**HW**

Given S[1..n], find an index k such that S[k]=k.

(An array S has items in sorted order. All items are different. If there are many such items, it is okay to find any one of such items.)

You have to describe your algorithm and analyze the complexity of your algorithm.

Submit **search2001571010.doc** or **search2001571010.docx** file, if your student ID is 2001571010.

Index bs(index low, index high){

Index mid;

If (low>high) return 0;

else{

mid=(low+high)/2;

if (s[mid]==mid) return mid;

else if (s[mid]<mid)

return bs(mid+1, high)

else return bs(low, mid-1)

}

}

만약 s[mid]<mid 이면 s 안의 모든 값은 정수이기 때문에 s[mid-1]<=s[mid]-1을 만족한다. 따라서 s[mid-1]<=s[mid]-1<mid-1, s[mid-1]<mid-1 이므로 s[mid]!=mid-1 이다. 따라서 귀납적으로 1 인덱스부터 mid 인덱스까지 조건을 만족하지 못하기 때문에 탐색할 필요가 없다. 하지만 mid+1 인덱스부터는 조건을 만족하는 인덱스가 존재할 수 있으므로 bs(mid+1, high)를 반환한다.

같은 방법으로 s[mid]>mid 일 때는 bs(low, mid-1)를 반환한다.

Worst-case: 만족하는 인덱스 k가 존재하지 않을 경우

함수 w(n)을 인덱스의 범위가 1~n인 경우에 최악의 상황에서 시행되는 연산의 수라고 하고 w(1)은 a라고 하자

계산의 편의를 위해 n=2^k라고 가정하자. 그럼 k=log n 이다.

W(n)=w(n/2)+a를 만족하므로

W(2^k) = w(2^(k-1))+a

= (w(2^(k-2))+a)+a

=w(2^(k-2))+2\*a

=w(2^0)+k\*a

=k\*a + a

=a\*log n +a

따라서 Big-O 표기법에 의해 시간복잡도는 O(log n)이다.

위 식이 n=1, 2일 때 성립하기 때문에 1~n-1까지 모두 성립한다고 가정하고 n일 때 만족함을 보이자.

W(n) =w(n/2) +a

=(a\*log n/2 + a) +a

=(a\*log n -a +a) +a

=a\*log n + a

따라서 모든 n에 대해 만족한다. 따라서 위 알고리즘의 시간복잡도는 O(log n)이다.