학습목표

- 탐색의 개념을 이해하고 설명할 수 있다.
- 순차와 이진 탐색 알고리즘을 이해하고 프로그램을 작성할 수 있다.

EBS tip

1 탐색의 개요

(1) 탐색의 개념

① 탐색(search)은 많은 자료들 중에서 원하는 자료를 찾는 방법을 말하는 것으로 검색이라고도 한다.

② 탐색은 우리들이 처리하는 작업 중에서 가장 시간이 많이 소요되는 작업 중의 하나이다. 그러한 이유로 많은 자료들을 탐색하기 위하여 컴퓨터를 사용하며, 원하는 자료를 찾을 때 탐색 알고리즘을 사용한다.

③ 탐색은 수많은 자료 중에서 원하는 자료를 얼마나 빨리 찾느냐에 따라 프로그램의 성능에 많은 영향을 주므로 프로그래밍 기법에 중요한 사항 중의 하나이다.

④ 탐색은 대부분 자료를 삽입하거나 삭제하는 과정에서 정렬과 함께 많이 사용한다.

◉ 실생활 속에서 탐색의 예

- 전화번호부에서 친구의 이름을 찾는 일
- 교과서에서 배운 내용을 색인표를 이용하여 찾는 일
- 사전에서 영어 단어를 찾는 일

(2) 탐색의 종류

① 많이 사용하는 탐색 방식에는 순차 탐색과 제어 탐색이 있다.

② 순차 탐색은 가장 간단한 탐색 기법으로 첫 번째 자료부터 차례로 원하는 자료를 찾는 방법이다.

③ 제어 탐색은 자료가 순서적으로 정렬되어 있는 경우에 사용할 수 있는 탐색 방법으로 이진 탐색, 피보나치 탐색, 보간 탐색 등이 있다.

④ 이진 탐색은 전체의 자료를 두 개로 분할하여 탐색해 나가는 방법이다.

⑤ 피보나치 탐색은 자료를 피보나치수열 순으로 탐색해 나가는 방법이다.

⑥ 보간 탐색은 자료를 사전이나 전화번호부에서 이용하는 것과 같이 있음직한 위치를 선택하여 탐색해 나가는 방법이다.

피보나치 수열은 임의의 항이 앞의 두 항의 합과 같은 수열이다.

2 순차 탐색

(1) 순차 탐색의 개요

① 순차 탐색은 자료들이 순서에 관계없이 무작위로 저장되어 있을 때 사용할 수 있는 가장 단순한 탐색 방법으로서, 탐색할 대상의 자료들을 처음부터 하나씩 차례로 비교하면서 찾아가는 탐색 방법이다.

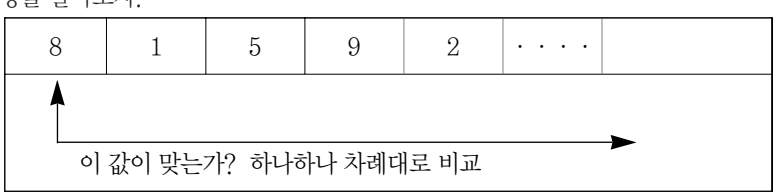
② 순차 탐색은 탐색 알고리즘 중에서 가장 간단하고 이해하기 쉬운 알고리즘이다.

③ 탐색 방법이 쉽고 간단하며, 찾으려는 대상 자료들이 정렬되지 않아도 되는 장점이 있지만 원하는 자료를 찾을 때까지 하나씩 모두 비교해야 하므로 검색 속도가 느린 비효율적인 탐색 방법이다.

순차 탐색을 선형 탐색이라고도 한다.

(2) 순차 탐색 알고리즘

찾으려는 카드가 9일 경우, 주어진 카드들로부터 9가 적힌 카드를 순서대로 찾는 과정을 알아보자.

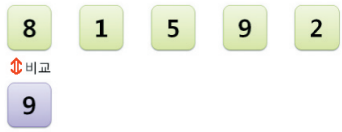


▲ 순차 탐색 과정

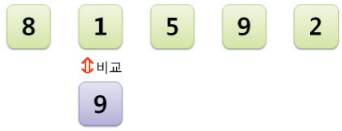
순차 탐색 알고리즘은 자료들이 순서대로 정렬되어 있지 않을 경우에 유용한 탐색 방법이다.

