알고리즘 문제해결기법

제2주 Where Does logN come from ?

문제 04: 랜선 자르기 [백준 1654번]

K개의 랜선을 가지고 있다. K개의 랜선은 길이가 제각각이다. K개의 랜선을 잘라서 모두 N개의 같은 길이의 랜선으로 만들고 싶다. 예를 들어 300cm 짜리 랜선에서 140cm 짜리 랜선을 두 개 잘라내면 20cm은 버려야 한다.
기존의 K개의 랜선으로 N개의 랜선을 만들 수 없는 경우는 없다고 가정하자. 그리고 자를 때는 항상 정수길이만큼 자른다고 가정하자. 이 때 만들 수 있는 최대 랜선의 길이를 구하는 프로그램을 작성하시오.
K는 1이상 10,000이하의 정수이고, N은 1이상 1,000,000이하의 정수이다. 항상 K≦N 이다.

문제 04: 랜선 자르기 [백준 1654번]

K=8
N=15

Decision Problem 버전에 대해서 생각해본다.

즉, 길이 L짜리 랜선 N개를 만들수 있는가?

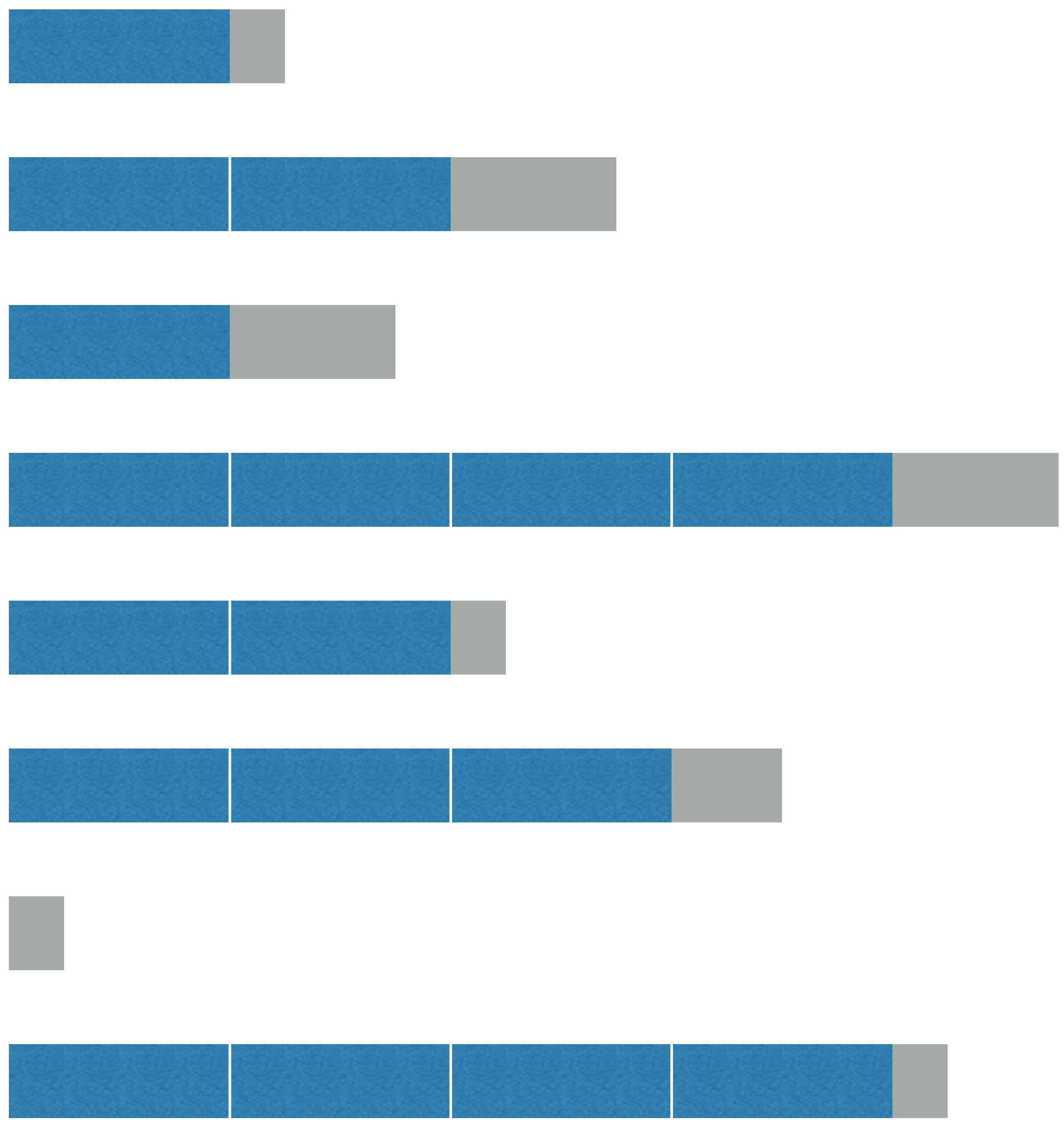
혹은

길이가 L짜리인 랜선을 최대 몇개 만들 수 있는가?

문제 04: 랜선 자르기 [백준 1654번]

길이 L짜리 랜선을 몇 개 만들 수 있는가?

K=8
N=15



$$\sum_{i=1}^{K} \lfloor d_i / L \rfloor$$

decision problem 버전은 쉽다.

문제 04: 랜선 자르기 [백준 1654번]

Given N positive integers $d_1, d_2, \ldots, d_N$,

find the maximum integer L with $\displaystyle\sum_{i=1}^{K} \lfloor d_i / L \rfloor \geq N$.

Let $f(L) = \displaystyle\sum_{i=1}^{K} \lfloor d_i / L \rfloor$. ← Computable in O(K)

$f(L)$ is a non-increasing function.

문제 04: 랜선 자르기 [백준 1654번]

$$upper = \min\{\max_{i=1...K} d_i, \frac{1}{N}\sum_{i=1}^{K} d_i\}$$

```
for (int L = upper; L >= 1; L—)
    if (f(L) >= N)
        return L;
```

f(L) is non-increasing

$$O(K \cdot d_{max})$$

문제 04: 랜선 자르기 [백준 1654번]

```
while (lower <= upper) {
    int mid = lower + (upper - lower)/2;  ←——— It's better. Why?
    int count = f(mid);
    if (count < N)
        upper = mid-1;
    else {
        if (mid > best) best = mid;
        lower = mid+1;
    }
}
```

