#05-2. 분할 정복

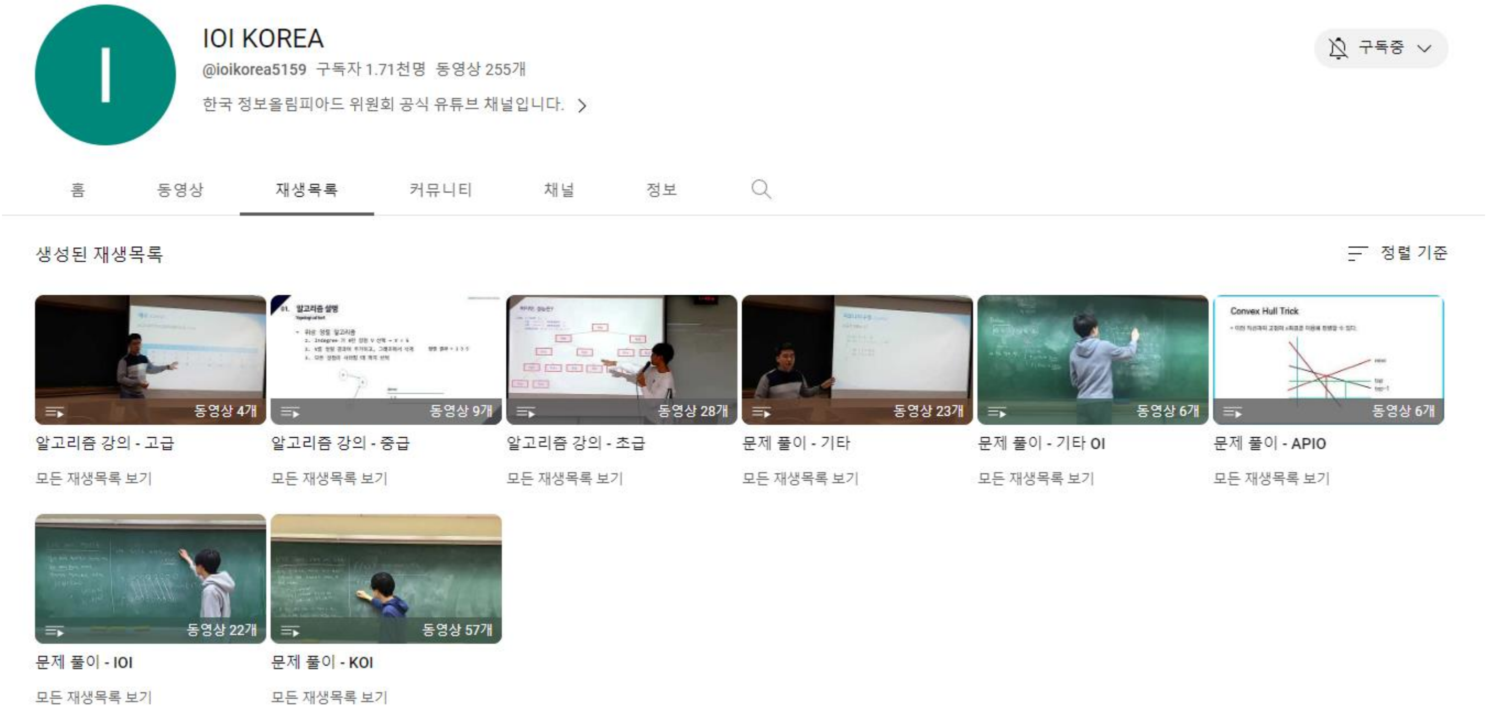
나정휘
https://justiceHui.github.io/

동영상 강의

동일한 내용을 설명하는 영상이 있음

- 분할 정복의 기초 (https://youtu.be/BqqjWGPXNaQ)
- 분할 정복의 응용 (https://youtu.be/MKkIbMPggGY)

