Arcade Frenzy

exc

arr[] = { 2, 4, 6, 10, 2, 1 } target = 12

inc

curr sum = 0 ~~2~~ ~~6~~ ~~12~~ ~~10~~ ~~20~~ ~~16~~ ~~10~~ ~~12~~ ~~2~~ 3

$\boxed{\text{len} = \cancel{3}\ 2}$

```java
public int arcadeFrenzy(int[] scores, int k) {
    int n = scores.length;

    int currSum = 0;
    int inc = -1;
    int exc = -1;

    int minLen = Integer.MAX_VALUE;

    while (true) {
        boolean f1 = false;

        while (inc + 1 < n && currSum < k) {
            inc++;
            currSum += scores[inc];

            f1 = true;
        }

        boolean f2 = false;

        while (exc < inc && currSum >= k) {
            int len = inc - exc;
            minLen = Math.min(len, minLen);

            exc++;
            currSum -= scores[exc];

            f2 = true;
        }

        if (f1 == false && f2 == false) {
            break;
        }
    }

    return minLen == Integer.MAX_VALUE ? -1 : minLen;
}
```

psum

exc

-1 0 1 2 3 4 5

{ 2, 7, 3, -8, 4, 10 }

K = 12

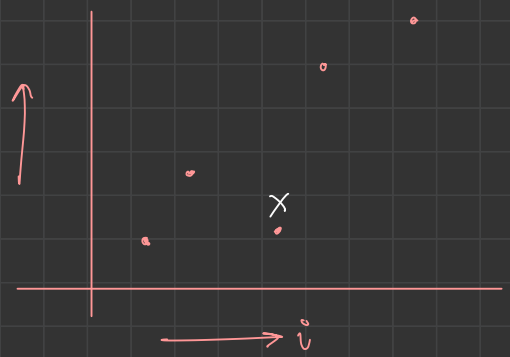
inc

14

len = 2

currSum = ~~0~~ ~~2~~ ~~9~~ ~~12~~ ~~10~~ ~~2~~ ~~6~~ ~~16~~ 9

minLen = ~~∞~~ 3



$\boxed{\text{minLen} = \cancel{\infty}\ \cancel{3}\ 2}$

0	1	2	3	4	5
2	7	3	-8	4	10

K = 12

currsum = $\cancel{0}$ $\cancel{2}$ $\cancel{9}$ $\cancel{12}$ $\cancel{4}$ $\cancel{8}$ 18

Monotonic Queues

inc queues

monotonic

(8,4) (18,5)

dq

sum, idx

(cum. sum)

1. check currSum >= K len = i+1;
2. try to exc. people len = (i - exc.idx)
   ↳ people to be exc are at first pos of dq
3. Make dq Monotonic
4. add yourself

```java
public int arcadeFrenzy(int[] scores, int k) {
    int n = scores.length;

    Deque<Pair> dq = new ArrayDeque<>();

    long currSum = 0;
    int minLen = Integer.MAX_VALUE;

    for (int i = 0; i < n; i++) {
        currSum += scores[i];

        if (currSum >= k) {
            int len = i + 1;
            minLen = Math.min(len, minLen);
        }

        while (dq.size() > 0 && currSum - dq.getFirst().sum >= k) {
            int len = i - dq.getFirst().idx;
            minLen = Math.min(len, minLen);
            dq.removeFirst();
        }

        while (dq.size() > 0 && dq.getLast().sum > currSum) {
            dq.removeLast();
        }

        dq.addLast(new Pair(currSum, i));
    }

    return minLen == Integer.MAX_VALUE ? -1 : minLen;
}
```

0 1 2 3 4 5

{2, 7, 3, -8, 4, 10}

K = 12

currsum = ~~0~~ ~~2~~ ~~9~~ ~~12~~ ~~4~~ ~~8~~ 18

minLen = ~~∞~~ ~~3~~ 2

(8,4) (18,5)

dq

{TC: O(N)
SC: O(N)}