

# 슬라이딩 윈도우 – 구간의 최소값

- N개의 수  $A[1], A[2], \dots, A[N]$ 이 주어졌을 때,  $d[i] = A[i - L + 1] \sim A[i]$  중 최소값이라고 할 때,  $d$ 를 구하는 문제
- 세그먼트 트리를 이용하여 문제를 해결할 수 있으나 구간의 크기가  $L$ 로 고정된 경우  $O(N)$ 에 해결할 수 있다.
- $N = 12, L = 3$ 인 경우 아래와 같은 문제이다.

A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]	A[11]	A[12]
1	5	2	3	6	2	3	7	3	5	2	6

D[1] = 1	D[4] = 2	D[7] = 2	D[10] = 3
D[2] = 1	D[5] = 2	D[8] = 2	D[11] = 2
D[3] = 1	D[6] = 2	D[9] = 3	D[12] = 2

# 슬라이딩 윈도우 – 구간의 최소값

- 덱 을 이용하여 문제를 해결할 수 있다.
- 덱에 값이 증가하는 순서대로 저장하여 가장 앞에 있는 값을 최소로 합니다.
- ① 가장 앞에 있는 값의 인덱스와 현재 값의 인덱스가 L보다 많이 차이 나는지 검사해야 합니다.
- ② 가장 뒤에 있는 값이 현재 값보다 큰 지 검사해서 크면 덱에서 빼야 합니다.
- ③ 현재 값을 덱에 넣습니다.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]	A[11]
1	5	2	3	6	2	3	7	3	5	2	6

값	1	값	1	5	값	1	2	값	2	3
인덱스	0	인덱스	0	1	인덱스	0	2	인덱스	2	3

값	2	3	6	값	2	값	2	3
인덱스	2	3	4	인덱스	5	인덱스	5	6

값	2	3	7	값	3	3
인덱스	5	6	7	인덱스	6	8

값	3	5	값	2	값	2	6
인덱스	8	9	인덱스	10	인덱스	10	11

# 슬라이딩 윈도우 – 구간의 최소값

```
deque<pi> dq;
```

```
vi ans(N);
```

```
for (int i = 0; i < N; ++i) {
```

```
    int here = A[i];
```

```
    if (!dq.empty() && dq.front().second <= i - L) { 최소값이 될 수 없을 인덱스 차이
```

```
        dq.pop_front();
```

```
    }
```

```
    while (!dq.empty() && dq.back().first > here) { 최소값이 될 수 없을 크기
```

```
        dq.pop_back();
```

```
    }
```

```
    dq.push_back(pi(here, i));
```

```
    ans[i] = dq.front().first;
```

```
}
```

```
for (int num : ans) {
```

```
    printf("%d ", num);
```

```
}
```