EBS tip

① 첫 번째 카드 값인 8과 찾으려는 카드 값 9를 비교하는데, 두 카드 값이 서로 다르기 때문에 다음 두 번째 카드로 이동한다.



② 두 번째 카드 값인 1과 찾으려는 카드 값 9를 비교하는데, 두 카드 값이 서로 다르기 때문에 다음 세 번째 카드로 이동한다.



③ 세 번째 카드 값인 5과 찾으려는 카드 값 9를 비교하는데, 두 카드 값이 서로 다르기 때문에 다음 네 번째 카드로 이동한다.

④ 네 번째 카드 값인 9와 찾으려는 카드 값 9를 비교하는데, 두 카드 값이 서로 같으므로 원하는 카드를 찾고 찾는 것을 종료한다.

⑤ 만약 마지막까지 원하는 카드를 찾지 못하면 찾는 것은 실패하고 종료한다.

예 순차 탐색 프로그램

프로그램 소스 코드	실행 결과

#include <stdio.h>
void main( ) {
  int a, num;
  int data[8]={2, 9, 30, 100, 5, 4, 32, 45};
  a=0;
  scanf("%d", &num);
  while(a<=7 && data[a]!=num)
  a++;
  if (a<=7)
  printf("%d번째에 있음\n", a+1);
  else
  printf("찾는 수는 없음");
}
	5를 입력한 경우
	
	5번째에 있음.

[프로그램 해설]

ⓐ scanf("%d", &num); : 찾을 자료를 입력한다.

ⓑ while(a<=7 && data[a]!=num) : 탐색할 조건을 보면 "a<=7"에서 a의 초깃값이 0이므로 0부터 7까지 즉, 배열에 저장된 자료만 선택한다. 또한 "data[a]!=num"에서 찾는 값과 배열 요소 값이 같지 않은 경우만 반복문을 수행한다.

ⓒ if (a<=7) : 반복문을 빠져 나왔을 때 a<=7이면 배열에 찾고자하는 값이 있는 것이다.

ⓓ else : 이 경우에는 a=8이므로 반복 조건에서 "a<=7"을 만족하지 않아 빠져 나온 경우이므로 찾는 값이 배열 내에 없는 경우이다.

3 이진 탐색

(1) 이진 탐색의 개요

① 정렬되어 있는 자료 중에서 원하는 자료를 찾는 방법이다.

② 정렬된 자료들을 두 부분으로 나누어 찾고자 하는 자료가 어느 부분에 속하는가를 결정하여 해당 부분에 대하여 검색을 수행한다.

③ 알고리즘은 자료들이 이미 정렬되어 있어야만 유용하게 쓰이는 알고리즘으로, 선형 탐색보다 효율적인 방식이다.

(2) 이진 탐색 알고리즘

① 이진 탐색은 원하는 자료를 처음부터 비교하는 것이 아니라, 가운데 있는 자료와 비교하여 같으면 탐색에 성공하고, 같지 않으면 탐색을 계속 수행한다.

② 가운데 자료가 원하는 값보다 크면 가운데 자료의 왼쪽 부분을 대상으로 탐색을 계속하고, 가운데 자료가 원하는 값보다 작으면 가운데 자료의 오른쪽 부분을 대상으로 탐색을 계속한다.

③ 자료의 집합이 오름차순으로 배열되어 있을 때 이진 탐색의 알고리즘은 다음과 같다.

