

#4. 평방 분할

나정휘

<https://justicehui.github.io/>

산술 기하 부등식

AM-GM Inequality

- Arithmetic Mean 산술 평균
- Geometric Mean 기하 평균
- Inequality 부등식

산술 기하 부등식

음이 아닌 실수 $x_1, x_2, \dots, x_n$ 에 대해 아래 식이 성립함

- $\frac{x_1 + x_2 + \dots + x_n}{n} \geq \sqrt[n]{x_1 x_2 \dots x_n}$
- 등호가 성립할 조건은 $x_1 = x_2 = \dots = x_n$

$n = 2$ 인 경우

- $a + b \geq 2\sqrt{ab}$
- $a = b$ 일 때 등호 성립
- 두 식의 곱이 일정할 때 합의 최솟값을 찾을 수 있음

산술 기하 부등식

간단한 정렬 알고리즘

- 길이가 N 인 배열을 정렬하는 알고리즘
 - 배열을 크기가 B 인 N/B 개의 덩어리로 분할
 - 각 덩어리를 $O(N^2)$ 알고리즘으로 정렬
 - N/B 개의 덩어리는 각각 정렬되어 있는 상태
 - 각 덩어리의 맨 앞에 있는 원소들의 최솟값을 정답에 추가
 - N 번 반복하면 전체 배열이 정렬됨
- 시간 복잡도는?

산술 기하 부등식

간단한 정렬 알고리즘 - 시간 복잡도

- 길이가 N인 배열을 정렬하는 알고리즘
 - 배열을 크기가 B인 N/B개의 덩어리로 분할
 - 각 덩어리를 $O(N^2)$ 알고리즘으로 정렬
 - N/B개의 덩어리는 각각 정렬되어 있는 상태
 - 각 덩어리의 맨 앞에 있는 원소들의 최솟값을 정답에 추가
 - N번 반복하면 전체 배열이 정렬됨
- 시간 복잡도는?

$$B^2 * N/B = NB$$

$$N/B * N = N^2/B$$

$$NB + N^2/B$$

산술 기하 부등식

간단한 정렬 알고리즘 - 시간 복잡도

- 길이가 N인 배열을 정렬하는 알고리즘
 - 배열을 크기가 B인 N/B개의 덩어리로 분할
 - 각 덩어리를 $O(N^2)$ 알고리즘으로 정렬
 - N/B개의 덩어리는 각각 정렬되어 있는 상태
 - 각 덩어리의 맨 앞에 있는 원소들의 최솟값을 정답에 추가
 - N번 반복하면 전체 배열이 정렬됨
- 시간 복잡도는?
 - 곱이 일정하므로 $NB = N^2/B$ 일 때 최소가 됨
 - $B = \sqrt{N}$ 이면 전체 시간 복잡도는 $O(N\sqrt{N})$

$$B^2 * N/B = NB$$

$$N/B * N = N^2/B$$

$$NB + N^2/B$$

평방 분할

BOJ 14438 수열과 쿼리 17

- 1: A_i 를 v 로 바꾼다.
- 2: $A_i, A_{i+1}, \dots, A_j$ 에서 크기가 가장 작은 값을 출력한다.
- 단순히 구현하면 1번 쿼리 $O(1)$, 2번 쿼리 $O(N)$

평방 분할

BOJ 14438 수열과 쿼리 17

- 정렬 알고리즘에서 본 방식을 사용해 보자.
 - 배열을 크기가 B 인 N/B 개의 버킷으로 분할
 - 각 버킷의 최솟값을 전처리
 - $O(N)$ 에 가능
 - 1: 원소의 값을 바꾼 뒤, 그 원소가 속한 버킷의 최솟값을 갱신
 - 버킷의 크기는 B 이므로 $O(B)$ 에 가능
 - 2: 구간에 완전히 포함된 버킷은 버킷의 최솟값을 취하고, 일부만 겹친 버킷은 모든 원소를 확인
 - 구간에 완전히 포함된 버킷은 최대 N/B 개
 - 일부만 겹친 버킷은 최대 2개이므로 최대 $2B$ 개의 버킷의 원소를 확인
 - $O(N/B + 2B)$
- $B = \sqrt{N}$ 이면 전체 시간 복잡도는 $O(N + Q\sqrt{N})$

평방 분할

$A = \{1, 3, 5, 7, 9, 0, 2, 4, 6, 8, 2, 6\}$, $B = 3$

Update(5, 3)

Update(6, 5)

Query(1, 9)

1			0			2			2		
1	3	5	7	9	0	2	4	6	8	2	6

1			3			2			2		
1	3	5	7	9	3	2	4	6	8	2	6

1			3			4			2		
1	3	5	7	9	3	5	4	6	8	2	6

1			3			4			2		
1	3	5	7	9	3	5	4	6	8	2	6

평방 분할



```
constexpr int B = 400;
int N, Q, A[101010], M[101010/B];

void Calc(int id){
    int s = id * B, e = min(N, (id + 1) * B);
    M[id] = 1e9;
    for(int i=s; i<e; i++) M[id] = min(M[id], A[i]);
}

void Init(){
    for(int i=0; i*B<N; i++) Calc(i);
}

void Update(int x, int v){
    A[x] = v; Calc(x / B);
}

int Query(int l, int r){
    int res = 1e9;
    while(l <= r && l % B != 0) res = min(res, A[l++]);
    while(l <= r && r % B + 1 != B) res = min(res, A[r--]);
    if(l <= r) for(int i=l/B; i<=r/B; i++) res = min(res, M[i]);
    return res;
}
```