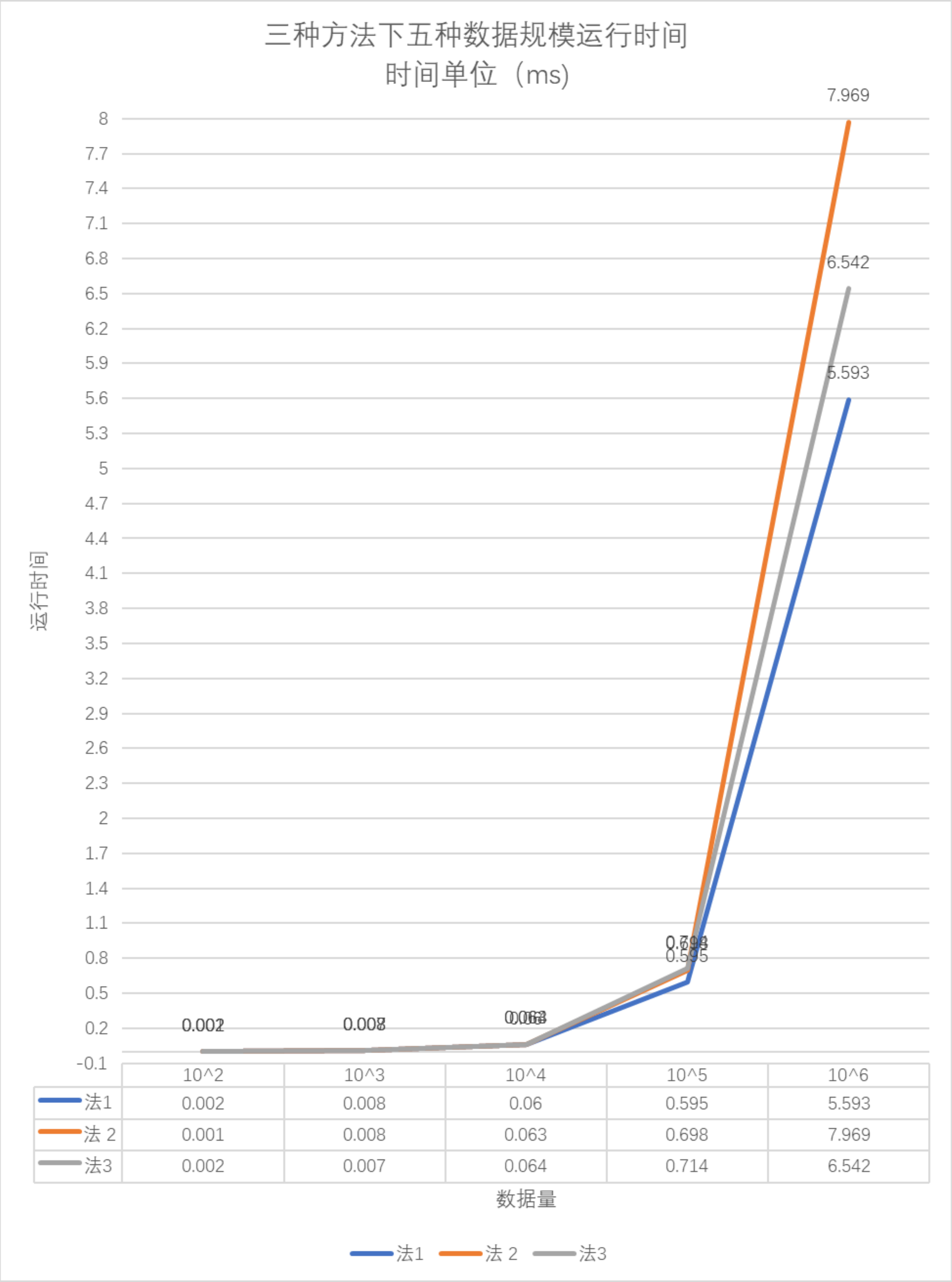
三、使用环境

推荐使用 C/C++集成编译环境。

四、实验过程

*1.　　分别画出在不同的**查询次数**下，各个数据结构的**查询性能**折线图*

2.



