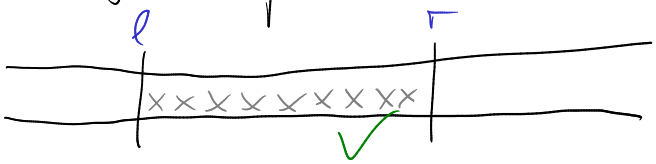
RMQs für andere Anfragen

- Range Equal Queries



↳ Was gilt, wenn alle Werte in einem Intervall gleich sind? → In diesem Fall ist das Min = Max.

↳ Für das Min. haben wir RMQs

↳ Für das Max haben wir RMaxQs

↳ Hier speichern wir die Pos. der Max, statt Min

↳ Bei den Cart. Bäumen sind die Knoten an den Max.

↳ Wenn $RMQ(l,r)^{**} = RMaxQ(l,r)^{**}$, dann RangeEqualQuery$(l,r)$ = true

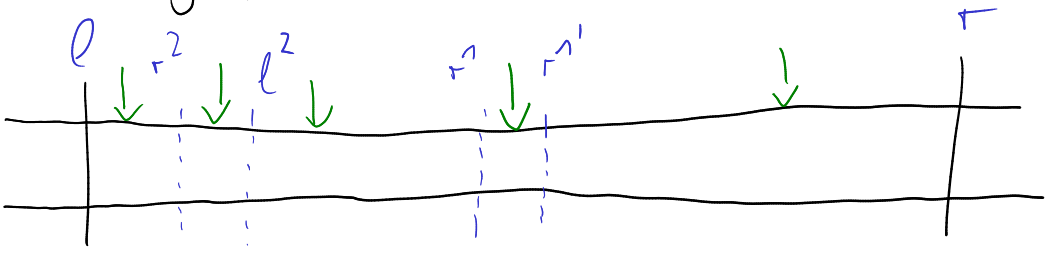
(Oder prüfe ob Werte an den Stellen * und ** gleich sind)

Range Top-10 Queries

↳ Wir nutzen wieder eine RMaxQ-Datenstruk.

↳ Wir stellen nun RMaxQs, bis wir die 10 größten Elemente haben

m RMaxQ



↳ Da wir nur die 10 größten Elemente haben wollen, ist die Anzahl der Queries konstant.

↳ Da wir nur konstant viele Queries haben können wir Range Top-10 Queries in konstanter Zeit beantworten

- Range Less-x Queries $\Leftrightarrow$ data[RMaxQ(l,r)<x

TLX-Bibliothek

↳ enthält viele nützliche Code-Schnipsel

In der Praxis sparen wir uns die Cart. Bäume.

↳ Den entsprechenden Eintrag in der Tabelle zu finden (der die Ergebnisse für die partielle RMQ enthält) dauert lange → evtl Cache-Miss

↳ Das Berechnen der Cart. Bäume dauert lange 1) Baum „konstruieren"

2) Bitmuster erzeugen (2s+1 Bits)

[3) einmalig alle Lösungen konstruieren]

↳ In der Praxis ist es besser über die partiellen Blöcke zu scannen

↳ Generell gilt: Auf jedem Fall testen, wie schnell Scannen ist

↳ Bei RMQs lohnt es sich bis ~ 1024 Elemente

↳ Intervalle sind in diesem Fall sehr klein:

$\frac{\lg n}{4}$ → selbst für sehr lange Eingaben $\ll 1024$