㉠ 중앙값을 구한다. 이때 중앙값을 가르키는 첨자[(배열의 맨 왼쪽 위치 + 배열의 맨 오른쪽 위치)/2]는 정수 값을 취한다.
㉡ 중앙값과 찾는 값이 같으면 찾는 값의 위치를 출력하고 끝낸다.
㉢ 중앙값이 찾는 값이 아니라면 찾는 값과 중앙값의 크기를 비교한다.
㉣ 찾는 값이 중앙값보다 작으면 배열의 맨 오른쪽 첨자를 [중앙값의 첨자−1]로 한다.
㉤ 찾는 값이 중앙값보다 크면 배열의 맨 왼쪽 첨자를 [중앙값의 첨자+1]로 한다.
㉥ 오른쪽 위치의 첨자에서 왼쪽 위치의 첨자를 뺀 값이 0 이상이면 ㉠부터 ㉤의 과정을 되풀이한다.
㉦ 오른쪽 위치의 첨자에서 왼쪽 위치의 첨자를 뺀 값이 0 보다 작으면 찾는 값이 없는 것이므로 찾는 값이 없음을 출력하고 끝낸다.

EBS tip

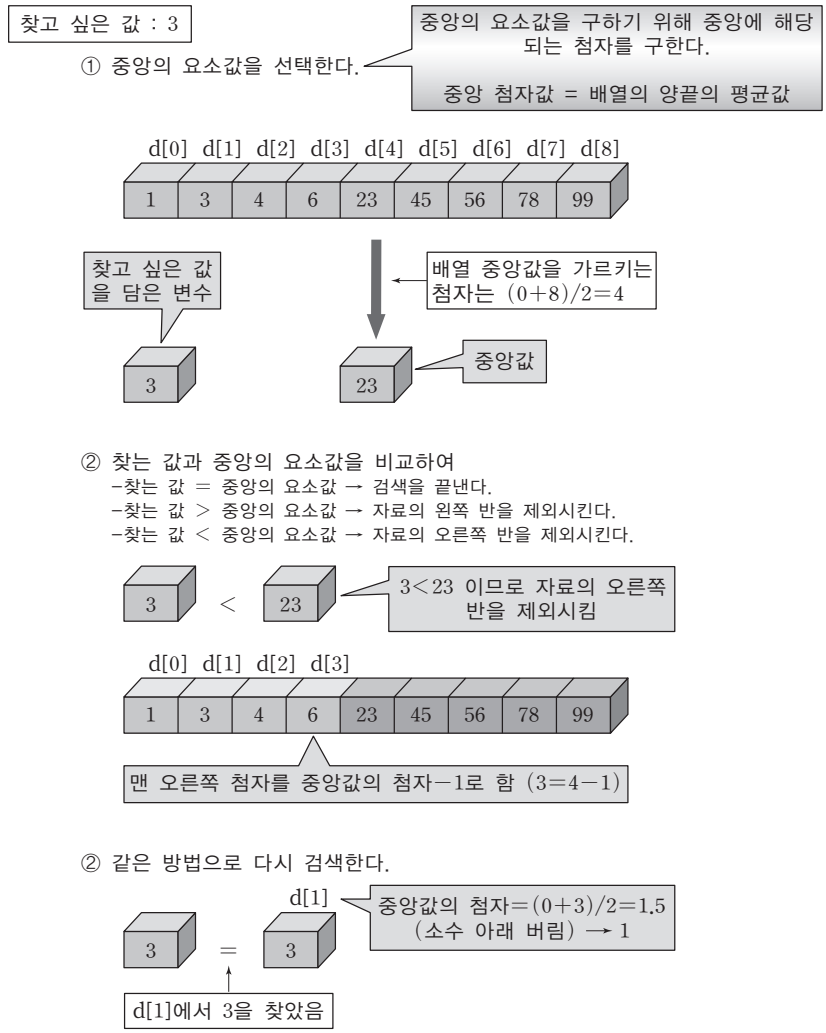
■ while의 조건문에 배열의 첨자가 초과되지 않았는가와 찾는 자료와 일치하는가의 두 가지 조건을 제시했는데 이것 대신 탐색하고자 하는 자료를 배열의 맨 뒤에 삽입시켜 배열의 첨자가 상한을 넘어 탐색하지 않도록 하는 방법도 있다.

◉ 중간 위치 찾기

(찾을 범위의 첫 번째 위치 + 찾을 범위의 마지막 위치) / 2 로 구한다.
전체 값의 개수가 10개이면 중간 위치는 (0 + 9) / 2 = 4가 되며, 전체 값의 개수가 11개라면 중간 위치는 (0 + 10) / 2 = 5가 된다.

