자료구조

실습과제 02

과목명 : 자료구조

담당 교수 : 박정희

분반 : 06

학번 : 202002546

이름 : 임우진

제출일자 : 2023.03.20.(월)

목차

1. 문제에서 작성 요구한 내용

순차 탐색과 이진 탐색을 구분하여 두 탐색 알고리즘의 **시간 복잡도**를 측정하여 비교하는 것이 요구사항이다.

$n = 2^3, 2^4, 2^5, 2^6, 2^7, 2^8$ 의 크기를 가진 배열에서

목표 값을 찾기 위해 두 탐색 알고리즘을 사용한다.

순차 탐색에서는 난수로 이루어진 배열 사이에서 목표 값을

찾아 인덱스를 출력하고

이진 탐색에서는 오름차순 정렬된 배열 사이에서 목표 값을 찾아 인덱스를 출력해야 한다.

시간 복잡도를 측정하기 위해 **count변수**를 생성하고

알맞은 위치에서 증가시켜 총 연산의 수를 얻어내야 한다.

2. 그 외 풀이 과정 및 코드 설명

```java
import java.util.*;
0개의 사용위치
public class MainJava {
    0개의 사용위치
    public static void main(String[] args) {

        int[] n_arr = {8, 16, 32, 64, 128, 256, 512, 1024}; //n배열

        for (int i = 0; i < 8; i++) {

            int[] seq_result = new int[2];
            int[] bin_result = new int[2];
            int[] bin_arr = new int[n_arr[i]];
            int[] seq_arr = new int[n_arr[i]];

            for (int j = 0; j < bin_arr.length; j++) {
                bin_arr[j] = j;
            }
            Random ran = new Random();
            for (int j = 0; j < seq_arr.length; j++) {

                int num = ran.nextInt(n_arr[i]);
                seq_arr[j] = num;
            }

            int tar = ran.nextInt(n_arr[i]);

            seq_result = seq_search(seq_arr, tar);
            bin_result = bin_search(bin_arr, tar);

            System.out.println("n= " + n_arr[i]);
            System.out.println("목표 값: " + tar);
            System.out.println("순차탐색(count, index): (" + seq_result[0] + ", " + seq_result[1] + ")");
            System.out.println("이진탐색(count, index): (" + bin_result[0] + ", " + bin_result[1] + ")");
            System.out.println("= = = = = = = = = = = = = = = = = = = = = = = =");
        }

    }
```

MainJava 클래스

main함수

정수 배열 n_arr

8번 반복

정수 배열 seq_result, bin_result, seq_arr, bin_arr 선언

bin_arr : 오름차순대로 배열 내 값 초기화

seq_arr : 난수 생성 후 배열 내 값 초기화

목표 값인 정수 tar을 난수로 초기화

seq_result를 seq_search함수에 seq_arr과 tar를 넣어 반환된 배열로 초기화

bin_result를 bin_search함수에 bin_arr과 tar를 넣어 반환된 배열로 초기화

출력 양식에 맞게 출력

```java
public static int[] seq_search(int[] array, int target) {
    int lng = array.length;
    int[] result = new int[2];

    for(int i = 0; i < lng; i++){
        result[0]++;
        if(array[i]==target){
            result[1] = i;
            return result;
        }

    }
    result[1]=-lng;
    return result;
}
```

seq_search 함수

정수 lng를 배열 길이로 초기화

정수 배열 result[2] : 첫 값 = count, 두번째 값 = 목표 인덱스

lng회 반복

count 증가

목표 값 발견 시, result[1]에 인덱스를 초기화하고 result 리턴

목표 값 탐색 실패 시, result[1]에 -lng 초기화하고 result 리턴

```java
public static int[] bin_search(int[] array, int target){
    int lng = array.length;
    int low = 0;
    int high = lng - 1;
    int mid;
    int[] result = new int[2];
    while(low<=high){
        mid = (low+high)/2;
        if(target==array[mid]){
            result[1] = mid;
            return result;
        } else if (target>array[mid]) {
            low = mid + 1;
            result[0]++;
        } else {
            high = mid - 1;
            result[0]++;
        }

    }
    result[1] = -lng;
    return result;
}
}
```