- 재귀 함수 말고도 다양한 영상이 있으니 많은 관심 부탁…
- 한국정보올림피아드위원회 공식 유튜브 채널임



분할 정복

분할 정복

- (분할) 큰 문제를 여러 개의 작은 문제로 쪼갠 다음
- (정복) 작은 문제들의 답을 이용해 큰 문제의 답을 구하는 방법

분할 정복의 과정

- 문제의 크기가 충분히 작다면 직접 해결
- 문제의 크기가 크다면 1개 이상의 부분 문제로 분할해서 해결한 뒤
- 부분 문제들의 답을 이용해 전체 문제의 답을 계산

분할 정복

분할 정복의 과정

```
void DivideAndConquer(InputType in, OutputType out){
    // 문제의 크기가 충분히 작은 경우 직접 해결
    if(in.size() <= Small){
        DirectSolve(in, out);
        return;
    }

    // 문제를 K개의 부분 문제로 분할함
    InputType in_small[K] = Divide(in, K);
    OutputType out_small[K];
    for(int i=0; i<K; i++){
        DivideAndConquer(in_small[i], out_small[i]);
    }
    out = Combine(out_small[0], out_small[1], ... , out_small[k-1]);
}
```

분할 정복의 예시 – 1

정수의 거듭 제곱

- 실수 a와 음이 아닌 정수 b에 대해 a^b를 구하는 문제

- b = 0이면 직접 해결 ($a^b=1$)
- b ≥ 1이면 더 작은 문제로 나눠서 해결
  - b가 짝수면 $a^{b/2} \times a^{b/2}$
  - b가 홀수면 $a^{(b-1)/2} \times a^{(b-1)/2} \times a$

- 시간 복잡도
  - $T(0) = O(1)$
  - $T(b) = T(b/2) + O(1)$
  - 따라서 전체 시간 복잡도는 $T(b) = O(\log b)$

분할 정복의 예시 – 1

정수의 거듭 제곱

```
int Pow(int a, int b){
    if(b == 0) return 1;
    int tmp = Pow(a, b / 2);
    if(b % 2 == 0) return tmp * tmp;
    else return tmp * tmp * a;
}
```