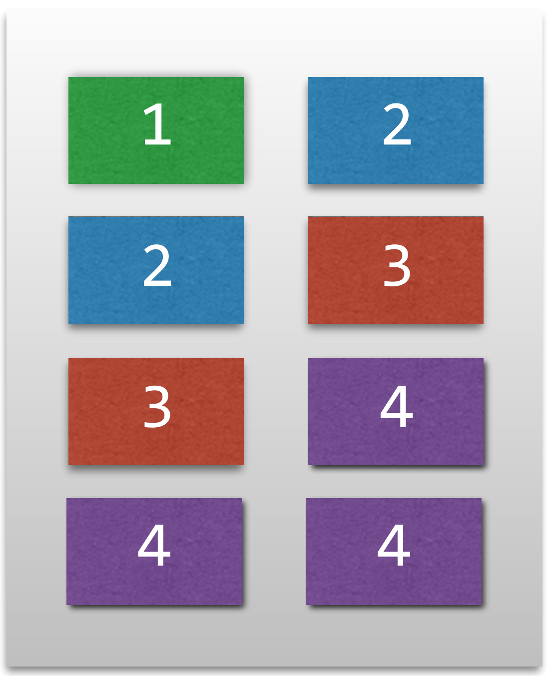
$$O(K \log d_{max})$$

문제 05: 파워포인트

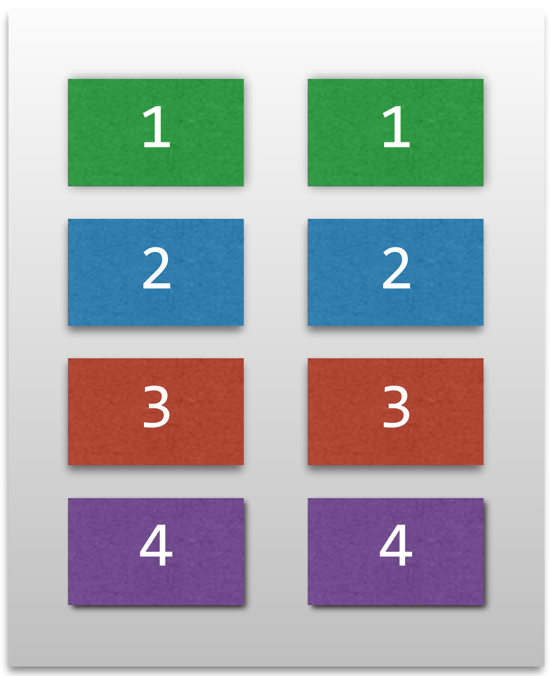
상품에 부착할 라벨을 인쇄하려고 한다. 모든 라벨은 내용은 서로 다르지만 크기와 모양은 같다. 한 장의 파워포인트 슬라이드에 라벨들을 디자인한 뒤 이 슬라이드를 여러 장 프린트한 후 가위로 잘라서 필요한 수량의 라벨들을 만들려고 한다.
모두 N종류의 라벨을 만들어야 한다. 각 라벨 종류별로 필요한 수량이 입력으로 주어진다. 즉 N개의 양의 정수가 입력으로 주어진다. 이 정수들을 q_1, q_2,⋯,q_N이라고 부르자. 한 장의 파워포인트 슬라이드에는 총 M개의 라벨을 넣을 수가 있다. N≦M이라고 가정한다.
예를 들어 입력된 정수들이 {8, 15, 24, 30}이고 M=8이라고 하자. 이 경우 만약 슬라이드에 1번 라벨을 1장, 2번 라벨을 2장, 3번 라벨을 2장, 그리고 4번 라벨을 3장 넣는다면 이 슬라이드를 12장 프린트하면 필요한 모든 라벨 개수를 충족한다. 만약 4종류의 라벨을 각각 2개씩 슬라이드에 넣는다면 슬라이드를 15장 프린트해야할 것이다. 프린트 횟수를 최소가 되도록 배치했을 때 필요한 프린트 횟수를 출력하는 프로그램을 작성하라. N과 M은 1,000,000이하이고, 입력 정수들은 모두 10^8 이하이다.

문제 05: 파워포인트

q_1=8, q_2=15, q_3=24, q_4=30이고 M=8인 경우의 예



12장 프린트하면 1번 12장, 2번 24장, 3번 24장, 4번 36장



15장 프린트하면 모두 30장씩

문제 05: 파워포인트

Trivial Lower and Upper Bound

- **최소 프린트 횟수를 p_{min}이라고 하자. 그러면 $1 \leq p_{min} \leq q_{max}$ 이다.**
- **P장을 프린트해서 모든 라벨을 필요한 개수만큼 만들수 있는가?**
  - i-번째 라벨은 슬라이드에 적어도 $\lceil \frac{q_i}{P} \rceil$ 개가 배치되어야 함
  - 따라서 $f(P) = \sum_{i=1}^{N} \lceil \frac{q_i}{P} \rceil \leq M$ 이면 가능
  - O(N) 시간에 검사 가능
  - $f(P)$는 비증가함수
  - 이진검색으로 $O(N \log q_{max})$ 시간복잡도