EBS tip

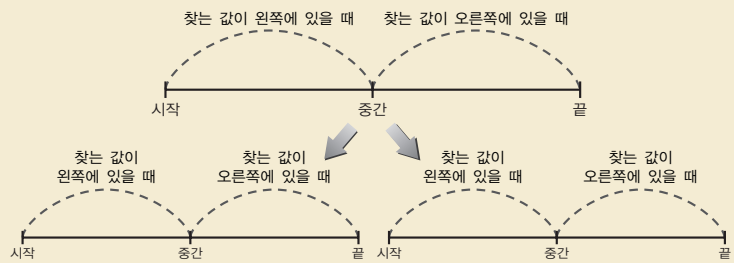
④ 이진 탐색 과정을 그림으로 설명하면 다음과 같다.



▲ 이진 탐색 과정

참고

이진 탐색은 자료가 오름차순이나 내림차순 어떤 방식으로 정렬되어 있어도 된다.



예 이진 탐색 프로그램

프로그램 소스 코드	실행 결과

#include <stdio.h>
void main( ) {
int a, left, right, mid, num;
int d[9]={1, 3, 4, 6, 23, 45, 56, 78, 99};
left=a=0;
right=8;
scanf("%d", &num);
while(1){
if (right-left>=0){
a++;
mid=(left+right)/2;
if (num==d[mid]){
printf("%d번째에 있음\n", mid+1);
printf("%d번 수행했음\n", a);
break;
}
if (num>d[mid])
left=mid+1;
else
right=mid-1;
}
else
{ printf("찾는 수는 없음");
break;
}
}
}
	19를 입력한 경우
	찾는 수는 없음
	
	56을 입력한 경우
	7번째에 있음
	2번 수행했음

[프로그램 해설]

ⓐ while(1) : 조건이 1이므로 항상 참이다. 따라서 반복문은 무한히 반복되는데 반복 루프 안에 이 반복문을 빠져 나갈 수 있는 break 문이 있다.

ⓑ if (right-left>=0) : 오른쪽 위치의 첨자에서 왼쪽 위치의 첨자를 뺀 값이 0 이상이어야 자료를 탐색할 수 있다. 만일 이 조건이 만족하지 않으면 찾는 값이 주어진 배열에는 없는 것이다.

ⓒ a++ : 탐색의 수행 횟수를 카운트한다.

ⓓ mid=(left+right)/2 : 중앙값을 찾기 위해 가운데 첨자를 구한다.

ⓔ if (num==d[mid]) : 찾는 값과 중앙값을 비교하여 같으면 찾는 값의 위치를 출력하고 반복문을 빠져 나간다.

ⓕ if (num>d[mid]) left=mid+1 : 찾는 값이 중앙값보다 크면 배열의 맨 왼쪽 첨자를 [중앙값의 첨자+1]로 수정한다.

ⓖ else right=mid-1 : 찾는 값이 중앙값보다 작으면 배열의 맨 오른쪽 첨자를 [중앙값의 첨자-1]로 수정한다.

EBS tip

◉ 정렬 과정

- 정렬된 100개의 값을 이진 탐색하면 다음과 같이 찾을 범위가 절반씩 줄게 된다.
  1번째 찾아보기 범위 : 100개
  2번째 찾아보기 범위 : 50개
  3번째 찾아보기 범위 : 25개
  4번째 찾아보기 범위 : 12개
  5번째 찾아보기 범위 : 6개
  6번째 찾아보기 범위 : 3개
  7번째 찾아보기 범위 : 1개
  따라서 최대 7번을 비교하면 원하는 값을 찾을 수 있다.

- 범위가 절반씩 줄게 된다는 의미를 거꾸로 생각하면 찾을 범위가 2배로 늘어날 때마다 찾아보는 횟수가 1씩 증가하게 된다. 즉 7번을 찾을 수 있다면 찾을 개수가 128개($=2^7$) 안에서 원하는 값을 찾을 수 있게 된다.

