# 서울시립대학교 소모임 알

컴퓨터과학부 김희중

## Search 2

# Binary Search

# Binary Search

이진 탐색



- 이진 탐색이라는 이름이 붙은 이유는 한번 비교할 때 마다 탐색 범위가 반으로 줄어들기 때문이다.
- 배열에서의 이진 탐색이 가장 흔하지만 배열에서만 쓰이는 탐색법이 아니다.
- 이분 탐색, 이분법은 모두 같은 말이다.

# Linear Search

선형 탐색



- 배열에서 특정 원소를 찾아야 하는 상황이다.
- 아래와 같이 7칸의 배열에 숫자가 들어있고 '8'을 찾아야 한다.

| 2 | 5 | 3 | 9 | 10 | 8 | 7 |
|---|---|---|---|---|---|---|

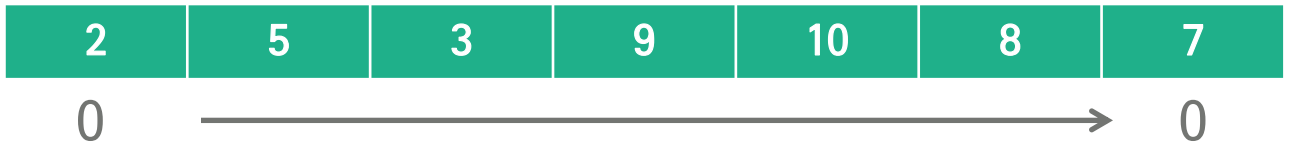
# Linear Search

선형 탐색



- 0번 index부터 마지막 6번 index까지 for문을 이용하여 탐색한다.

```
for(int i=0; i<7; i++){
        if(A[i] == 8){
            //탐색 성공
        }
    }
```

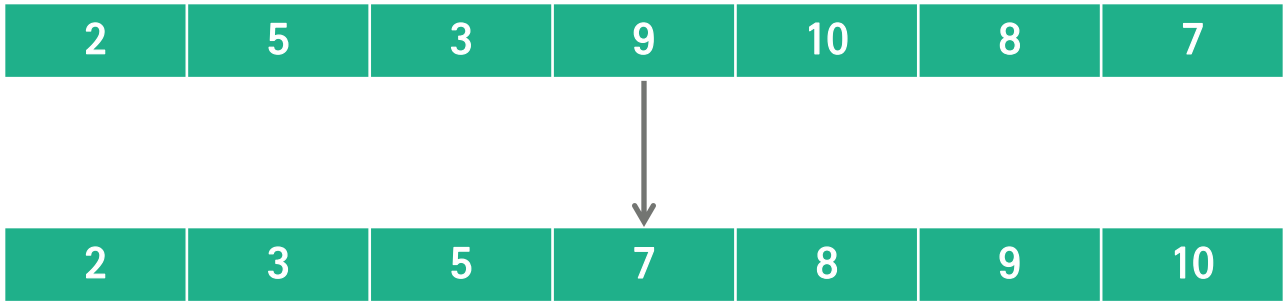


# Binary Search

이진 탐색



- 배열에서 8을 이진 탐색으로 찾아야 하는 상황이다.
- 정렬이 되어있어야 한다.



# Binary Search



이진 탐색

- Low = 0, high = 6;
- Mid = (0 + 6)/2 = 3;
- S[3]은 7이기 때문에 8보다 작다, 따라서 low를 mid+1인 4로 갱신한다.

# Binary Search



이진 탐색

- Low = 4, high = 6;
- Mid = (4 + 6)/2 = 5;
- S[5]은 9이기 때문에 8보다 크다, 따라서 high를 mid-1인 4로 갱신한다.

# Binary Search

이진 탐색



- Low = 4, high = 4;
- Mid = (4 + 4)/2 = 4;
- S[4]은 8로 원하는 값을 찾았으니 탐색을 종료한다.

# Binary Search

이진 탐색



- Binary Search는 꼭 알아 둘 것, 후에 Parametric Search로도 연결된다.

```
int low=0, high=6, mid;
while(low <= high){
        mid = (low + high)/2;

        if(A[mid] == 8){
            //탐색 성공
            break;
        }
        else if(A[mid] < 8){
            low = mid+1;
        }
        else{
            high = mid-1;
        }
}
```

# Binary Search



정렬은 어떻게…?