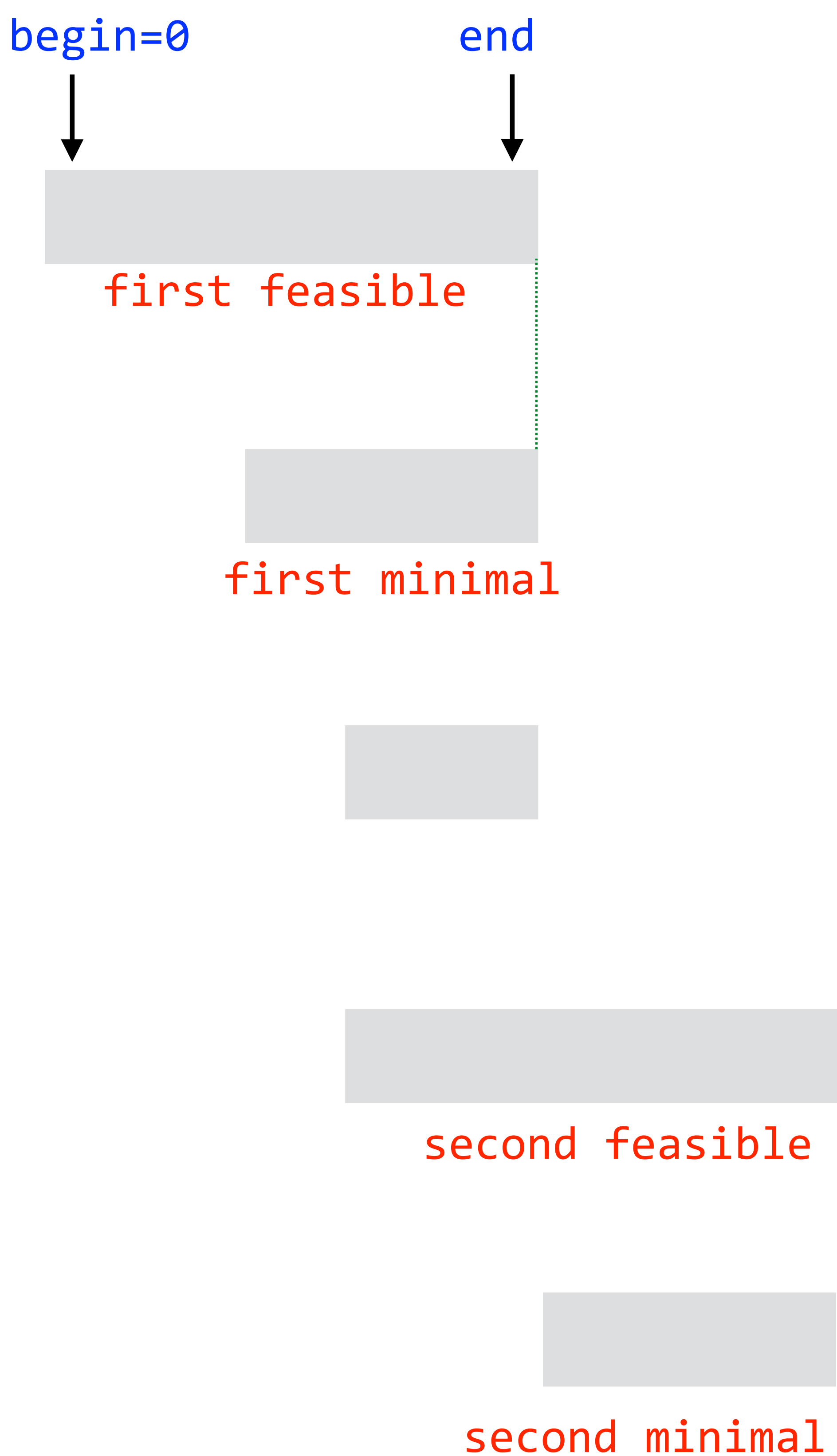
문제 06: 부분 수열의 길이

음수일 수도 있다.

길이가 N인 정수 수열과 정수 K가 주어진다. 이 때, 합이 K보다 크거나 같은 연속 부분 수열 중에서 길이가 가장 짧은 것을 찾는 프로그램을 작성하시오. N ($1 \le N \le 500{,}000$)과 X ($-10^9 \le K \le 10^9$)이다.

모든 minimal한 구간을 찾아서
그 중에 가장 짧은 것을 고른다.

양수인 경우의 minimal feasible 구간 찾기



합이 K이상이 될때까지 end를 전진

합이 K이상인 동안 begin을 전진

begin을 한 칸 전진

(minimal은 다른 minimal을 포함하지 않음)

합이 K이상이 될때까지 end를 전진

합이 K이상인 동안 begin을 전진

첫 번째 minimal 구간 찾기

왜 이 구간을 지나쳤을까?

처음으로 합이 K이상이 될때까지 더했다.
전에는 이것이 첫 번째 feasible한 구간이었다. 하지만…

처음 나오는 음수는 버려야 한다.

첫 번째 minimal 구간 찾기

왜 이 구간을 지나쳤을까?

K=32

3	4	1	3	-17	6	5	22	-5	3	16	9	2	23	19	3	6	31	5
3	7	8	11	-6	0	5	27	22	25	41								

처음으로 합이 K이상이 될때까지 더했다.
전에는 이것이 첫 번째 feasible한 구간이었다. 하지만…

합이 음수가 되는 구간은 도움이 안됨. 버려도 될까?

첫 번째 minimal 구간 찾기

② 이 구간의 합은
음수가 아니다. 왜?

