1-1 이진탐색

```c
#include <stdio.h>

int bsearch(int arr[], int key, int beg, int end);
int main()
{
    int a[] = {10, 20, 25, 35, 45, 55, 60, 75, 80, 90, 95, 120};
    int beg = 0, end, mid, key;
    key = 55;
    end = sizeof(a) / sizeof(int);

    scanf("%d", &key);
    mid = bsearch(a, key, beg, end);

    if (mid == -1)
        printf("%d is not found\n", key);
    else
        printf("A[%d] = %d\n", mid, a[mid]);

}
int bsearch (int arr[], int key, int beg, int end)
{
    int mid=(beg+end)/2;
    if (end-beg <= 1)
        return arr[mid] == key ? mid : -1;
    if(key < arr[mid])
        return bsearch(arr, key, beg,  mid);
    else
        return bsearch(arr, key, mid, end);
}
```

문제 출력 디버그 콘솔 터미널 포트

```
202020827@cslinux2:~/com/2$ gcc bsearch.c -o bsearch
202020827@cslinux2:~/com/2$ ./bsearch
35
A[3] = 35
202020827@cslinux2:~/com/2$ ./bsearch
22
22 is not found
202020827@cslinux2:~/com/2$
```

1-2. pointer를 이용한 버전 실험

```c
#include <stdio.h>

int *bsearch(int *arr, int key, int n);

int main() {
    int a[] = {10, 20, 25, 35, 45, 55, 60, 75, 80, 90, 95};
    int *aptr, key;
    int end = sizeof(a) / sizeof(int);
    puts("Enter key: ");
    scanf("%d", &key);
    aptr = bsearch(a, key, end);
    if (aptr == NULL)
        printf("%d is not found\n", key);
    else
        printf("A[]=%d,%d\n", *aptr , (int) (aptr-a) ) ;
}

int *bsearch(int *arr, int key, int n) {
    int mid = n / 2;
    if (n == 1)
        return arr[0] == key ? arr : NULL;
    if (key < arr[mid])
        return bsearch(arr, key, mid);
    else
        return bsearch(arr + mid, key, n - mid);
}
```

문제 출력 디버그 콘솔 터미널 포트

```
202020827@cslinux2:~/com/2$ gcc pointer.c -o pointer
202020827@cslinux2:~/com/2$ ./pointer
Enter key:
10
A[]=10,0
202020827@cslinux2:~/com/2$ ./pointer
Enter key:
25
A[]=25,2
202020827@cslinux2:~/com/2$ ./pointer
Enter key:
23
23 is not found
202020827@cslinux2:~/com/2$
```

- **광수의 아이디어에 따라 3 개의 동전 중 가짜 동전을 찾을 때 몇 번에 찾을 수 있는가?**

  - 3 개의 동전 중에서 가짜 동전을 찾는 방법은 한 번의 비교로 찾을 수 있다. 세 동전 중 두 개를 비교하고, 나머지 하나가 가짜인지를 확인하는 방식, 따라서 **1 번**에 찾을 수 있다.
  - 정답 : 1 번

- **9 개의 동전에서 교재에서 제시한 방법으로 가짜 동전을 찾는다면 최악의 경우 몇 번에 찾을 수 있는가?**

  - 교재에서 제시한 방법은 3 개씩 나누어 비교하는 방법으로, 9 개의 동전은 첫 번째 비교에서 3 개씩 나누고, 그 중 가짜 동전이 포함된 그룹을 찾아가며 반복한다.
  - 최악의 경우 **2 번**에 가짜 동전을 찾을 수 있다. 첫 번째 비교에서 3 개씩 나누고, 두 번째 비교에서 나머지 그룹을 다시 비교하면 되므로 2 번의 비교로 찾을 수 있다.
  - 정답 : 2 번

- **광수의 아이디어를 변형하여 9 개를 3 개씩 3 등분하여 찾는 방법으로 바꾸면 몇 번에 찾을 수 있겠는가?**

  - 9 개의 동전을 3 개씩 3 등분하여 비교하면, 각 그룹을 비교하는 과정에서 한 번의 비교로 3 개의 그룹 중 가짜 동전이 포함된 그룹을 찾을 수 있다.
  - 이후 다시 그 그룹에서 3 개를 비교하여 가짜 동전을 찾는다.
  - **2 번**에 가짜 동전을 찾을 수 있다.

- **90 개의 동전에서 1 개의 가짜 동전을 찾는 방법을 교재의 방법(광수의 아이디어)으로 찾을 때 몇 번에 찾을 수 있는가?**

  - 교재의 방법에서는 동전을 3 등분하여 비교하는 방식을 사용한다. 90 개의 동전에서 가짜 동전을 찾는다면, 90 개를 3 개씩 나누어 첫 번째 비교를 하고, 그 후 가짜 동전이 있는 그룹을 다시 3 등분하여 비교를 반복한다.
  - 최악의 경우 **4 번**의 비교로 가짜 동전을 찾을 수 있습니다. (90 → 30 → 10 → 3 → 1)

- **90 개의 동전을 3 등분하는 방법으로 찾으면 몇 번에 찾을 수 있는가?**

  - 90 개의 동전도 마찬가지로 3 등분하여 비교하는 방법으로 찾을 수 있다. 각 단계에서 3 등분하여 비교하므로, 위와 동일하게 **4 번**의 비교로 가짜 동전을 찾을 수 있다.

Fakecoin.c

```c
int *find_bad_coin3(int *coins, int n) {
    int i, one3rd = (n + 2) / 3, sum1 = 0, sum2 = 0, sum3 = 0;
    static int count = 0;

    if (n <= 3) {
        count++;
        printf("Found it! with %d iteration\n", count);
        if (coins[0] < coins[1]) return coins;
        else if (coins[1] < coins[2]) return coins + 1;
        else return coins + 2;
    }

    int *coins2 = coins + one3rd;
    int *coins3 = coins + 2 * one3rd;

    for (i = 0; i < one3rd; i++) {
        sum1 += coins[i];
        sum2 += coins2[i];
        sum3 += coins3[i];
    }

    count++;

    if (sum1 < sum2)
        return find_bad_coin3(coins, one3rd);
    else if (sum2 < sum3)
        return find_bad_coin3(coins2, one3rd);
    else
        return find_bad_coin3(coins3, one3rd);
}

```

문제 출력 디버그 콘솔 터미널 포트

```
202020827@cslinux2:~/com/2$ ./fakecoin
Usage: ./fakecoin <number of coins>
202020827@cslinux2:~/com/2$ ./fakecoin 10
Found it! with 3 iteration
coins[0] = 5, found bad coin at index: 0, coins[0] = 5
202020827@cslinux2:~/com/2$ ./fakecoin 5
Found it! with 2 iteration
coins[1] = 5, found bad coin at index: 1, coins[1] = 5
202020827@cslinux2:~/com/2$ ./fakecoin 3
Found it! with 1 iteration
coins[2] = 5, found bad coin at index: 2, coins[2] = 5
202020827@cslinux2:~/com/2$
```