	10^2	10^3	10^4	10^5	10^6
法1	0.002	0.008	0.06	0.595	5.593
法 2	0.001	0.008	0.063	0.698	7.969
法3	0.002	0.007	0.064	0.714	6.542

五、总结

对上机实践结果进行分析，问题回答，上机的心得体会及改进意见。

法 1

```
#include<bits/stdc++.h>
#include <fstream>
#include<iostream>
#include<string>
#include<math.h>
#include <stdlib.h>
#include <time.h>
#include<cstring>
int search_array[10000000];
typedef struct name{
    int num;
    char intro[10000000];
}NAME;
using namespace std;
struct BSTNode
{
    int value;
    BSTNode *left;
    BSTNode *right;
    BSTNode(int _value)
    {
        value=_value;
        left=NULL;
        right=NULL;
    }
    BSTNode()
    {
    }
};
void calRootsofoptimalBinaryTree(float *p,float *q,int * *
root,int n)
{
    typedef float *pFloat;
    pFloat *weight=new pFloat[n];
    pFloat *cost=new pFloat[n];
    for (int i=0; i<n; i++)
    {
        weight[i]=new float[n];
        cost[i]=new float[n];
    }

    for (int i=0; i<n; i++)
    {
        int j=0;
        for (j=0; j<n; j++)
        {
            weight[i][j]=0;
            cost[i][j]=0;
        }
    }
    for (int i=0; i<n; i++)
    {
        weight[i][i]=q[i];
    }
    int l=1;
    for (l=1; l<n; l++)
    {
        for (int i=0; i<n-l; i++)
        {
            int j=l+i;
            weight[i][j]=weight[i][j-1]+p[j]+q[j];
            int k=i+1;
            int m=k;
            float minVal=cost[i][k-1]+cost[k][j];
            for (; k<=j; k++)
            {
                if(cost[i][k-1]+cost[k][j]<minVal)
                {
                    minVal=cost[i][k-1]+cost[k][j];
                    m=k;