③ 따라서 이 구간의 합은 K 이상이 될 수 없다.

① 어떤 minimal feasible 구간이 (파란 구간)
노란 구간에 속해 있다고 가정해보자.

K=32

3	4	1	3	-17	6	5	22	-5	3	16	9	2	23	19	3	6	31	5
3	7	8	11	-6	0	5	27	22	25	41								

한 번도 sum이 K이상이 되지 않으면서
처음으로 sum이 0 혹은 음수가 되는 구간이다.
그러면 어떤 minimal한 feasible 구간도 이 부분에 포함되지 않는다.

첫 번째 minimal 구간 찾기

③ 따라서 이 구간의 합은
음수이다.

④ 따라서 만약 파란구간의 합이 K이상이면
이 구간만의 합도 K이상이다. 이것은 원래 파란구간이
minimal이라는 가정과 모순된다.

② 이 구간의 합은
음수가 아니다. 왜?

① 어떤 minimal feasible 구간이 (파란 구간)
노란 구간과 겹친다고 가정해보자.

K=32

3	4	1	3	-17	6	5	22	-5	3	16	9	2	23	19	3	6	31	5
3	7	8	11	-6	0	5	27	22	25	41								

한 번도 sum이 K 이상이 되지 않으면서
처음으로 sum이 0 혹은 음수가 되는 구간이다.
그러면 어떤 minimal한 feasible 구간도 이 부분과 겹치지
않는다.

첫 번째 minimal 구간 찾기

K=32

3	4	1	3	-17	6	5	-4	-5	3	16	9	3	23	19	3	6	31	5
3	7	8	11	-6	6	11	7	2	5	21	30	33						

한 번도 sum이 K 이상이 되지 않으면서
처음으로 sum이 0 혹은 음수가 되는 구간이다.
어떤 minimal한 feasible 구간도
이 부분과 겹치지 않는다.

따라서 이 구간은 버려도 된다.

이전 구간은 버리고
sum은 새로 시작한다.

첫 번째 minimal 구간 찾기

K=32		6	5	-4	-5	8	16	5	1	23	19	3	6	31	5
	0	6	11	7	2	10	26	31							

이전 구간은 버리고
sum은 새로 시작했다.

지금 까지 sum은 계속하여 양수를 유지했고
계속해서 K 미만이다.

이 초록구간에 feasible한 구간이 포함될수는 없다. 왜?

첫 번째 minimal 구간 찾기

① 초록 구간은 적어도 하나의 minimal feasible한 구간을 포함한다. 당연 !

② 초록 구간에 포함된 minimal feasible 구간은 반드시 초록 구간의 끝에서 끝난다. **왜?**

③ 초록 구간에 포함된 minimal feasible 구간의 시작점은?

K=32

	6	5	-4	-5	8	16	5	13	23	19	3	6	31	5
0	6	11	7	2	10	26	31	44						

지금 까지 sum은 계속하여 양수를 유지했고
처음으로 K 이상이 되었다.

첫 번째 minimal 구간 찾기

초록 구간에 포함된 minimal feasible 구간의 시작점은?

첫 번째 minimal 구간 찾기

초록 구간에 포함된 minimal feasible 구간의 시작점은?

minimal feasible

K=32

	6	5	-4	-5	8	16	5	13	23	19	3	6	31	5
0	6	11	7	2	10	26	31	44						

이 자리를 begin이라고 부르자.

i

begin에서 i사이에서
sum이 44-32=12 이하인 마지막 지점

다음 minimal 구간 찾기

2. 이 자리를 새로운 begin이라고 부르자.

K=32

	6	5	-4	-5	8	16	5	13	23	19	3	6	31	5
0	6	11	7	2	10	26	31	44	67					
							5	18	41					

3. 이렇게 새로운 sum을 계산할 필요는 없다.
현재의 sum에서 base를 빼면 된다.

1. 다음 minimal feasible 구간은
이것 보다 뒤에서 시작된다. 즉 여기까지는 버려도 된다.
26을 base라고 부르자.

첫 번째 minimal 구간 찾기

1. 누적합 sum을 구해나간다.
2. 만약 sum ≤ 0 이 되면 그때까지의 정수들은 모두 버리고 새로 시작한다.
3. 만약 sum ≥ K가 되면 그 지점이 첫 번째 minimal 구간의 끝점이다.
4. 첫 번째 minimal 구간은 sum이 sum-K 이하인 마지막 지점 다음 위치에서 시작된다.
5. 4에서 찾은 시작점을 포함하여 그 앞 부분을 모두 버리고 새로 시작한다.

