今日の課題（Java）: **Streaming Interval Union Length**

問題

あなたはログ収集システムを作っている。
システムには「時刻区間」が次々に登録され、都度「これまでに登録された区間の合計被覆長（union length）」を知りたい。

ここで区間は **半開区間 [l, r)**（l は含む、r は含まない）とする。

入力

最初に整数 `Q`（クエリ数）。

続いて `Q` 行のクエリが与えられる：

- `ADD l r`：区間 `[l, r)` を追加する（同じ区間が何回追加されてもよい）
- `LEN`：いままで追加された全区間の **被覆長の合計** を出力する

制約（目安）

- `1 <= Q <= 2 * 10^5`
- `0 <= l < r <= 10^9`
- `LEN` は1回以上出る

出力

`LEN` クエリのたびに、被覆長（union length）を1行で出力せよ。

例

入力

```
7
ADD 1 5
LEN
ADD 2 3
LEN
ADD 10 20
ADD 4 12
LEN
```

出力

```
4
4
19
```

説明：

- `[1,5)` の長さは 4
- `[2,3)` を足しても union は変わらないので 4
- その後 `[10,20)` と `[4,12)` を足すと union は `[1,20)` になるので 19

ねらい（実装上のテーマ）

愚直に区間を全部持って毎回マージすると遅い。
TreeMap（平衡BST）で「互いに重ならない区間集合」を維持し、`ADD` のたびに重なる区間を吸収・統合しつつ、合計長 `total` を更新する方針で解け。

実装ヒント（使ってよい標準機能）

- `TreeMap<Long, Long>` を使い、キーを `start`、値を `end` として「非重複区間」を保持する
- 追加する `[l,r)` に対して
  - `floorEntry(l)` で左側から重なり得る区間を探す
  - `ceilingEntry(l)` / `higherEntry(start)` を辿りながら重なり区間を削除して統合
- `total`（long）を持っておいて
  - 区間を削除したら `total -= (end-start)`
  - 新しい統合区間を入れたら `total += (newEnd-newStart)`

注意

- 半開区間なので、例えば `[1,3)` と `[3,5)` は **連結して union 長は 4**（隙間なし）
  → 実装では「重なる」判定を `nextStart <= curEnd` のように **接している場合もマージ**すると扱いやすい。

追加チャレンジ（余力があれば）

1. `DEL l r`（区間削除）も対応してみる（かなり難化）
2. `KTH x`（union 上で x 番目の点が属する実時刻を返す）を考えてみる（座標圧縮＋セグ木の発想へ）

提出形式

- `Main.java` 1ファイル
- 入力は標準入力、出力は標準出力
- `Q` が大きいので **高速I/O**（BufferedInputStream + 自前パーサ or FastScanner）推奨