                }
            }
            cost[i][j]=weight[i][j]+cost[i][m-1]+cost[m][j];
            root[i][j]=m;
        }
    }
}

BSTNode *buildTree(int i,int j,int * * root,int * a,int n)
{
    if (i>=n||j>=n)
    {
        return NULL;
    }
    int m=root[i][j];
    BSTNode *rootnode=new BSTNode();
    rootnode->value=a[m];
    rootnode->right=NULL;
    rootnode->left=NULL;
    if (i<m-1)
    {
        rootnode->left=buildTree(i,m-1,root,a,n);
    }
    if (j>m)
    {
        rootnode->right=buildTree(m,j,root,a,n);
    }
    return rootnode;
}
BSTNode *OptimalBinaryTree()
```

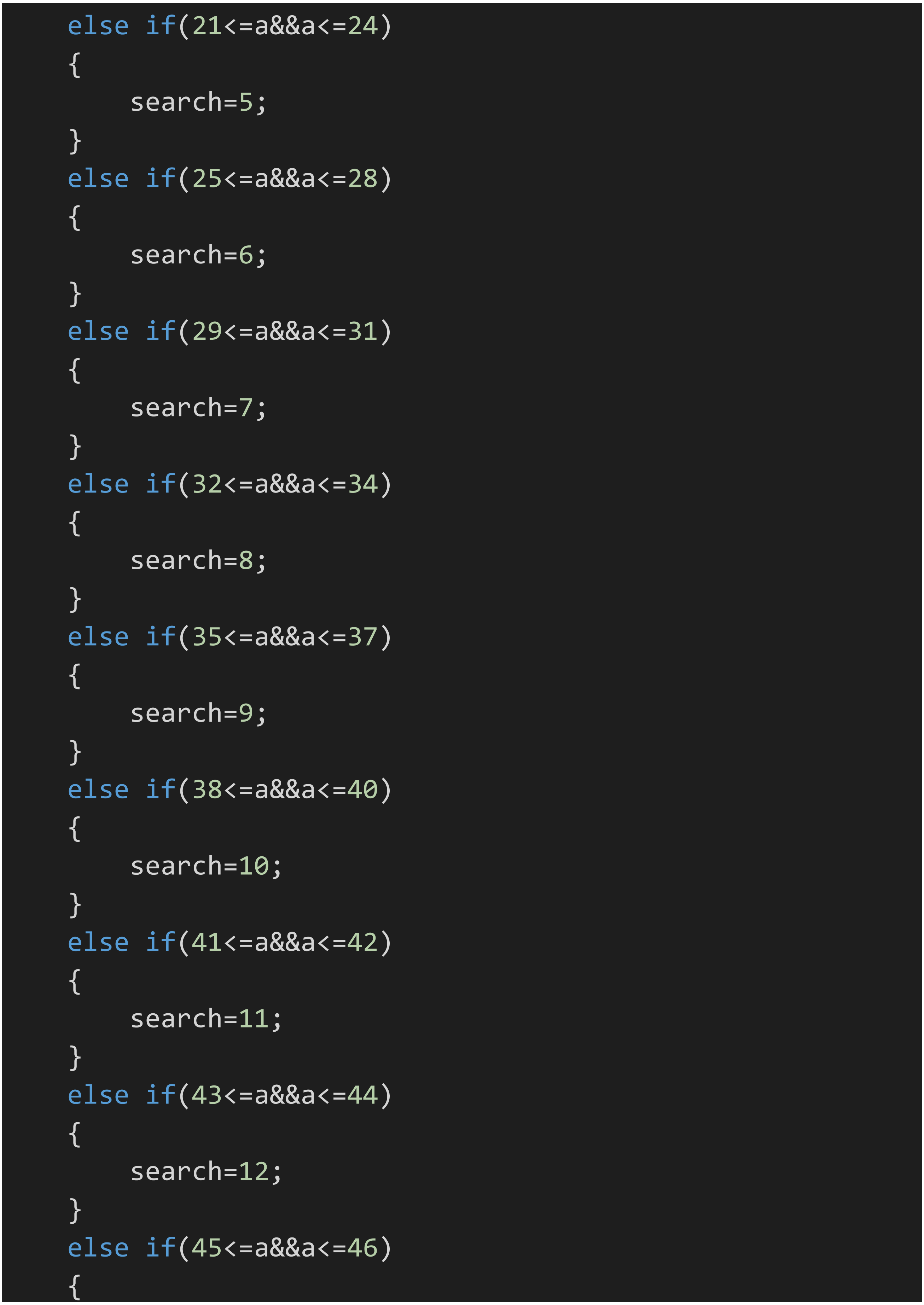
```
{
    int * * root=new int *[19];
    int i=0;
    for (i=0; i<19; i++)
    {
        root[i]= new int[19];
        memset(root[i],0,sizeof(int) * 19);
    }
    int
a[]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18};
    float
p[]={0,0.113,0.094,0.094,0.075,0.075,0.075,0.057,0.057,0.0
57,0.057,0.038,0.038,0.038,0.038,0.038,0.019,0.019,0.019};
    float q[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    calRootsofoptimalBinaryTree(p,q,root,19);
    BSTNode *r=buildTree(0,18,root,a,19);
}
int Search_Tree(BSTNode *node,int item)
{
     while(node!=NULL)
    {
        if(node->value==item)
        {
            return item;
        }
        if(item<node->value)
        {
            node=node->left;
        }
        else
        {
            node=node->right;
        }
    }
```

```
     return 0;

}
int Search_flower(int a)
{
     int search=0;//对于生成的53个随机数分段用来模拟不同花卉的不同查询频度，先初始化
    if(1<=a&&a<=6)
    {
        search=1;
