**김희철 교수님**

**201900776**

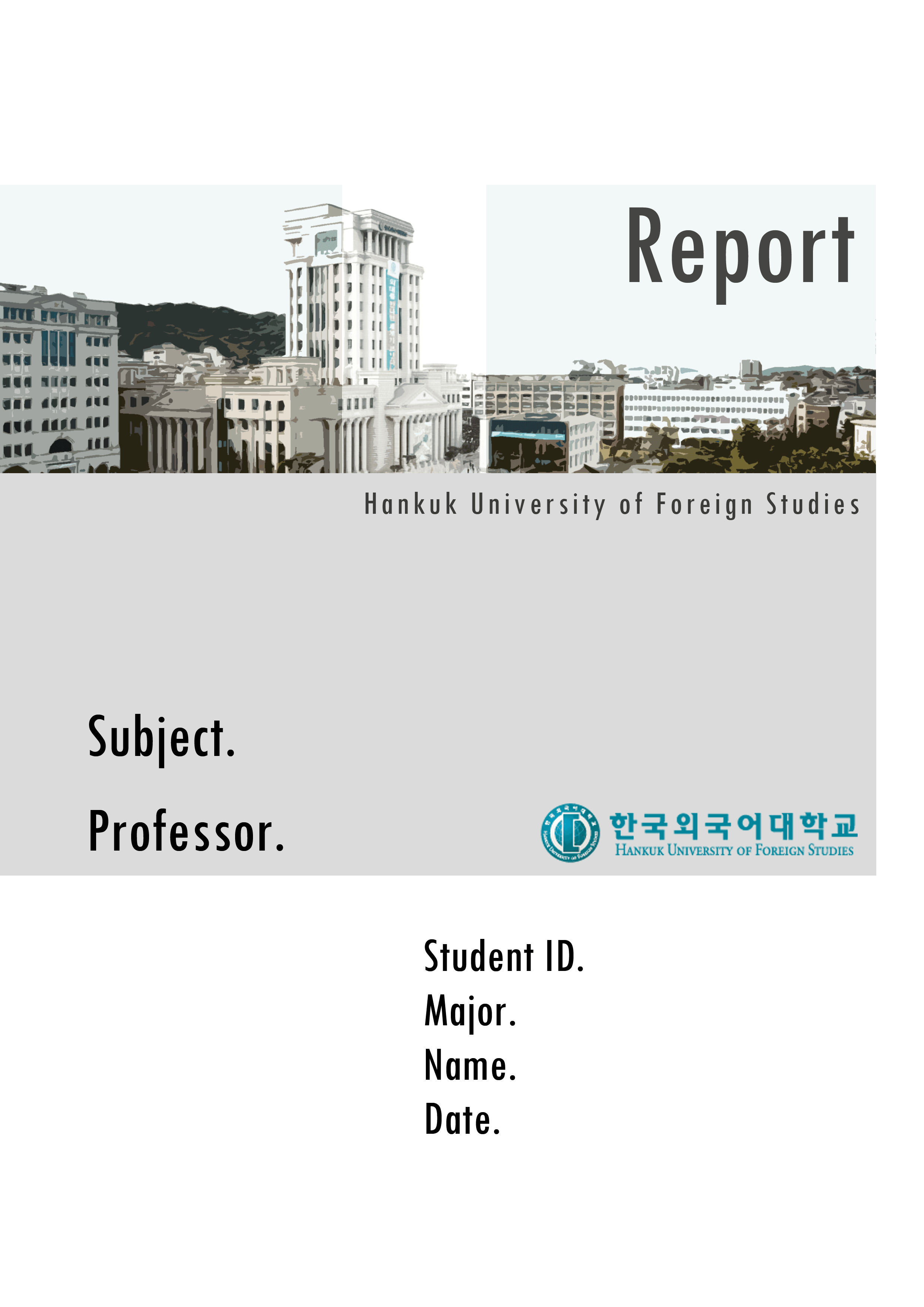
**2022년 09월 21일**

**김보석**

**컴퓨터전자시스템공학**

**알고리즘**

**프로그래밍과제 1**

****

**2. 문제 기술**

오름차순으로 정렬된 n(2이상 100,000이하 정수)개의 수(1,000,000,000이하 정수) 중 K(정수)와 가장 가까운 수를 찾는 프로그램을 작성하시오. K와 가장 가까운 수가 여러 개일 경우, 이들을 오름차순으로 출력하시오.

- 입력된 n개의 수에서 입력된 k의 값과 가장 인접한 수를 출력하는 문제이다. 인접한 수가 2개 일 경우 2개 모두 출력한다.

**3. 알고리즘 및 자료구조**

**알고리즘**

**If k가 n개의 수의 최대값보다 클 경우**

**N개의 수 중 최대값 출력**

**Else if k가 n개의 수의 최소값보다 작을 경우**

**n개의 수 중 최소값 출력**

**else**

**이진탐색 함수 호출**

**--------------------------**

**이진탐색 함수**

**int left = 0;**

**int right = *n*-1;**

**int mid;**

***while*(left <= right){**

**mid = (left + right) / 2;**

***if* (*key* == *list*[mid]){**

***return* mid;}**

***else* *if* (*key* < *list*[mid]){**

**right = mid - 1;**

**}**

***else***

**left = mid + 1;**

**}**

***return* mid;**

**----------------------------------------------**

**if(arr[mid] == k)**

**arr[mid] 출력**

**else if(arr[mid] <k)**

**arr[mid]와 arr[mid+1] 중 k와 거리가 더 가까운 값을 출력**

**거리가 같은 경우**

**Arr[mid], arr[mid + 1] 모두 출력**

**else**

**arr[mid]와 arr[mid-1] 중 k와 거리가 더 가까운 값을 출력**

**거리가 같은 경우**

**Arr[mid], arr[mid - 1] 모두 출력**

**자료구조**

배열 자료구조가 쓰였다.