bin_search 함수

정수 lng, low, high 값을 각각 배열길이와 0과 lng-1로 초기화

정수 배열 result[2] : 첫 값 = count, 두번째 값 = 목표 인덱스

low가 high보다 작거나 같을 동안 반복

mid = (low+high)/2로 초기화

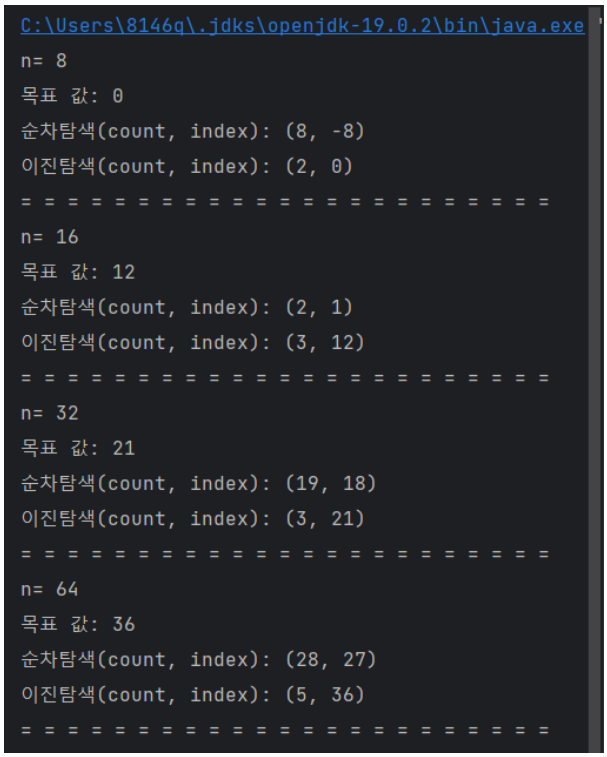
mid를 목표값을 비교하여 같으면 result[1] = mid로 초기화

작으면 low값을 mid-1로 초기화하고 count증가

크면 high값을 mid+1로 초기화하고 count증가

목표 값 탐색 실패 시, result[1]에 -lng 초기화하고 result 리턴

3. 실행 화면 캡쳐



```
C:\Users\8146q\.jdks\openjdk-19.0.2\bin\java.exe
n= 8
목표 값: 0
순차탐색(count, index): (8, -8)
이진탐색(count, index): (2, 0)
= = = = = = = = = = = = = = = = = = = = = = = = =
n= 16
목표 값: 12
순차탐색(count, index): (2, 1)
이진탐색(count, index): (3, 12)
= = = = = = = = = = = = = = = = = = = = = = = = =
n= 32
목표 값: 21
순차탐색(count, index): (19, 18)
이진탐색(count, index): (3, 21)
= = = = = = = = = = = = = = = = = = = = = = = = =
n= 64
목표 값: 36
순차탐색(count, index): (28, 27)
이진탐색(count, index): (5, 36)
= = = = = = = = = = = = = = = = = = = = = = = = =
```

```
n= 128
목표 값: 50
순차탐색(count, index): (29, 28)
이진탐색(count, index): (6, 50)
= = = = = = = = = = = = = = = = = = = = = = = = =
n= 256
목표 값: 113
순차탐색(count, index): (27, 26)
이진탐색(count, index): (6, 113)
= = = = = = = = = = = = = = = = = = = = = = = = =
n= 512
목표 값: 324
순차탐색(count, index): (353, 352)
이진탐색(count, index): (8, 324)
= = = = = = = = = = = = = = = = = = = = = = = = =
n= 1024
목표 값: 209
순차탐색(count, index): (928, 927)
이진탐색(count, index): (8, 209)
= = = = = = = = = = = = = = = = = = = = = = = = =

종료 코드 0(으)로 완료된 프로세스
```

4. 알고리즘 비교표 및 시간 복잡도 계산

Case 1 :

n	순차 탐색	이진 탐색
2^3	8	2
2^4	2	3
2^5	19	3
2^6	28	5
2^7	29	6
2^8	27	6
2^9	353	8
2^{10}	928	8

Case 2 :

n	순차 탐색	이진 탐색
2^3	8	1
2^4	16	3
2^5	11	4
2^6	64	4
2^7	1	5
2^8	256	3
2^9	8	8
2^{10}	406	8

Case 3 :

n	순차 탐색	이진 탐색
2^3	8	2
2^4	2	4
2^5	19	3
2^6	64	5
2^7	128	4
2^8	170	5
2^9	512	8
2^{10}	1024	9

Case 4 :

n	순차 탐색	이진 탐색
2^3	4	2
2^4	1	3
2^5	32	2
2^6	16	5
2^7	128	6
2^8	256	5
2^9	447	8
2^{10}	760	9

순차 탐색 : 시간 복잡도 = O(n)

n이 증가할수록 연산횟수가 증가한다.

최악의 경우 모든 인덱스를 탐색해야하므로 n회 연산을 실행한다.

이진 탐색 : 시간 복잡도 = $O(\log n)$

n의 크기에 비례하지 않고 순차 탐색보다 효율적으로 탐색한다.

최악의 경우 $\log_2 n$회 연산을 실행한다.