    }
    else if(7<=a&&a<=11)
    {
        search=2;
    }
    else if(12<=a&&a<=16)
    {
        search=3;
    }
    else if(17<=a&&a<=20)
    {
        search=4;
    }
    else if(21<=a&&a<=24)
    {
        search=5;
    }
    else if(25<=a&&a<=28)
    {
        search=6;
    }
    else if(29<=a&&a<=31)
    {
        search=7;
    }
    else if(32<=a&&a<=34)
    {
        search=8;
    }
    else if(35<=a&&a<=37)
    {
        search=9;
    }
    else if(38<=a&&a<=40)
    {
        search=10;
    }
    else if(41<=a&&a<=42)
    {
        search=11;
    }
    else if(43<=a&&a<=44)
    {
        search=12;
    }
    else if(45<=a&&a<=46)
    {
        search=13;
    }
    else if(47<=a&&a<=48)
    {
        search=14;
    }
    else if(49<=a&&a<=50)
    {
        search=15;
    }
    else if(a==51)
    {
        search=16;
    }
    else if(a==52)
    {
        search=17;
    }
    else if(a==53)
    {
        search=18;
    }
    return search;
}
int main()
{
    NAME* list = new NAME[19];
    ifstream siFile("flower.txt");
    for(int i=1;i<=18;i++)//按频度从高到低放入数组里
    {
        siFile>>list[i].num;
       siFile.get(list[i].intro,1000000000,'\n');
    }
   siFile.close();
   ofstream soFile("output.txt");
    BSTNode *BST=OptimalBinaryTree();
    int N;
    cin>>N;
    srand((unsigned)time(NULL));
    for(int i=1;i<=N;i++)
    {
    int a=rand()%53+1;
    search_array[i]=Search_flower(a);//构建出一个数组存放每次搜索花卉的编号
    }
    double start=clock();
    for(int i=1;i<=N;i++)
     {
    if(search_array[i]==Search_Tree(BST,search_array[i]))
    {
