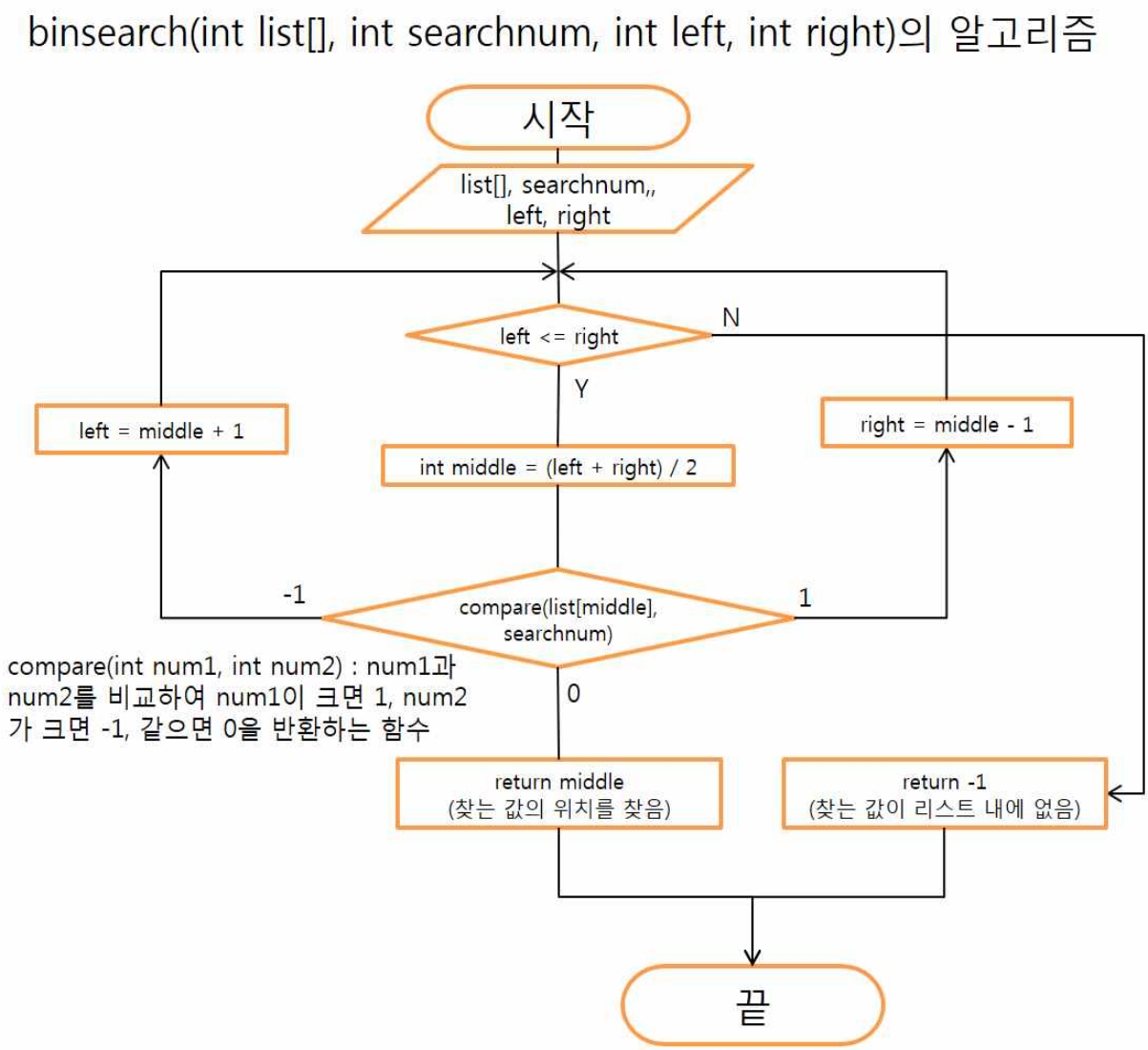
2. 이진검색

21410785 황 희

1. 개요

- 이진검색 [binsearch(int list[], int searchnum, int left, int right)] 의 알고리즘



binsearch함수는 찾는 값이 리스트의 몇 번째 인덱스에 있는 지 찾는 함수이다. 우선 매개변수로 리스트와 찾을 값, 리스트의 첫 번째 인덱스와 마지막 인덱스를 받는다. 이해하기 쉽도록 첫 번째 인덱스는 left, 마지막 인덱스는 right라 하겠다. 먼저, left가 right보다 큰지 검사한다. 크지 않다면 middle변수를 선언하여 (left + right)/2로 치환해준다. 그 다음, list[middle]값과 찾는 값을 비교하여 만약 찾는 값이 middle위치의 값보다 작다면 right를 middle - 1로 치환한다. 만약 찾는 값이 middle위치의 값보다 크다면 left를 middle + 1로 치환한다. 같다면 middle위치의 값이 찾는 값과 같은 값이므로 middle을 반환해준다. list[middle]이 찾는 값과 같아질 때까지 계속 반복해준다. 그러다가 left가 right보다 커진다면 찾는 값이 리스트에 없다는 뜻이므로 -1을 리턴해준다.