분할 정복의 예시 – 2

합병 정렬 알고리즘

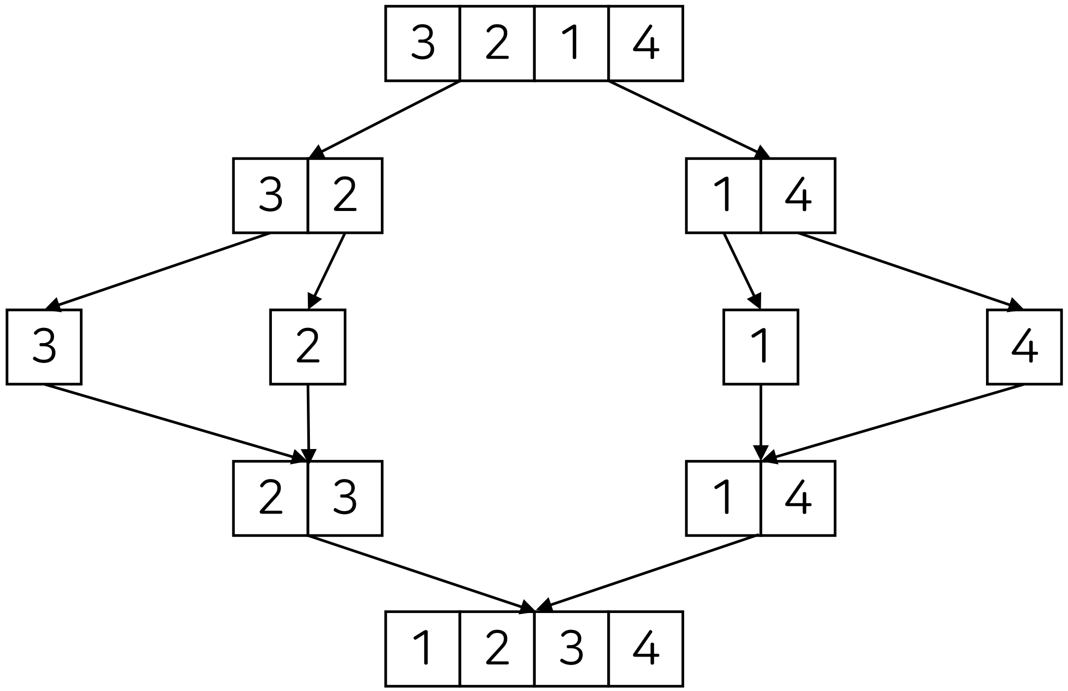
- 배열의 구간 [l, r]을 정렬하는 알고리즘

- l = r이면 직접 해결
  - 원소가 하나인 배열은 이미 정렬된 상태

- l < r이면 더 작은 문제로 나눠서 해결
  - m을 l과 r의 중간 지점이라고 할 때
  - [l, m]과 [m+1, r]을 각각 정렬한 다음 (분할)
  - 정렬된 두 배열을 정렬된 하나의 배열로 합병 (정복)

분할 정복의 예시 - 2

합병 정렬 알고리즘



분할 정복의 예시 – 2

합병 정렬 알고리즘

- 정렬된 두 배열 A, B를 정렬된 하나의 배열 V로 합병하는 방법
  - 두 배열 모두에서 가장 작은 값은 A[0] 또는 B[0]
  - 둘 중 더 작은 값을 V[0]에 저장

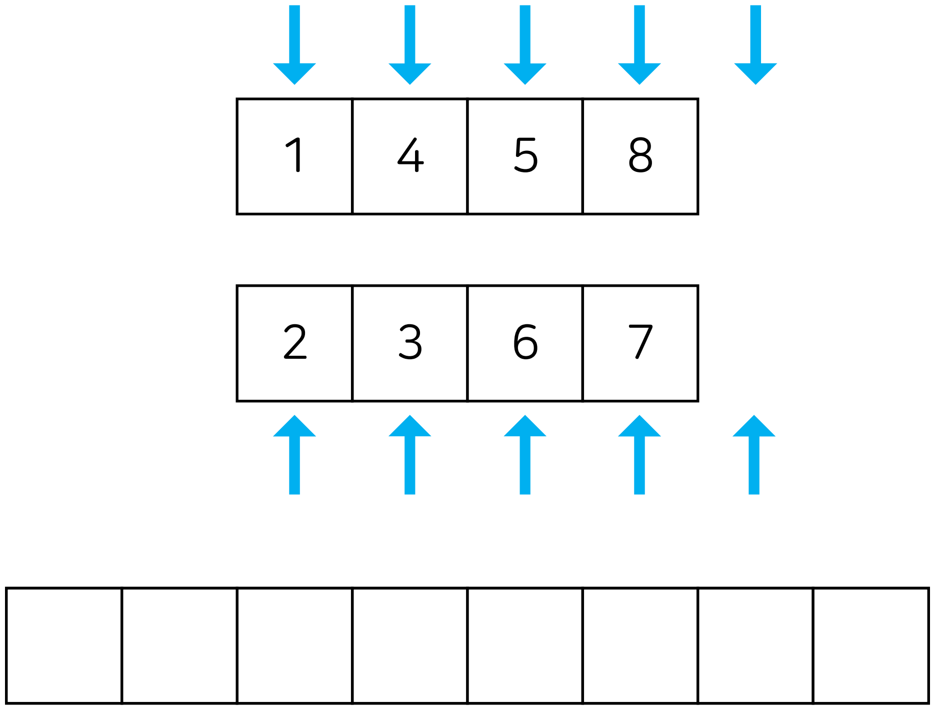
  - A[0] ≤ B[0]이었다면 V[0] = A[0]
  - A[0]을 제외했을 때 가장 작은 값은 A[1] 또는 B[0]
  - 둘 중 더 작은 값을 V[1]에 저장