첫 번째 minimal 구간 찾기

```
/* separately handle the case where K < 0 */

int sum[MAX+1] = {0, }, base = 0, begin = 1;
int minLen = ∞;
for (int i=1; i<=N; i++) {      /* data는 data[1],…,data[N]이다. */
    sum[i] = sum[i-1] + data[i];
    if (sum[i]-base <= 0) {
        base = sum[i];
        begin = i+1;
    }
    else if (sum[i]-base >= K) {
        int k = latestIndexLessThanOrEqualTo(sum[i]-K, begin, i);
        if (i - k < minLen)
            minLen = i - k;
        base = sum[k];
        begin = k + 1;
    }
}
return minLen;
```

begin ≤ k < i이면서
sum[k] ≤ sum[i]-K인 가장 큰 k를 반환
(그런 k가 없을 경우 begin-1을 반환)

latestIndexLessThanOrEqualTo(V, begin, end)

- begin≤k<end이면서 sum[k]≤sum[end]-K인 가장 큰 k를 반환
- sum[begin]에서 sum[end-1]까지의 수열이 오름차순이라는 보장은 없음
- 단순하게 sum[begin]에서 sum[end-1]까지 검사하는 방법

N=16
K=N/2 = 8

data		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
sum	0	1	2	3	4	5	6	7	8	9	10	11	12	13				

base = 0

begin
= k

end

k=begin임을 알아내기 위해서 이 값들을 모두 검사해야 한다.
즉 O(K)개의 값을 검사해야 함

latestIndexLessThanOrEqualTo(V, begin, end)

- begin≤k<end이면서 sum[k]≤sum[end]-base-K인 가장 큰 k를 반환
- sum[begin]에서 sum[end-1]까지의 수열은 오름차순이라는 보장은 없음
- 단순하게 sum[begin]에서 sum[end-1]까지 검사하는 방법

N=16
K=N/2 = 8

data		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
sum	0	1	2	3	4	5	6	7	8	9	10	11	12	13			

base = 1

이제 여기가 begin이 된다.

end

다시 sum[2]에서 sum[8]까지 K-1개의 값을 검사해야 함

따라서 전체 시간복잡도는 N/2×N/2 = O(N²)이 된다.

- **최소 우선순위 큐 (minimum priority queue)는 다음의 두 가지 연산을 지원하는 자료구조**
  - `insert(x)`: 새로운 원소 x를 삽입
  - `extractMin()`: 최소값을 삭제하고 반환
  - `peekMin()`: 최소값을 삭제하지 않고 반환
- **이진 힙(binary heap)은 우선순위 큐를 구현하는 가장 잘 알려진 방법**
  - 두 연산 모두 O(logN) 시간에 지원

우선순위 큐

```
int sum[MAX+1] = {0, }, base = 0, begin = 1;
int minLen = ∞;
for (int i=1; i<=N; i++) {.  /* data는 data[1],…,data[N]이다. */
    sum[i] = sum[i-1] + data[i];
    insert(new Item(i,sum[i]));  ← 인덱스 i와 sum[i]를 저장하는 Item 객체를 우선순위큐에 삽입
    if (sum[i]-base <= 0) {
        base = sum[i];
        begin = i+1;
    }
    else if (sum[i]-base >= K) {
        maxIndex = begin-1;
        while (peekMin().val <= sum[i]-K) {
            item = extractMin();
            if (item.index > maxIndex)
                maxIndex = item.index;
        }
        if (i - maxIndex < minLen)
            minLen = i - maxIndex;
        base = sum[maxIndex];
        begin = maxIndex + 1;
    }
}
return minLen;
```

어떤 sum값도 큐에 두 번 들어가지 않는다.
따라서 시간복잡도는 O(NlogN)

C, C++, Java에서의 우선순위 큐

- C
  - Binary heap을 이용해서 직접 구현
  - https://www.youtube.com/watch?v=2bp2ZSS3O0g&list=PL52K_8WQO5oUuH06MLOrah4h05TZ4n38l&index=14
- C++
  - `std::priority_queue` 클래스
- Java
  - `Class PriorityQueue<E>`