- 자꾸 정렬하라고 하는데… 정렬을 할 줄 모른다.
- 정렬도 굉장히 중요한 파트이지만 다음에 다루도록 하고 O(nlogn) 복잡도의 정렬 라이브러리 사용법을 알아보도록 하자.

# Binary Search



배열의 정렬

- #include <algorithm> 헤더를 선언한다.
- using namespace std; 로 namespace 사용을 선언한다.
- sort(배열이름 + 정렬 시작 index, 배열 이름 + 정렬 마지막 인덱스 +1); 하면 오름차순 정렬 완성!

# Binary Search



배열의 정렬

- 코드를 보는편이 이해가 훨씬 빠르다.
- 배열 A의 0번째 원소부터 99번째(N-1)번째 원소까지 내림차순으로 정렬한다. (사실상 처음부터 끝 까지)

```
int A[100];

int N = 100;
sort(A, A+N);
```

# Binary Search



배열의 정렬

- 전체가 아닌 부분만 정렬할 때는?
- 배열 A의 10번 인덱스부터 21번 인덱스까지 오름차순으로 정렬한다.

```
int A[100];

int N = 100;
sort(A+10, A+22);
```

# Binary Search



벡터의 정렬

- 배열과 비슷한 **벡터의 정렬**도 거의 똑같다.
- #include <algorithm> 헤더를 선언한다.
- using namespace std; 로 namespace 사용을 선언한다.
- Sort(벡터의 정렬 시작주소, 벡터의 정렬 마지막 **바로 뒤 주소**);

# Binary Search



벡터의 정렬

- 벡터의 전체를 정렬할 때는 벡터의 begin(), end()메서드를 이용한다.
- 벡터의 맨 처음 원소부터 마지막 원소까지 정렬한다.

```
vector<int> A;
A.push_back(…);
…

sort(A.begin(), A.end());
```

# Binary Search



벡터의 정렬

- 마찬가지 부분만 정렬할 수 있다.
- 벡터의 5번 index부터 11번(12-1)번 index까지 오름차순으로 정렬한다.

```
vector<int> A;
A.push_back(…);
…

sort(A.begin()+5, A.begin() + 12);
```

# Binary Search



벡터의 정렬

- begin()메서드와 end()메서드를 이해하고 활용해야 한다.
- 벡터의 5번 index부터 뒤에서 3번째 index까지 오름차순으로 정렬한다.

```
vector<int> A;
A.push_back(…);
…

sort(A.begin()+5, A.end() - 3);
```

# 수 찾기

# 1920



https://www.acmicpc.net/problem/1920

- 앞서 설명한 정렬과 Binary Search를 이용하는 문제이다.

# 날짜 계산

# 1476

https://www.acmicpc.net/problem/1476



- 완전 탐색의 기본 = 다 해본다.
- 가능한 모든 경우의 수를 알아야 한다.
- 가능한 모든 경우의 수를 탐색하는 방법.
- = 1년으로 시작해서 맞아떨어지는 때가 찾아질 때 까지 년도를 늘려나간다.(무한 반복)

# 1, 2, 3 더하기

# 9095



https://www.acmicpc.net/problem/9095

- 완전탐색은 재귀함수로 가장 많이 구현한다.
- DFS, 백트래킹과 관련이 있다.