  - 반복
  - 두 배열에서 각각 가장 작은 값을 가리키는 인덱스를 저장해야 함

분할 정복의 예시 – 2

합병 정렬 알고리즘

- 정렬된 두 배열 A, B를 정렬된 하나의 배열 V로 합병하는 방법



분할 정복의 예시 – 2

합병 정렬 알고리즘

- 정렬된 두 배열 A, B를 정렬된 하나의 배열 V로 합병하는 방법
  - 매번 두 화살표 중 하나는 한 칸 뒤로 이동
  - 두 화살표는 합쳐서 r-l+1번 뒤로 이동

  - 따라서 배열의 길이를 n이라고 하면 시간 복잡도는 O(n)

분할 정복의 예시 - 2

합병 정렬 알고리즘

- 시간 복잡도
  - n을 정렬해야 하는 배열의 길이(= r-l+1)라고 하면
  - $T(1) = O(1)$
  - $T(n) = 2T(n/2) + O(n) = O(n \log n)$
    - 가로 길이 n, 세로 길이 log n인 직사각형의 넓이와 동일

<table>
<tr><td colspan="8">n</td></tr>
<tr><td colspan="4">n/2</td><td colspan="4">n/2</td></tr>
<tr><td colspan="2">n/4</td><td colspan="2">n/4</td><td colspan="2">n/4</td><td colspan="2">n/4</td></tr>
<tr><td>n/8</td><td>n/8</td><td>n/8</td><td>n/8</td><td>n/8</td><td>n/8</td><td>n/8</td><td>n/8</td></tr>
</table>

...

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

분할 정복의 예시 – 2

합병 정렬 알고리즘

```cpp
constexpr int MAX_N = 1'000;

void Merge(int A[], int s, int m, int e){
    static int tmp[MAX_N];
    int i = s, j = m + 1, idx = s;
    while(i <= m && j <= e){
        if(A[i] < A[j]) tmp[idx++] = A[i++];
        else tmp[idx++] = A[j++];
    }
    while(i <= m) tmp[idx++] = A[i++];
    while(j <= e) tmp[idx++] = A[j++];
    for(int k=s; k<=e; k++) A[k] = tmp[k];
}

void MergeSort(int A[], int s, int e){
    if(s == e) return;
    int m = (s + e) / 2;
    MergeSort(A, s, m);
    MergeSort(A, m+1, e);
    Merge(A, s, m, e);
}
```

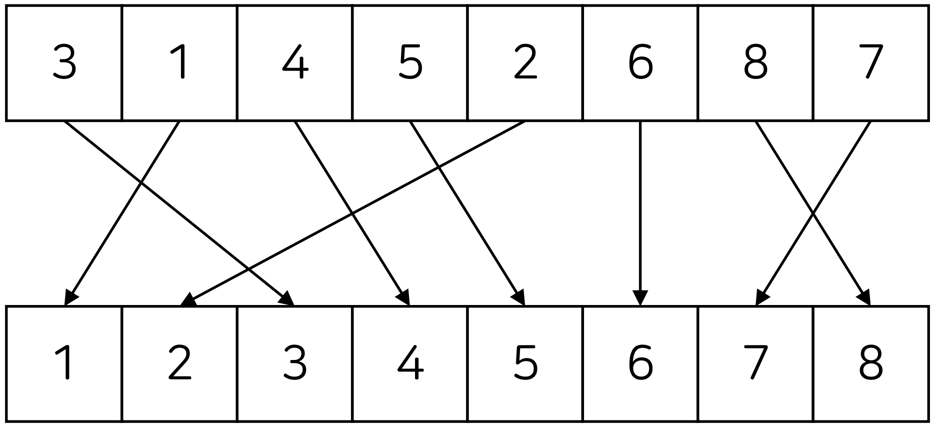
분할 정복의 예시 – 3

반전 수 세기 문제

- i < j & A[i] > A[j]를 만족하는 순서쌍 (i, j)의 개수를 세는 문제

- 각각의 A[i]에 대해, 자신보다 뒤에 있으면서 자신보다 작은 A[j]의 개수를 세는 문제
  - 단순하게 세면 $O(N^2)$
  - 원소들의 실제 위치가 아닌 전후 관계만 중요하다는 것에 주목

