**2.二分查找：**

//二分查找只能查找有序列

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Search\_Bin(int \*a,int key,int n)

{

int low=0,mid,high=n-1;

while (low<=high)

{

mid=low+(high-low)/2;//这样写可以防止数据溢出

if(key<a[mid]) high=mid-1;

else if (key>a[mid]) low=mid+1;

else return mid+1;

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**3.插值查找：**

//插值查找是二分查找的改进

//背记插值公式就是了

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Search\_Insert(int \*a,int key,int n)

{

int low=0,mid,high=n-1;

while (low<=high)

{

mid=low+(high-low)\*((key-a[low])/(a[high]-a[low]));

if(key<a[mid]) high=mid-1;

else if (key>a[mid]) low=mid+1;

else return mid+1;

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**4.斐波那契查找**

//其实这种方法也是插值法的变种

//也是在mid上面做文章，在黄金比例中分割数组

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*斐波那契数组\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Fib\_num(int i)

{

if(i<2) return i==0?0:1;

return Fib\_num(i-1)+ Fib\_num(i-2);

}

int \* Fibonacciin(int n)//函数不能返回一个数组，可以用全局变量和堆空间解决

{

int \*F=(int\*)malloc(n\*sizeof(int));//并不推荐这样做，最好是void(int\*F,int n)

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

{

F[i]=Fib\_num(i);

}

return F;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*斐波那契查找\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Search\_Fib(int \*a,int key,int n)

{

int \* Fib;

Fib=Fibonacciin(n);

int low=0,high=n-1,mid,i,k=0;

//以Fiber(k-1)-1:Fiber(k-2)-1精心分割

//为什么是Fiber(k)-1:因为有F(k)=F(k-1)+F(k-2)--->F(k)-1=(F(k-1)-1)+(F(k-2)-1)+1

while (Fib[k]-1<n) k++;//找到斐波数列最接近n的那一个下标

//扩展数组

for(i=n;i<Fib[k]-1;i++) a[i]=a[n];

while (low<=high)

{

mid=low+Fib[k-1]-1;

if(key<a[mid])

{

high=mid-1;

k=k-1;

}

else if (key>a[mid])

{

low=mid+1;

k=k-2;

}

else

{

if(mid<=n)return mid+1;

else return n;//如果都不是那就是最后一个的

}

}

free(Fib);

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/