

#2주차 알고리즘반

Binary Search, Parametric Search & LIS

T. 여충관

Binary Search

Binary Search에 대해서 알아보자!

하나 다른 숫자는 무엇일까요?

666
666
666
666
666
666
666
666
6666666666666666666666666666686666666666666
666
666
666

Q. 만약 프로그램으로 이 작업을 한다면?

- ➡ 하나하나 다 찾아야함;
- ➡ N개 중에서 찾는다면 최대 N번을 다 찾아야함;
- ➡ 시간 복잡도는 $O(N)$
- ➡ 이를 선형탐색(Linear Search)이라 한다.



더 좋은 방법?

Binary Search(이진 탐색)

항상 정렬이 되어 있어야함. 단, **오름차순**으로

EX) N = 10

	Left				Mid		Right			
ARR	1	2	3	4	5	6	7	8	9	10

Left : 찾고자 하는 값이 속한 구간의 왼쪽 끝 (시작은 1)

Right : 찾고자 하는 값이 속한 구간의 오른쪽 끝 (시작은 n)

Mid : 구간의 가운데  $(Left + Right) / 2$

Value : 찾고자 하는 값

* Left, Right, Mid 모두 값이 아닌 인덱스이다!!

Left와 **Right**를 바꾸면서 **Value**를 찾는다!

Binary Search(이진 탐색)




Value가 Arr[mid]보다 크다 → Mid+1 ~ Right로 구간이 바뀜

Value가 Arr[mid]보다 작다 → Left+1 ~ Mid로 구간이 바뀜

Value가 Arr[mid]와 같다 → 찾았으므로 끝낸다.

Binary Search(이진 탐색)

이를 식으로 나타내 보면?

$\text{Value} > \text{Arr}[\text{mid}]$		$\text{Left} = \text{mid} + 1$
$\text{Value} < \text{Arr}[\text{mid}]$		$\text{Right} = \text{mid} + 1$
$\text{Value} == \text{Arr}[\text{mid}]$		return Value

Binary Search(이진 탐색)

Left, Right값 설정하기

Mid값 구하기

Arr[mid]와 Value를 비교해 Left혹은 Right값을 변경

반복!

Q. Value가 없는 경우 무한 반복되나요?

A. Value를 찾지 못할 경우에는 $left > right$ 가 됩니다.

즉, $left \leq right$ 동안 반복해주면 해결!

연습문제

BOJ 1920: 수 찾기

구현

```
int main()
{
    int n, m, value;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
        scanf("%d", &arr[i]);

    sort(arr + 1, arr + n + 1); // 정렬

    scanf("%d", &m);
    for (int i = 1; i <= m; i++) {
        scanf("%d", &value);
        printf("%d\n", Binary_Search(1, n, value));
    }
    return 0;
}
```

main

```
int Binary_Search(int left, int right, int value) {
    int mid = (left + right) / 2;
    if (left > right) return 0; // 값이 존재하지 않음
    if (value > arr[mid])
        return Binary_Search(mid + 1, right, value);
    else if (value < arr[mid])
        return Binary_Search(left, mid - 1, value);
    else
        return 1; // 값이 존재함
}
```

재귀 함수로 구현

```
int Binary_Search(int left, int right, int value) {
    int mid;
    while (left <= right) {
        mid = (left + right) / 2;
        if (value > arr[mid]) left = mid + 1;
        else if (value < arr[mid]) right = mid - 1;
        else return 1; // 값이 존재함
    }
    return 0; // 값이 존재하지 않음
}
```

반복문으로 구현

다음중에
편한것만
사용하세요
!!!!

과제 (어렵지 않아요~)

BOJ 10816: 숫자카드2

BOJ 3020: 개똥벌레

BOJ 2792: 보석상자

Parametric Search

Parametric Search에 대해서 알아보자!

Parametric Search

- ➡ 답이 가능한 범위를 둔다.
- ➡ 이진 탐색을 통해 범위를 좁히며 답을 결정함.

P.S 이 알고리즘은 문제로 이해하는게 더 수월함!

연습문제

BOJ 1654: 랜선 자르기

풀이 - 브루트 포스

1. 랜선의 길이(L)를 1씩 증가 시킨다.
2. 가지고 있는 조각을 L로 나눈 몫의 합을 구함.
3. N개의 조각이 만들 수 없는 경우 발생
4. 그 때의 L-1이 답이 됨.

- ➡ 시간 복잡도 $O(K*L)$ 인데 L의 최대값이 $2^{31}-1$ 이다.
- ➡ 따라서 TL(시간초과)가 나게 된다.

Q. L길이의 조각을 N개 이상 만들 수 있는가?

Left	Mid (200)	Right ($2^{31}-1$)
ARR	YES	NO

➡ 이진 탐색으로 L의 범위를 좁혀 가면서 YES에 해당하는 L의 최대값을 찾자!

Parametric Search

1. 이진탐색을 통해 L의 값을 결정
2. L의 길이로 최대 몇 조각을 만들 수 있는지 계산
3. K개 이상의 조각을 만들 수 있다면

➡ L값을 갱신해주고 Left를 $\text{Mid} + 1$ 로 바꿔준다!

Why? L보다 큰 범위에서 답을 찾을 가능성이 있기 때문!

4. K개의 조각을 만들 수 없다면

➡ Right를 $\text{Mid} - 1$ 로 바꿔준다!

Why? L보다 작은 범위에서 답을 찾을 가능성이 있기 때문!