- 다음과 같이 그림을 그렸을 때 발생하는 교점의 개수와 동일



분할 정복의 예시 – 3

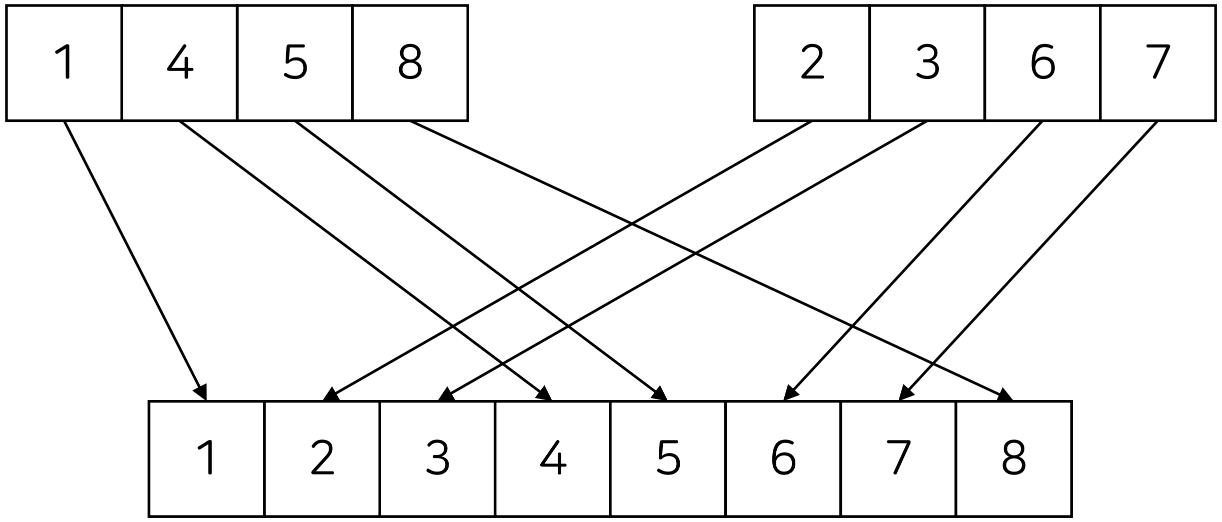
반전 수 세기 문제

- 배열을 절반으로 분할
  - [l, m]과 [m+1, r] 안에서 발생하는 반전 수는 재귀적으로 계산 (분할)
  - [l, m]과 [m+1, r] 간의 반전 수만 세면 된다. (정복)
    - [l, m]의 원소 x보다 작은 [m+1, r]의 원소 y의 개수
    - 합병 정렬과 유사한 방법으로 해결 가능

분할 정복의 예시 – 3

반전 수 세기 문제

- 배열을 절반으로 분할
  - [l, m]과 [m+1, r] 안에서 발생하는 반전 수는 재귀적으로 계산 (분할)
  - [l, m]과 [m+1, r] 간의 반전 수만 세면 된다. (정복)
    - [l, m]의 원소 x보다 작은 [m+1, r]의 원소 y의 개수
    - 합병 정렬과 유사한 방법으로 해결 가능



분할 정복의 예시 – 3

반전 수 세기 문제

- 정복 과정의 구현 방법
  - 지금까지 오른쪽에서 온 화살표의 개수 cnt를 관리
  - 왼쪽에서 화살표가 올 때마다 정답을 cnt 만큼 증가
- 시간 복잡도
  - $T(n) = 2T(n/2) + O(n) = O(n \log n)$