        soFile << list[search_array[i]].num;
        soFile << "\t";
        soFile << list[search_array[i]].intro;
        soFile << endl;
    }
    }
    soFile.close();
    double end=clock();
    double diff=(double)(end-start)/CLOCKS_PER_SEC;
    cout<<endl<<diff;
    return 0;
}
```

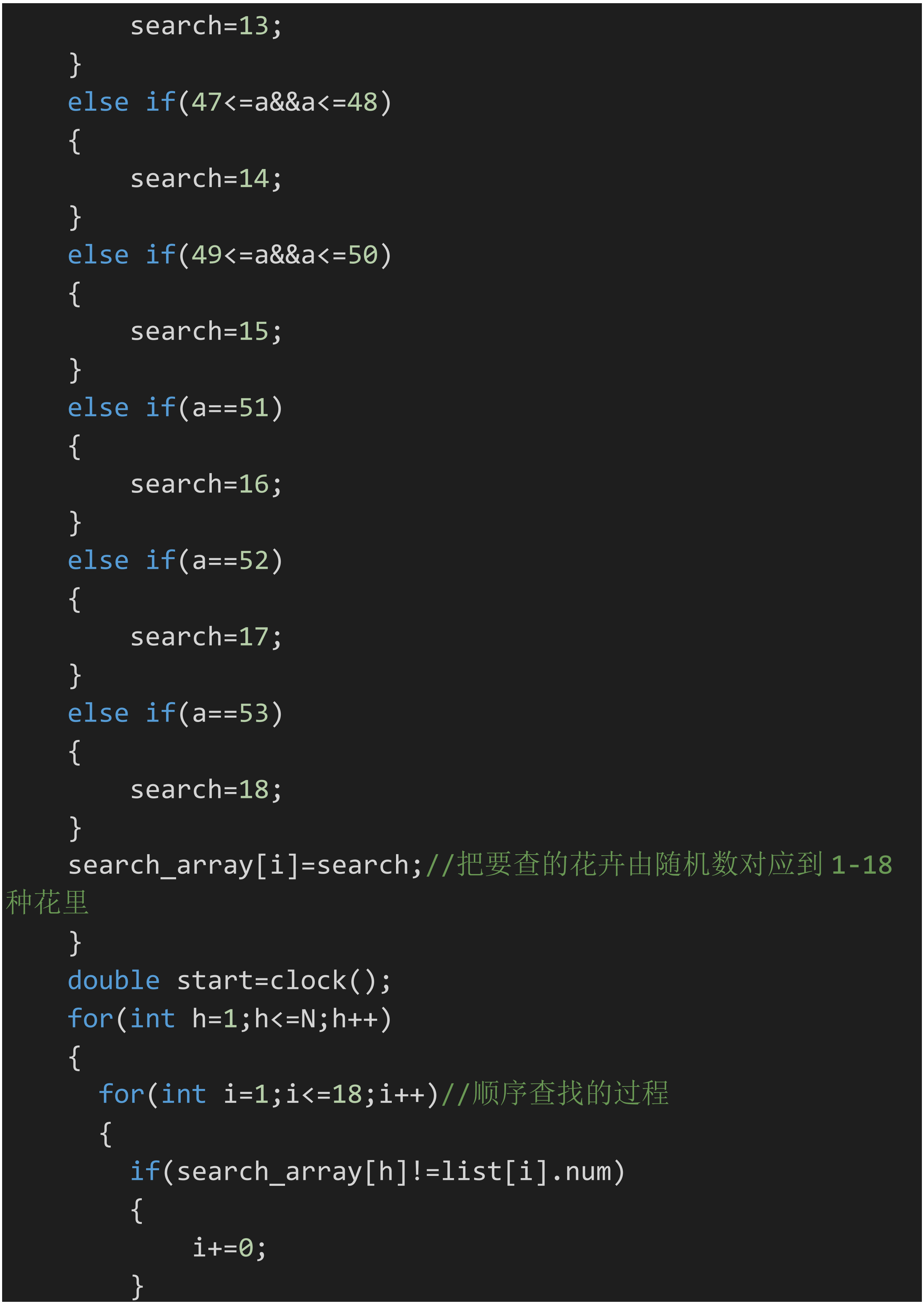
法二：

```
#include<bits/stdc++.h>
#include <fstream>
#include<iostream>
#include<string>
#include<math.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
typedef struct name{
    int num;
    char intro[10000000];
}NAME;
int search_array[10000000];//n 次查询每次对应查询的花卉编号数组
int main()
```

```
{
    int N;//查询次数
    cin>>N;
    NAME* list = new NAME[19];
    ifstream siFile("flower.txt");
    for(int i=1;i<=18;i++)//按频度从高到低放入数组里
    {
        siFile>>list[i].num;
        siFile.get(list[i].intro,1000000000,'\n');
    }
    siFile.close();
    ofstream soFile("output.txt");
    srand((unsigned)time(NULL));
    for(int i=1;i<=N;i++)
    {
    int a=1+rand()%53;
    int search;//对于生成的 53 个随机数分段用来模拟不同花卉的不
同查询频度
    if(1<=a&&a<=6)
    {
        search=1;
    }
    else if(7<=a&&a<=11)
    {
        search=2;
    }
    else if(12<=a&&a<=16)
    {
        search=3;
    }
    else if(17<=a&&a<=20)
    {
        search=4;
    }
```



```
    else if(21<=a&&a<=24)
    {
        search=5;
    }
    else if(25<=a&&a<=28)
    {
        search=6;
    }
    else if(29<=a&&a<=31)
    {
        search=7;
    }
    else if(32<=a&&a<=34)
    {
        search=8;
    }
    else if(35<=a&&a<=37)
    {
        search=9;
    }
    else if(38<=a&&a<=40)
    {
        search=10;
