**《数据结构》实验报告（第一章）**

**学号：\_\_\_\_\_\_58122231\_\_\_ 姓名 ：\_\_\_\_\_\_陆文韬\_\_\_\_\_\_\_**

**实验题号： P69, 1; P72, 10\_ 实验日期：\_2023.10.24\_\_\_\_**

**实验一**

**1．问题描述：**描述实验内容和要求以及需要解决的问题。

这道题中涉及到两个函数，一个是二次函数，一个是指数函数，题目要求我们计算，随着n的增长，何时指数函数的数值会比二次函数大。

**2．算法思想：**详细描述解决相应问题所需要的算法设计思想。

在解决这个问题的过程中，我们分别计算两个函数的数值，，再比较两个函数数值的大小关系，最后通过n从1开始遍历来找到结果

**3．功能函数：**描述所设计的功能函数。如果有多个函数，需要描述它们之间的关系。

func1:计算二次函数的数值

func2:计算指数函数的数值

compare:比较两个函数数值的大小

**4．测试数据：**设计测试数据，或具体给出测试数据。

N从1开始递增

**5．测试情况：**给出程序的测试情况，分析运行结果，显示实验结果截图。

****

N从1开始增加，当n大于8时，第二个函数的值比第一个函数的值大

**6．实验总结：**写出实验过程中遇到的问题，以及问题的解决过程。分析算法的时间复杂度和空间复杂度，总结实验心得体会。

在程序中采用了while循环来遍历n，因此时间复杂度为O(n)

程序仅仅定义了一个变量n，n的大小为4字节，为空间复杂度

**7. 源代码：**给出项目所有源程序清单。

#include<iostream>

using namespace std;

//计算二次函数的数值

double func1(int n)

{

return n\*n;

}

//计算指数函数的数值

double func2(int n)

{

return pow(2,n)/4.0;

}

//比较两个函数的大小

int compare(int n)

{

if(func1(n)>func2(n))

return 1;

else if(func1(n)<func2(n))

return -1;

else

return 0;

}

int main()

{

int n=1;

while(compare(n)>0)

{

n++;

}

cout<<"当n大于"<<n<<"时第二个函数比第一个函数大。"<<endl;

return 0;

}

**实验二**

**1．问题描述：**描述实验内容和要求以及需要解决的问题。

数组为1-100的不同大小，对数组内的元素采用二分查找，列出不同数组大小的情况下，找到一个元素所需的时间

**2．算法思想：**详细描述解决相应问题所需要的算法设计思想。

首先设计一个二分查找的函数进行元素查找，因为单次查找的时间很短不利于测量，我们采用测多算少的方法。

**3．功能函数：**描述所设计的功能函数。如果有多个函数，需要描述它们之间的关系。

BinarySearch：参数为数组，要查找具体元素的数值，数组元素的个数

**4．测试数据：**设计测试数据，或具体给出测试数据。

N从1迭代到100

**5．测试情况：**给出程序的测试情况，分析运行结果，显示实验结果截图。

****

**6．实验总结：**写出实验过程中遇到的问题，以及问题的解决过程。分析算法的时间复杂度和空间复杂度，总结实验心得体会。

时间复杂度：O(n\*c\*m) n从1～100，c为对于每个n，测量次数的多少，m为数组元素的个数

空间复杂度：arr[100]为一个包含一百个元素的数组，每个元素占用4字节，数组占用400字节

**7. 源代码：**给出项目所有源程序清单。

#include<iostream>

#include<random>

#include<ctime>

using namespace std;

int BinarySearch(int \*a,const int x,const int n)

{

int left=0;

int right=n-1;

while(left<=right)

{

int middle = (left+right)/2;

if(x<a[middle])right = middle-1;

else if(x>a[middle])left= middle+1;

else return middle;

}

return -1;

}

int main()

{

int arr[100];

int n;

srand(time(NULL));

for(n=1;n<=100;n++)

{

for(int i=0;i<n;i++)

{

arr[i]=i+1;

}

clock\_t start,end;

start = clock();

for(int count=1;count<=1000000;count++)

{

int i = rand()%n;

BinarySearch(arr,i+1,n);

}

end = clock();

cout<<n<<' '<<double(end-start)/CLOCKS\_PER\_SEC<<"s"<<endl;

}

}