# 부분집합의 합

# 1182



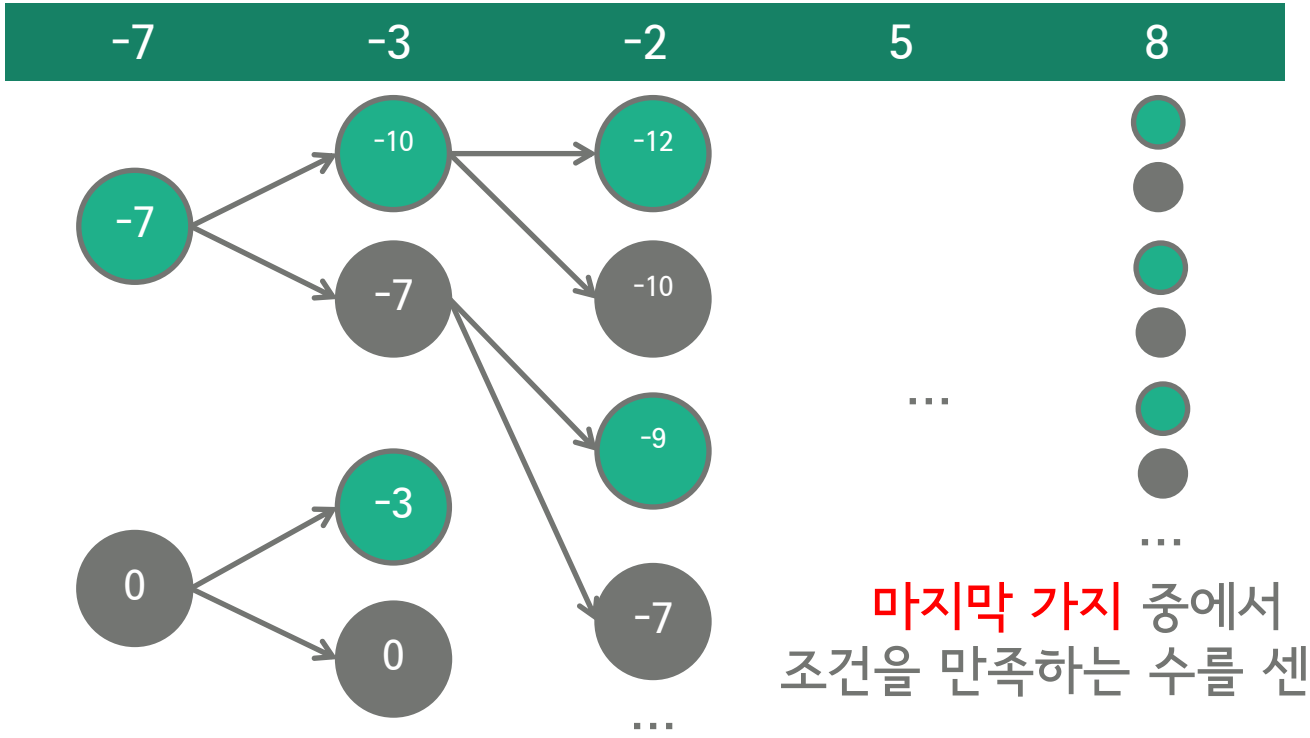
https://www.acmicpc.net/problem/1182

- 지난 시간 풀었던 부분집합의 합을 조금 다른 방식으로 해결해보자.
- 대부분의 완전 탐색 문제를 해결할 때에는 가지 만들기를 할 수 있어야한다.
- 이 문제는 단계마다 해당 원소를 (선택한다, 선택하지 않는다) 이 두 가지의 가지를 만들어낸다.

# 1182

https://www.acmicpc.net/problem/1182





# 1182

https://www.acmicpc.net/problem/1182



- 가지 만들기는 **재귀함수**로 그대로 옮겨낼 수 있다.

```
//idx-1번째 선택을 완료했고 원소의 합이 sum일 때 합이 S인 경우의 수
int solve(int idx, int sum){
    if(idx == N){
        if(sum == S){
            return 1;
        }
        return 0;
    }

    int ret = solve(idx+1, sum+s[idx]) + solve(idx+1, sum);

    return ret;
}
```

바로 여기 !

# 로또

# 6603



https://www.acmicpc.net/problem/6603

- 완전탐색 문제는 N중 for문을 이용해서 해결할 수도 있다.
- 문제에서 N을 특정 상수로 정해준다면 재귀함수가 아닌 방법으로 해결할 수도 있다.
- 후보가 되는 숫자는 정해지지 않았지만 골라야 하는 숫자는 6개로 고정되어 있기 때문에 6중 for문으로 해결할 수 있다.

# 6603



https://www.acmicpc.net/problem/6603

```
int a, b, c, d, e, f;
        for(a=0; a<K; a++){
            for(b=a+1; b<K; b++){
                for(c=b+1; c<K; c++){
                    for(d=c+1; d<K; d++){
                        for(e=d+1; e<K; e++){
                            for(f=e+1; f<K; f++){
                                printf("%d %d %d %d %d %d\n", s[a], s[b], s[c], s[d], s[e], s[f]);
                            }
                        }
                    }
                }
            }
        }
```

# 6603



https://www.acmicpc.net/problem/6603

- 하지만 뽑아야 하는 숫자의 개수가 6과 같은 상수가 아니라 미지수라면?
- 재귀함수를 이용해 구현해야 한다!
- 앞의 6중 for문을 재귀함수로 구현해보도록 한다.