    }
    else if(41<=a&&a<=42)
    {
        search=11;
    }
    else if(43<=a&&a<=44)
    {
        search=12;
    }
    else if(45<=a&&a<=46)
    {
```

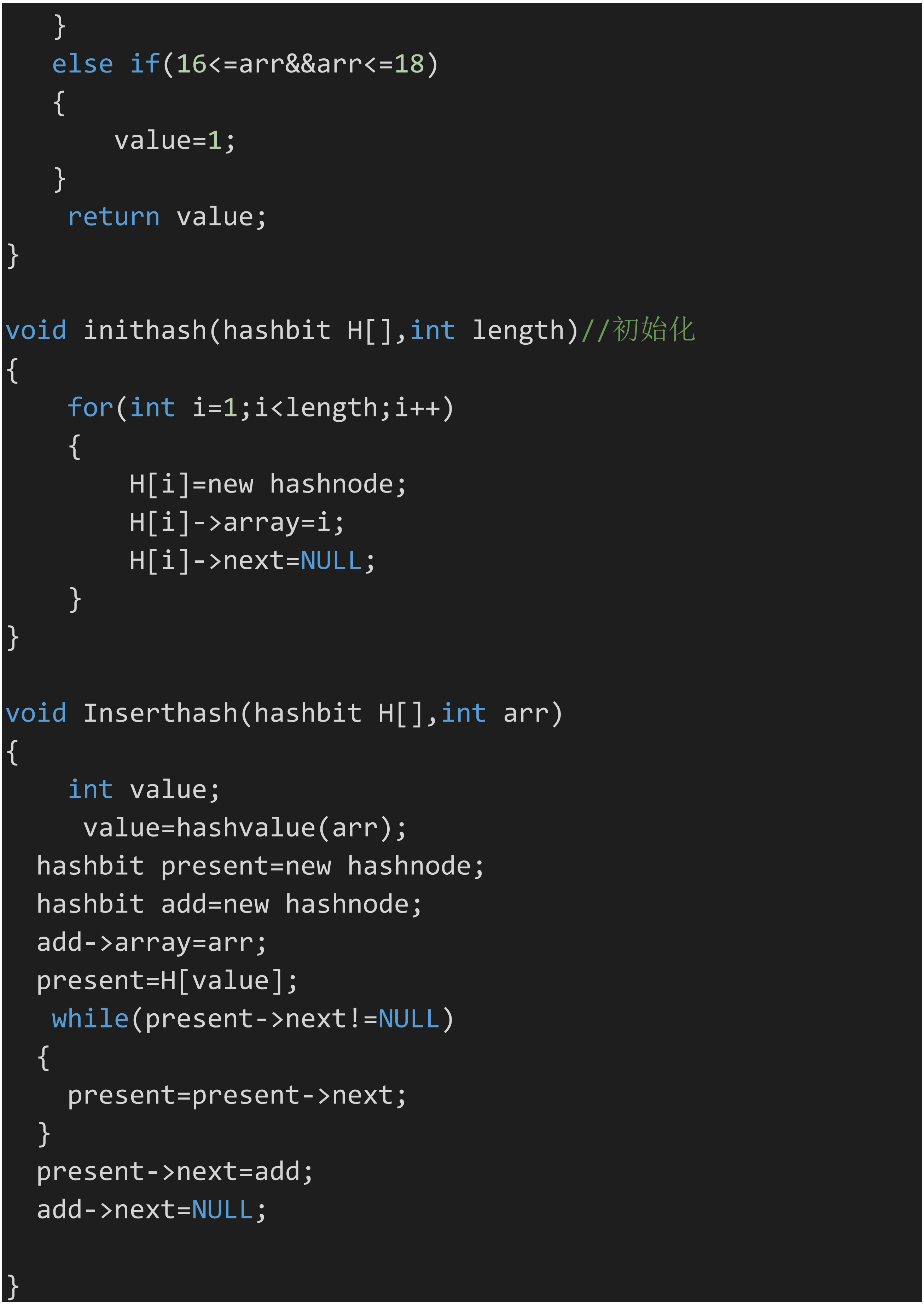


```
        search=13;
    }
    else if(47<=a&&a<=48)
    {
        search=14;
    }
    else if(49<=a&&a<=50)
    {
        search=15;
    }
    else if(a==51)
    {
        search=16;
    }
    else if(a==52)
    {
        search=17;
    }
    else if(a==53)
    {
        search=18;
    }
    search_array[i]=search;//把要查的花卉由随机数对应到1-18种花里
    }
    double start=clock();
    for(int h=1;h<=N;h++)
    {
      for(int i=1;i<=18;i++)//顺序查找的过程
      {
        if(search_array[h]!=list[i].num)
        {
            i+=0;
        }
        else if(list[i].num==search_array[h])
        {
        soFile << list[i].num;
        soFile << "\t";
        soFile << list[i].intro;
        soFile << endl;

        }
     }
    }
    soFile.close();
     double end=clock();
    double diff=(double)(end-start)/CLOCKS_PER_SEC;
    cout<<endl<<diff;
     return 0;
}
```

法三：

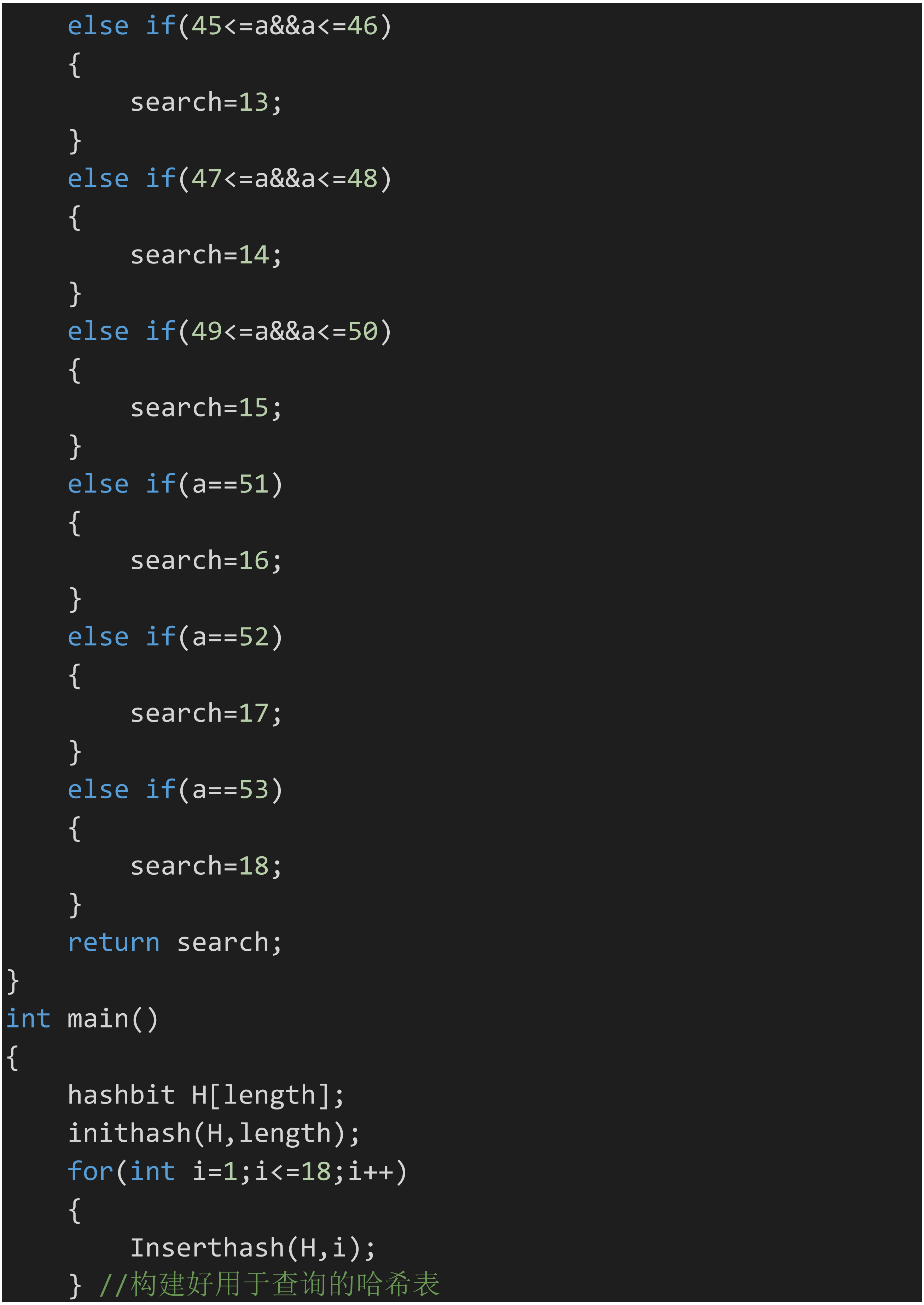
```
#include<bits/stdc++.h>
#include <fstream>
#include<iostream>
#include<string>
#include<math.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
using namespace std;
typedef struct hash
```

```
{
    int array;
    struct hash* next;
}hashnode,*hashbit;
typedef struct name{
    int num;
    char intro[10000000];
}NAME;
int search_array[10000000];//n 次查询每次对应查询的花卉编号数组
int length=7;
int hashvalue(int arr)//这里哈希函数的选定是：例如第 i 种花，K[i]即为其对应的搜索频率
{
    int value;
