普通的顺序查找算法的代码

int Sq\_search(int \*a,int n,int key)

{

int i;

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

{

if(a[i]==key)

return i;

}

return 0;

}

这时候的效率为：2n次判断

question:如何将2n化为n？

我的思路：

1. 我在想，本质就是一坨数据，然后要一个一个匹配，是否=key,
2. 能不能贪心一下？跳跃式？不过这样的话，理论上还是要2n，所以我们考虑问题，贪心思想是排除了最坏情况的理想条件下的
3. 要将时间复杂度缩短，可以分配给空间呀！就是我循环少循环一次，但是每次循环就判断两次这样，不过好像偏题了，人家要的是减少你的判断次数，甚至要求降到一半
4. 既然是问题是在判断次数上，首先我想着看看哪些判断是不可避免的 不可避免的是对a[i]=key的判断，然后，那么，对i<n这个判断能去掉吗？我想来想去想不出来，好像怎么样都不能把i<n这个去掉，所以放弃

答案：

int Sq\_search(int \*a,int n, int key)

{

a[0] = key;

while(a[i]!=key)

{

1. -;

}

return i;

}

我的总结：

1. 没错，问题就是集中在如何消去i<n的判断上，
2. 它用的是，既然我的必要判断是a[i]=key,我能不能用while来一次性让它一直走呀走，本来a[i]=key这个判断条件呢，它是静态的，而我需要一个动态判断，将静态判断升级为动态判断的方法就是直接拿它做循环的判断条件，但是for为啥不行呢？真的不行吗？
3. 我想到用while，让它一直判断循环，但是while需要一个跳出循环的条件，我当时没想到是利用a[0]==key,刚好利用了必要判断本身，没错，发挥了这个判断的价值，实现一个判断解决了
4. 所以我们，对于问题2n化n,我们应该想到只需一个判断条件，如何用一个判断条件实现本来人家要两个判断条件的算法，首先先找出哪个是我们的必要判断条件，然后再挖掘如何发挥这个判断条件的剩余价值