**4. 시간복잡도**

이진탐색 함수가 가장 많이 쓰이는 기본 연산이므로

이진 탐색의 시간 복잡도는 연산 한번 할 때 마다, n의 값이 절반 씩 줄어드므로

log n이다.(밑은 2)

따라서 O(logn)

**5. 느낀점**

이진 탐색 트리를 이용해서 mid 값을 리턴하는 것은 수업 때 배운 내용을 기억하면서 해결했다. 문제는 그 다음, 절댓값 크기 비교하는 데서 애를 먹었다. 먼저 절댓값 함수를 인터넷으로 조사해야 했고, arr[mid]가 k보다 큰 경우와 작은 경우를 나눠서 생각해야 했고, 또, 같을 경우도 생각해야 했다.

뭔가 더 간단하게 구현할 수 있을 것 같은데, 아직 문제를 많이 풀어보지 않아서 그런지 좋은 방법이 생각나지 않아서 아쉬었다. 실력을 더 키워야 겠다.

**6. 프로그램 코드**

***#include* <iostream>**

***#include* <cstdlib>**

***#include* <algorithm>**

**using namespace std;**

**int binarySearch(int \**list*, int *key*, int *n*){ *//이진 탐색 함수 list배열포인터, 입력된 key 값, 입력된 수의 개수를 매개변수로 받는다.***

**int left = 0; *//left는 0***

**int right = *n*-1; *//right는 n - 1***

**int mid; *//mid 선언***

***while*(left <= right){ *//right가 left보다 클 경우 실행***

**mid = (left + right) / 2; *//mid는 left와 right를 2로 나눈 몫이다.***

***if* (*key* == *list*[mid]){ *//  key값이 list[mid]와 같을 경우 mid 반환***

***return* mid;}**

***else* *if* (*key* < *list*[mid]){ *//key값이 list[mid]보다 작을 경우 right = mid - 1로 조사 범위를 좁힌다.***

**right = mid - 1;**

**}**

***else  //key 값이 list[mid]보다 클경우 left = mid + 1로 조사 범위를 좁힌다.***

**left = mid + 1;**

**}**

***return* mid; *//right 가 left보다 작아지면 mid 반환***

**}**

**int main(){**

**int mid; *//mid 선언***

**int result1;**

**int result2; *// 출력값이 두개 일 때 사용***

**int n; *//입력된 수의 개수***

**cin >> n;**

**int \*arr = new int[n]; *//n크기의 배열 선언***

***for*(int i; i < n; i++){ *//입력된 수로 배열 초기화***

**cin >> arr[i];**

**}**

**int k; *//입력된 key값***

**cin >> k;**

**int minVal = \*min\_element(arr, arr + n); *//입력된 수중 최댓값***

**int maxVal = \*max\_element(arr, arr + n); *//입력된 수중 최소값***

***if* (k > maxVal) *//key값이 maxVal보다 클 경우 maxVal 출력***

**cout << maxVal;**

***else* *if* (k < minVal) *//key값이 minVal보다 작을 경우 minVal 출력***

**cout << minVal;**

***else*{ *//k가 minval <= k <=maxVal 일 경우***

**mid = binarySearch(arr, k, n); *//이진탐색함수 호출: key 값과 가장 인접한 값이 있는 인덱스 mid 반환***

***if*(arr[mid] == k){ *//arr[mid]가 key값과 일치할 때 arr[mid]출력***

**result1 = arr[mid];**

**cout << result1;}**

***else* *if*(arr[mid] < k){*//arr[mid]가 key값보다 작을 경우 실행***

**int self = abs(arr[mid] - k); *//arr[mid]와 key값 과의 거리***

**int big = abs(arr[mid + 1] - k); *//arr[mid + 1]과 key 값 과의 거리***

***if*(self < big){ *//self가 big보다 작을 경우 arr[mid] 출력***

**result1 = arr[mid];**

**cout << result1 << endl;}**

***else* *if*(self == big){ *//self와 big이 같은 경우 arr[mid]와 arr[mid + 1] 모두 출력***

**result1 = arr[mid];**

**result2 = arr[mid + 1];**

**cout << result1 << result2 << endl;**

**}**

***else*{ *//self가 big보다 클 경우 arr[mid + 1] 출력***

**result1 = arr[mid + 1];**

**cout << result1;**

**}**

**}**

***else* *if*(arr[mid] > k){*//arr[mid]가 key값보다 클 경우 실행***

**int self = abs(arr[mid] - k);*//arr[mid]와 key값 과의 거리***

**int small = abs(arr[mid - 1] - k);*//arr[mid - 1]과 key 값 과의 거리***

***if*(self < small){*//self가 small보다 작을 경우 arr[mid 출력]***

**result1 = arr[mid];**

**cout << result1 << endl;**

**}**

***else* *if*(self == small){*//self와 small이 같은 경우 arr[mid]와 arr[mid - 1] 모두 출력***

**result1 = arr[mid - 1];**

**result2 = arr[mid];**

**cout << result1 << " " <<result2 << endl;**

**}**

***else*{ *//self가 small 보다 클 경우 arr[mid - 1] 출력***

**result1 = arr[mid - 1];**

**cout << result1;}**

**}**

**}**

***return* 0;**

**}**

**문제 2번과 문제 3번을 해결하지 못했습니다. 죄송합니다.**