  if(arr==1)
   {
       value=6;
   }
   else if(2<=arr&&arr<=3)
   {
       value=5;
   }
   else if(4<=arr&&arr<=6)
   {
       value=4;
   }
   else if(7<=arr&&arr<=10)
   {
       value=3;
   }
   else if(11<=arr&&arr<=15)
   {
       value=2;
```



```
   }
   else if(16<=arr&&arr<=18)
   {
       value=1;
   }
    return value;
}

void inithash(hashbit H[],int length)//初始化
{
    for(int i=1;i<length;i++)
    {
        H[i]=new hashnode;
        H[i]->array=i;
        H[i]->next=NULL;
    }
}

void Inserthash(hashbit H[],int arr)
{
    int value;
     value=hashvalue(arr);
  hashbit present=new hashnode;
  hashbit add=new hashnode;
  add->array=arr;
  present=H[value];
   while(present->next!=NULL)
  {
    present=present->next;
  }
  present->next=add;
  add->next=NULL;

}
```

```
bool Search(hashbit H[],int arr)
{

    int value;
  value=hashvalue(arr);
  hashbit present=new hashnode;
   hashbit add=new hashnode;
    present=H[value];
    while(present->next!=NULL)
    {
     present=present->next;
     if(present->array==arr)
      return true;
    }
    return false;
}
int Search_flower(int a)
{
     int search=0;//对于生成的53个随机数分段用来模拟不同花卉
的不同查询频度，先初始化
    if(1<=a&&a<=6)
    {
        search=1;
    }
    else if(7<=a&&a<=11)
    {
        search=2;
    }
    else if(12<=a&&a<=16)
    {
        search=3;
    }
    else if(17<=a&&a<=20)
    {
        search=4;
    }
    else if(21<=a&&a<=24)
    {
        search=5;
    }
    else if(25<=a&&a<=28)
    {
        search=6;
    }
    else if(29<=a&&a<=31)
    {
        search=7;
    }
    else if(32<=a&&a<=34)