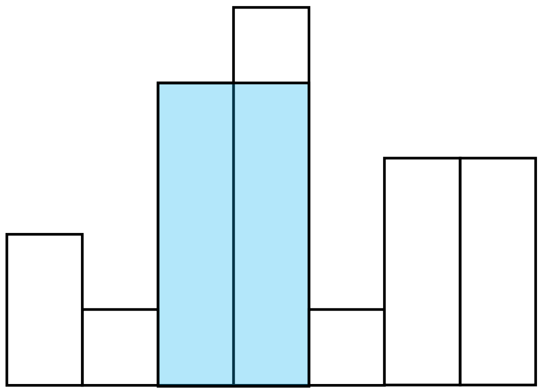
```
int Merge(int A[], int s, int m, int e){
    static int tmp[MAX_N];
    int i = s, j = m + 1, idx = s, cnt = 0, res = 0;
    while(i <= m && j <= e){
        if(A[i] < A[j]) tmp[idx++] = A[i++], res += cnt;
        else tmp[idx++] = A[j++], cnt += 1;
    }
    while(i <= m) tmp[idx++] = A[i++], res += cnt;
    while(j <= e) tmp[idx++] = A[j++], cnt += 1;
    for(int k=s; k<=e; k++) A[k] = tmp[k];
    return res;
}
```

질문?

분할 정복의 예시 – 4

히스토그램에서 가장 넓은 직사각형

- 히스토그램에서 넓이가 가장 큰 직사각형을 구하는 문제
  - 배열에서 (j-i+1) * min(A[i], A[i+1], … , A[j])의 최댓값을 구하는 문제



- 배열을 단순히 절반으로 나눠서 효율적으로 해결할 수 있을까?

분할 정복의 예시 - 4

히스토그램에서 가장 넓은 직사각형

- 배열을 반으로 잘라보자.
    - A[m]을 포함하는 모든 직사각형들의 넓이 중 최댓값을 구하면
    - 고려하지 않은 구간은 [l, m-1] 또는 [m+1, r]에 완전히 포함됨

    - [l, m-1]과 [m+1, r]에 포함된 구간은 재귀적으로 잘 처리할 수 있으므로
    - [l, r]에서 A[m]을 포함하는 모든 구간을 확인하는 방법만 찾으면 된다.

분할 정복의 예시 – 4

히스토그램에서 가장 넓은 직사각형

- A[m]을 포함하는 모든 구간을 확인
  - 단순하게 확인하면 $O(N^2)$ / 더 빠르게 해야 함

  - [m, m]부터 시작해서 한 칸 씩 확장하는 방식으로 진행
  - 왼쪽과 오른쪽 중 어느 방향으로 확장해야 할까?

  - 확장 방향에 관계없이 (j-i+1)의 값은 동일하게 1 증가
  - 따라서 min(A[i], A[i+1], … , A[j])가 덜 작아지는 방향으로 확장
  - 확장은 r-l번 하므로 선형 시간에 처리 가능

- $T(n) = 2T(n/2) + O(n) = O(n \log n)$

분할 정복의 예시 - 4

히스토그램에서 가장 넓은 직사각형

```
int MaximumRectangle(int A[], int s, int e){
    if(s > e) return 0;
    if(s == e) return A[s];
    int m = (s + e) / 2;

    int res = A[m], cnt = 1, mn = A[m];
    int i = m - 1, j = m + 1;

    while(s <= i && j <= e){
        if(A[i] > A[j]) cnt++, mn = min(mn, A[i--]);
        else cnt++, mn = min(mn, A[j++]);
        res = max(res, cnt * mn);
    }
    while(s <= i){
        cnt++; mn = min(mn, A[i--]);
        res = max(res, cnt * mn);
    }
    while(j <= e){
        cnt++; mn = min(mn, A[j++]);
        res = max(res, cnt * mn);
    }
    res = max(res, MaximumRectangle(A, s, m-1));
    res = max(res, MaximumRectangle(A, m+1, e));
    return res;
}
```

질문?

분할 정복의 예시 – 5

BOJ 9537. 잘생긴 GCD

- 어떤 수열의 최대 공약수를 수열의 모든 원소들의 최대 공약수라고 정의하자.
- 어떤 수열의 점수를 수열의 최대 공약수와 수열의 길이의 곱으로 정의하자.
- 수열이 주어졌을 때, 수열의 연속한 부분 수열들 중 점수의 최댓값을 구하는 문제

- ex. {30, 60, 20, 20, 20}
  - 수열의 최대 공약수는 10, 수열의 길이는 5
  - 따라서 수열의 점수는 50
- ex. {60, 20, 20, 20}
  - 수열의 최대 공약수는 20, 수열의 길이는 4
  - 따라서 수열의 점수는 80

분할 정복의 예시 - 5

BOJ 9537. 잘생긴 GCD

- 마찬가지로 A[m]을 지나는 모든 구간들의 최대 점수를 구해보자.