2. 본문

- 코드(설명은 주석으로 대체)

```
#include <stdio.h>
#include<stdlib.h>
#define MAX_SIZE 101 // 배열 최대크기 101로 설정
#define SWAP(x,y,t) ((t)=(x), (x)=(y), (y)=(t)) // x와 y를 바꿔주는 swap함수
void sort(int[], int); // sort함수 선언
int compare(int x, int y) // x와 y를 비교해서 -1이나 0이나 1리턴
{
        if (x < y) return -1; // y가 x보다 크면 -1 리턴
        else if (x == y) return 0;  // x와 y가 같으면 0 리턴
        else return 1; // x가 y보다 크면 1 리턴
}

int binsearch(int list[], int searchnum, int left, int right) // 이진검색 함수
{
        int middle;
        while (left <= right) { // left 인덱스가 right 인덱스를 넘어가면 중단
                middle = (left + right) / 2; // middle 인덱스를 (left+right)/2 로 지정
                switch (compare(list[middle], searchnum)) {
                        // compare함수를 이용해서 찾는 값과 비교하여 그에따른 결과실행
                case  -1: left = middle + 1;
                        // middle보다 찾는값이 크므로 left인덱스를 middle + 1 인덱스로 설정
                        break;
                case   0: return middle;
                        // middle == searchnum이므로 middle인덱스 리턴
                case   1: right = middle - 1;
                // middle보다 찾는값이 작으므로 rigth인덱스를 middle - 1인덱스로 설정
                }
        }
        return -1; // searchnum이 list안에 없는 것으로 판단, -1리턴
}

void main(void)
{
        int i, n, list[MAX_SIZE];
        printf("Enter the number of numbers to generate: ");
        scanf("%d", &n); // 랜덤한 수로 배열을 만들기 위해 배열크기 입력
        if (n<1 || n>MAX_SIZE) { // 배열크기가 1보다 작거나 max size보다 크므로 적절x
                fprintf(stderr, "Improper value of n\n");
                exit(1);
        }
        for (i = 0; i<n; i++) { // 배열 초기화
                list[i] = rand() % 1000; // 랜덤값으로 배열초기화
```

```
                printf("%d  ", list[i]);
        }
        sort(list, n); // 배열을 정렬함
        printf("\n Sorted array:\n ");
        for (i = 0; i<n; i++) // 정렬된 배열 출력
                printf("%d  ", list[i]);
        printf("\n");
        int search;
        printf("찾을 숫자 입력 : ");
        scanf("%d", &search); // 인덱스를 찾고자하는 수 입력
        printf("%d라는 숫자는 %d번째에 있습니다(binary search).\n", search, binsearch(list,
search, 0, n - 1) + 1);
        // binsearch함수 이용, 찾는값 인덱스 찾기
}
void sort(int list[], int n)// 리스트를 정렬하는 함수
{
        int i, j, min, temp; // sort에 필요한 변수 선언
        for (i = 0; i<n - 1; i++) { // i가 n-1보다 작을동안
                min = i; // min을 i로 초기화
                for (j = i + 1; j<n; j++) // j를 i+1로 초기화시키고, n보다 작을동안
                        if (list[j]<list[min]) // list[j]가 list[min]보다 작다면
                                min = j; // min인덱스에 j인덱스 입력
                SWAP(list[i], list[min], temp); // list[i]과 list[min]교환
        }
}
```

- 결과창

```
Enter the number of numbers to generate: 20
41  467  334  500  169  724  478  358  962  464  705  145  281  827  961  491  9
95  942  827  436
 Sorted array:
 41  145  169  281  334  358  436  464  467  478  491  500  705  724  827  827
942  961  962  995
찾을 숫자 입력 : 478
478라는 숫자는 10번째에 있습니다(binary search).
계속하려면 아무 키나 누르십시오 . . .
```

먼저 배열의 크기를 20으로 지정해주고 무작위의 수가 들어있는 배열을 생성하였다. 정렬이 되어있지 않으면 이진 검색이 불가능하므로 정렬을 해주었고, 찾는 숫자를 입력하자 binsearch함수에 의해 찾는 값이 존재하는 인덱스를 반환해주었다.

3. 결론

이진검색은 리스트에서 원하는 숫자를 찾기 편리하다. 선형검색은 배열의 처음부터 끝까지 일일이 찾는 숫자를 비교하여야하기 때문에 찾고자 하는 값이 배열의 끝에 있으면 그만큼 처리속도가 늘어나지만, 이진검색은 middle위치에 있는 값보다 크거나 작으면 left나 right인덱스를 당김으로써 검색해야하는 배열을 계속 반으로 줄여주기 때문에 선형검색에 비해 처리속도도 훨씬 빠르고, 그만큼 메모리의 사용도 적기 때문에 더 효율적이다. 하나 주의할 점은, 이진검색을 실행하기 위해서는 반드시 정렬 된 배열이 필요하다는 것이다. 정렬이 되어있지 않다면 찾는 값과 middle위치에 있는 값과의 비교는 의미가 없어진다.

2017-12-06

황 희