    {
        search=8;
    }
    else if(35<=a&&a<=37)
    {
        search=9;
    }
    else if(38<=a&&a<=40)
    {
        search=10;
    }
    else if(41<=a&&a<=42)
    {
        search=11;
    }
    else if(43<=a&&a<=44)
    {
        search=12;
    }
```



```
    else if(45<=a&&a<=46)
    {
        search=13;
    }
    else if(47<=a&&a<=48)
    {
        search=14;
    }
    else if(49<=a&&a<=50)
    {
        search=15;
    }
    else if(a==51)
    {
        search=16;
    }
    else if(a==52)
    {
        search=17;
    }
    else if(a==53)
    {
        search=18;
    }
    return search;
}
int main()
{
    hashbit H[length];
    inithash(H,length);
    for(int i=1;i<=18;i++)
    {
        Inserthash(H,i);
    } //构建好用于查询的哈希表
    int N;
    cin>>N;//查询次数
    NAME* list = new NAME[19];
    ifstream siFile("flower.txt");
    for(int i=1;i<=18;i++)//把花的相关文件放入一个数组里
    {
        siFile>>list[i].num;
        siFile.get(list[i].intro,1000000000,'\n');
    }
    siFile.close();
    ofstream soFile("output.txt");
     srand((unsigned)time(NULL));
    for(int i=1;i<=N;i++)
    {
    int a=rand()%53+1;
    search_array[i]=Search_flower(a);//构建出一个数组存放每次搜索花卉的编号
    }
     double start=clock();
    for(int h=1;h<=N;h++)
    {
     if(Search(H,search_array[h])==1)
    {
        soFile << list[search_array[h]].num;
        soFile << "\t";
        soFile << list[search_array[h]].intro;
        soFile << endl;
    }
    }
    soFile.close();
    double end=clock();
    double diff=(double)(end-start)/CLOCKS_PER_SEC;
    cout<<endl<<diff;
    return 0;
```

```
}
```