구현

```
int main()
{
    scanf("%d%d", &n, &k);
    for (int i = 1; i <= n; i++)
        scanf("%d", &len[i]);

    printf("%lld", PS(1, inf));
    return 0;
}
```

main

```
#define inf 2147483647
#include <bits/stdc++.h>
using namespace std;
int n, k, ans;
int len[10005];

long long PS(long long left, long long right) {
    long long mid, piece;
    while (left <= right) {
        mid = (left + right) / 2; // 2^31-1은 int 최대범위이다.
        piece = 0;
        for (int i = 1; i <= n; i++) piece += (len[i] / mid);
        if (piece >= k) { // k개의 조각을 만들 수 있는 경우
            if (ans < mid) ans = mid;
            left = mid + 1;
        }
        else right = mid - 1; // k개의 조각을 만들 수 없는 경우
    }
    return ans;
}
```

parametric 함수 구현

과제

BOJ 2512: 예산

BOJ 1072: 게임

BOJ 2805: 나무 자르기

BOJ 2613: 숫자구슬

LIS

LIS에 대해서 알아보자!

LIS란?

Longest Increasing Subsequence

Longest 최장

Increasing 증가

Subsequence 부분 수열

LIS

우리는 이미 $O(N^2)$ 풀이에 대해 배웠다....!

하지만..

연습문제

BOJ 10215:
가장 긴 증가하는 부분 수열2

LIS

19084261

te06008

 12015

시간 초과

LIS

첫째 줄에 수열 A의 크기 N ($1 \leq N \leq 1,000,000$)이 주어진다.

→ $O(N^2)$ 풀이로 구현할 경우,
 $(1,000,000)^2 = 1,000,000,000,000$

lower_bound(First, Last, val)

정렬되어 있는 배열에서 구간 `[first, last)`에
val 이상인 수가 처음으로 나타나는 위치를
Binary Search(이진 탐색)로 찾아주는 함수입니다.

표준 헤더 `<algorithm>`에 들어있습니다.

시간복잡도 : $O(\log N)$

lower_bound(First, Last, val)

arr[i]	0	1	2	3	4
Value	5	13	30	54	56

lower_bound(arr, arr+5, 6)

→ 1 반환

lower_bound(arr, arr+5, 100)

→ 해당하는 위치가 없으므로 5 반환

LIS by $O(N \log N)$

1. 전체 문제

- 길이 N 의 수열 안에서 LIS의 길이를 구하는 것

2. 부분 문제

- $k < N$ 을 만족하는 양수 K 에 대해 $arr[1] \sim arr[k]$ 까지 LIS 값.

 부분문제 역시 길이 K 수열의 LIS 길이를 구하는 것이다!

3. DP Table

- $DP[i]$: 길이 i 의 LIS중, **마지막 수가 가장 작은 LIS의 마지막 수**

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}

index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1

index							
DP							

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index							
DP							

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1						
DP	2						

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1						
DP	2						

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2					
DP	2	6					

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2					
DP	2	6					

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3				
DP	2	6	9				

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3				
DP	2	6	9				

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3				
DP	2	4	9				

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3				
DP	2	4	9				

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3				
DP	2	4	5				

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3				
DP	2	4	5				

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3				
DP	2	4	5				

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3	4			
DP	2	4	5	7			

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3	4			
DP	2	4	5	7			

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}



index	1	2	3	4	5	6	7
arr	2	6	9	4	5	7	1



index	1	2	3	4			
DP	1	4	5	7			

설명이 복잡하니 예시로....

arr = {2, 6, 9, 4, 5, 7, 1}

index	1	2	3	4			
DP	1	4	5	7			

DP배열의 길이  LIS의 길이!

But. 수를 덮어쓰면서 DP배열이 만들어지므로
DP배열의 원소가 LIS의 원소는 아님!

구현

```
int arr[1000005];
vector<int> DP;
int main() {
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) scanf("%d", &arr[i]);
    for (int i = 1; i <= n; i++) {
        if (DP.size()==0 || arr[i] > DP.back()) {
            //벡터의 가장 마지막 값보다 크다면(증가하는 순서) push
            DP.push_back(arr[i]);
        }
        else {
            //arr[i]이상의 값이 처음으로 나타나는 위치에 arr[i]를 넣는다.
            int index = lower_bound(DP.begin(), DP.end(), arr[i]) - DP.begin();
            DP[index] = arr[i];
        }
    }
    printf("%d", DP.size());
    return 0;
}
```

LIS 비교 $O(N^2)$ vs $O(N\log N)$

	$O(N^2)$	$O(N\log N)$
구현법	바깥쪽 for문으로 전체를 순회 하면서 $O(N)$, 안쪽으로 현재 원소보다 작은 걸 찾음 $O(N)$	배열 전체를 순회하면서 $O(N)$, lower_bound로 현재 원소보다 크거나 같은 수를 찾음 $O(\log N)$
DP[i] 정의	arr[i]를 마지막으로 하는 LIS의 길이	길이가 i인 LIS중 마지막 수가 가장 작은 LIS의 마지막 수
DP Table의 크기 M	$N == M$	$M == (\text{LIS 길이}) \leq N$
DP Table	DP Table로 LIS와 그 길이를 구할 수 있다.	DP Table로는 LIS의 길이만 구할 수 있다!

과제

BOJ 3745: 오름세

BOJ 2352: 반도체 설계

BOJ 1365: 꼬인 전깃줄

끝