기출 모의고사

정답 및 해설 p. 17

2005학년도 대수능

1 다음은 정렬된 배열에서 숫자 n을 탐색하는 알고리즘이다. 배열 A에서 35를 탐색하기 위해 수행해야 할 알고리즘의 반복 횟수로 알맞은 것은?

(1) 숫자 n이 탐색할 배열 범위의 가운데 요솟값과 같을 경우, 탐색 과정을 종료한다.
(2) 숫자 n이 탐색할 배열 범위의 가운데 요솟값보다 클 경우, 첫 번째 요소부터 가운데 요소까지를 탐색 대상에서 제외한다.
(3) 숫자 n이 탐색할 배열 범위의 가운데 요솟값보다 작을 경우, 가운데 요소부터 끝 요소까지를 탐색 대상에서 제외한다.
(4) (1)로 되돌아 간다.

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
배열 A	11	15	17	21	29	35	41

① 2 ② 3 ③ 5
④ 6 ⑤ 7

2 다음 프로그램을 실행한 후 15를 입력했을 때 출력되는 값은?

```
#include <stdio.h>
void main( ) {
  int a, num;
  int data[8]={22, 92, 33, 123, 15, 14, 35, 47};
  a=0;
  scanf("%d", &num);
  while(data[a] != num && a<=7)
    a++;
  if (a<=7)
      printf("%d\n", a+1);
  else
      printf("%s\n", "찾을 수 없음");
}
```

① 3 ② 4 ③ 5
④ 6 ⑤ 7

2008학년도 대수능

3 다음은 검색 프로그램이다. 이 프로그램에 대한 설명으로 옳은 것을 〈보기〉에서 고른 것은?

```
#include <stdio.h>
int a[ ]={0, 40, 20, 60, 10, 30, 50, 70};
int s=8;

int search(int x) {
  int b=1;
  while(b < s) {
    if (a[b] == x)
      return b ;
    else if (a[b] > x)
      b = b * 2;
    else
      b = b * 2 + 1 ;
  }
  return – 1 ;
}
```

보기

ㄱ. 검색에 실패하면 음수를 반환한다.
ㄴ. 변수 b는 배열에 저장된 값을 의미한다.
ㄷ. 배열 a는 검색 전에 내림차순으로 정렬되어 있어야 한다.
ㄹ. 검색에 성공하면 찾은 값이 존재하는 배열의 첨자를 반환한다.

① ㄱ, ㄴ ② ㄱ, ㄷ ③ ㄱ, ㄹ
④ ㄴ, ㄷ ⑤ ㄴ, ㄹ

4 다음은 탐색 프로그램의 일부이다. 7을 찾기 위해 비교하는 횟수는?

```
#include <stdio.h>
void main( ) {
 int n, z, x=0, y=9;
 int dat[10]={3,7,10,15,24,36,41,55,71,89};
 scanf("%d", &n);
 do{
   z=(x+y)/2;
   if (dat[z]==n) {
      printf("%d 찾음", n);
      break;
 }
 if (dat[z]>n)
   y=z-1;
 else
   x=z+1;
 } while(x<=y);
}
```

① 1 ② 2 ③ 3
④ 4 ⑤ 5

5 다음은 정렬된 자료에서 원하는 자료를 찾는 프로그램의 일부이다. 68을 탐색하기 위해서는 while 문을 몇 번 수행해야 하는가?

```
#include <stdio.h>
void main( ) {
  int a, left, right, mid, num;
  int data[11]={11, 16, 21, 23, 45, 68, 78, 99};
  left=0;
  right=7;
  a=0;
  scanf("%d", &num);
  while(1)
  {
   if (right-left>=0)
   {
       a++;
       mid=(left+right)/2;
       if (num==data[mid])
       {
          printf("위치 : %d \n", mid+1);
          printf("%d번 수행했음 \n", a);
          break;
       }
       if (num > data[mid])
         left=mid+1;
       else
         right=mid-1;
   }
       else
            {
              printf("찾는 수는 없음");
              break;
            }
  }
}
```

① 1 ② 2 ③ 3
④ 4 ⑤ 5