- 구간 [l, r] 안에 존재하는 A[m]을 포함하는 구간 [i, j]는 다음과 같이 만들 수 있다.
  - [l, m] 구간에서 i 선택, [m, r] 구간에서 j 선택
  - [i, j]는 A[m]을 포함하는 구간

- 하지만 A[m]을 포함하는 구간을 모두 고려하는 것은 여전히 $O(n^2)$
- 더 빠르게 할 수 있을까?

분할 정복의 예시 – 5

BOJ 9537. 잘생긴 GCD

- 관찰 1. $gcd_k(A) = gcd(A_{1..k})$라고 정의했을 때, $gcd_*(A)$의 결과는 최대 $O(\log A_1)$가지
  - 만약 최대 공약수가 g에서 h로 바뀐다면 h는 g보다 작은 g의 약수
  - $h \le g/2$이고 1이 되면 더 이상 변하지 않으므로 최대 $O(\log A_1)$번만 바뀜

- 관찰 2. $l < i \le m$인 i가 $gcd(A_{i-1..m}) = gcd(A_{i..m})$이면 i가 끝점인 구간은 고려하지 않아도 됨
  - 최대 공약수는 바뀌지 않는데 구간의 길이는 늘어나므로 i 대신 i-1만 확인해도 됨
  - 마찬가지로 $gcd(A_{m..j}) = gcd(A_{m..j+1})$이면 j 대신 j+1만 확인해도 됨

- 관찰 3. 실제로 확인해야 하는 구간은 최대 $O(\log^2 A_m)$가지
  - 왼쪽 끝점을 선택하는 경우 $O(\log A_m)$가지
  - 오른쪽 끝점을 선택하는 경우 $O(\log A_m)$가지

분할 정복의 예시 – 5

BOJ 9537. 잘생긴 GCD

- 유효한 왼쪽 끝점과 오른쪽 끝점을 구하는데 $O(n \log A_m)$
- A[m]을 포함하는 구간을 확인하는데 $O(\log^3 A_m)$

- 시간 복잡도
  - 수열의 최댓값을 x라고 하면
  - $T(n) = 2T(n/2) + O(n \log x + \log^3 x) = O(n \log n \log x + n \log^3 x)$

분할 정복의 예시 - 5

BOJ 9537. 잘생긴 GCD

```
long long Solve(long long A[], int s, int e){
    if(s >= e) return A[s];
    int m = (s + e) / 2;
    auto res = max(Solve(A, s, m-1), Solve(A, m+1, e));

    vector<pair<long long, int>> l, r;
    l.emplace_back(A[m], m);
    r.emplace_back(A[m], m);
    for(int i=m-1; i>=s; i--){
        auto g = gcd(l.back().first, A[i]);
        if(g == l.back().first) l.back().second = i;
        else l.emplace_back(g, i);
    }
    for(int i=m+1; i<=e; i++){
        auto g = gcd(r.back().first, A[i]);
        if(g == r.back().first) r.back().second = i;
        else r.emplace_back(g, i);
    }
    for(auto [g1,le] : l) for(auto [g2,ri] : r) res = max(res, gcd(g1, g2) * (ri - le + 1));
    return res;
}
```

질문?

더 공부할 거리

세그먼트 트리

- 분할 정복의 원리를 이용해 구간의 합/최솟값 등을 빠르게 구하는 자료구조

센트로이드 분할

- 센트로이드: 트리에서 중간 지점의 역할을 하는 정점
- 트리에서 분할 정복을 할 수 있도록 도와주는 도구
- 배열에서 모든 구간을 고려하는 것 → 트리에서 모든 경로를 고려하는 것

오프라인 동적 연결성 문제

- 시간 축에 대한 분할 정복