二分查找

因为计算机是二进制构建的计算系统，所以与2有关的思想，常常运用在算法里。

将问题规模减半，称为二分；将计算过程速度加倍，称为倍增。

1、核心思想

二分查找是一种算法，其输入是一个有序的元素列表（必须是有序的），如果查找的元素包含在列表中，二分查找返回其位置，否则返回-1

比如说有一个1-100的数字，我随机的选择其中一个数字（假设为60），你需要以最少的次数猜到我所选择的数字，每次猜测后，我会告诉你大了，小了，对了。

```
假设你第一次从1开始猜，小了

第二次：2   小了

第三次：3   小了

.....

第五十九次：59 小了

第六十次：60 对了
```

这是简单的查找，每次猜测只能排除一个数字，如果我想的数字是100，那么你可能需要从1猜到100了！

那么有没有更好的查找方式呢？

答案当然是有的。

如果我选的数字是60

```
第一次：你从50开始猜，那么我告诉你小了，就排除了接近一半的数字，因为你至少知道1-50都小了

第二次：你猜75，那么我告诉你大了，这样剩下的数字又少了一半！或许你已经想到了，我们每次猜测都是选择了中间的那个数字，从而使得每次都将余下的数字排除了一半。

第三次：接下来，很明显应该猜测63，大了

第四次：然后你猜56，小了

第五次：然后你猜59 小了

第六次：猜测61，大了

第七次，你就能很明确的告诉我，答案是60！
```

这样的查找方式，很明显比第一种要高效很多。第一种需要猜测60次才能猜出正确答案，而使用第二种方式，只需要七次就能猜出正确答案

或许看到这里你已经明白了，这就是二分查找的方法。为什么二分查找要求有序，从这里也可以看出来。一般而言，对于包含n个元素的列表，用二分查找最多需要logn步，而简单查找最多需要n步。

2、二分查找样例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  int a[100];//注意这里的数组下标，即a[0]=1,a[1]=2……a[99]=100
  int guess;//猜测字符
  int flag=0;//设置标志位，区分是否查找成功
  int count=0;//统计比较次数
  int low=0,mid,high=99;
  //初始化
  cout<<"1、初始化"<<endl;
  for(int i=0;i<100;i++){
      a[i]=i+1;
  }
  cout<<"2、要查找的数字"<<endl;
  cout<<"guess:";
  cin>>guess;
  cout<<"3、二分查找"<<endl;
  //二分查找核心代码
  while(low<=high){
      count++;
      mid=(low+high)/2;
     cout<<"第"<<count<<"次查找，其中low="<<low<<"   high="<<high<<"   mid="
<<mid<<endl;
     if(guess==a[mid]){
         flag=1;
         cout<<"success！比较次数:"<<count<<"次"<<endl;
         break;//查找成功就退出，如果想要继续查找也是可以的
     }
     if(guess>a[mid]){
         low=mid+1;
     }
     if(guess<a[mid]){
          high=mid-1;
     }
  }
  if(flag==0)
      cout<<"fail！"<<endl;
  return 0;
}
```