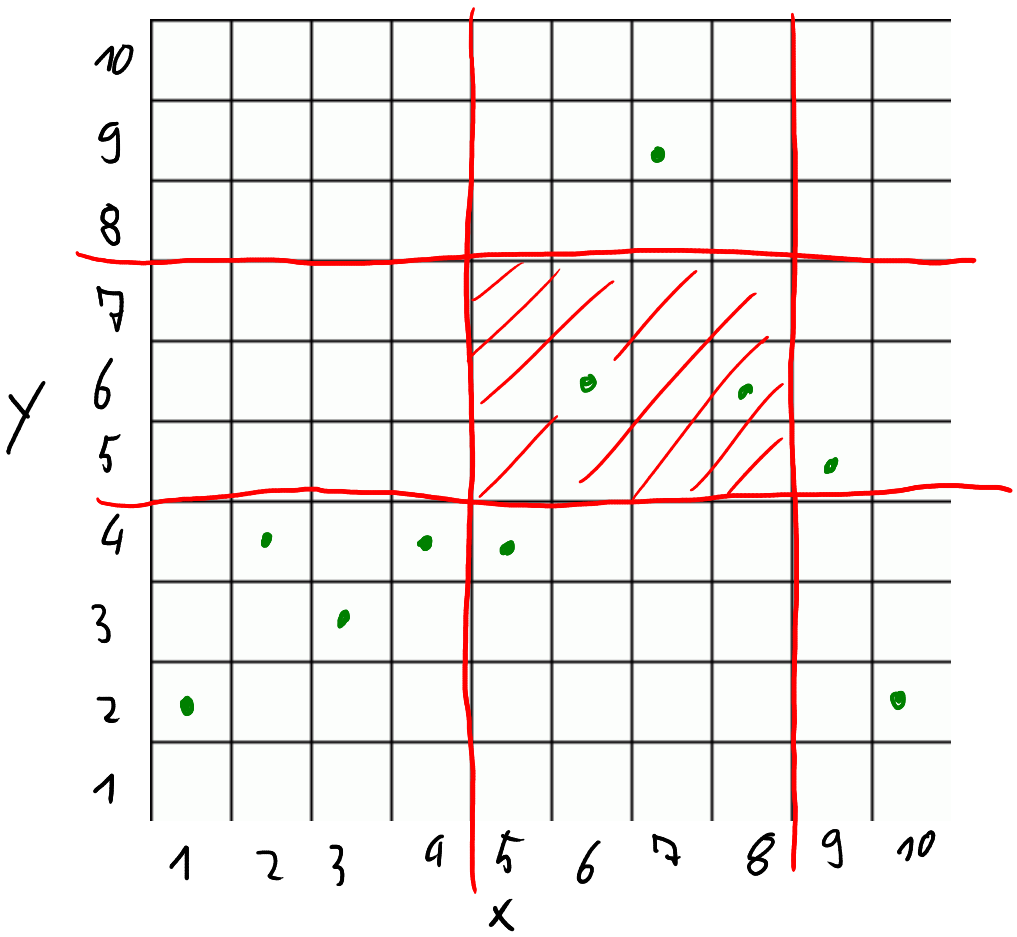
Bereichsanfragen mit WT

$$Q = [x_\ell = 5, x_r = 8] \times [y_b = 5, y_t = 7]$$



1. Sortiere Eingabe nach x-Koordinaten
2. Erstelle Text T aus Y-Koordinaten (in Reihenfolge nach x-Koor. sortiert) $* \sigma = n$
3. Erstelle WT für T

Um die Anfragen zu beantworten stellen wir Access-Anfragen auf dem WT für die Positionen $[x_\ell, x_r]$

↳ Bei Traversieren des WT überprüfen wir ob die Punkte in $[y_b, y_e]$ sind und geben nur die Punkte aus, die in diesem Intervall liegen

Mehr Informationen über WT
↳ Navarro, Wavelet Trees for All

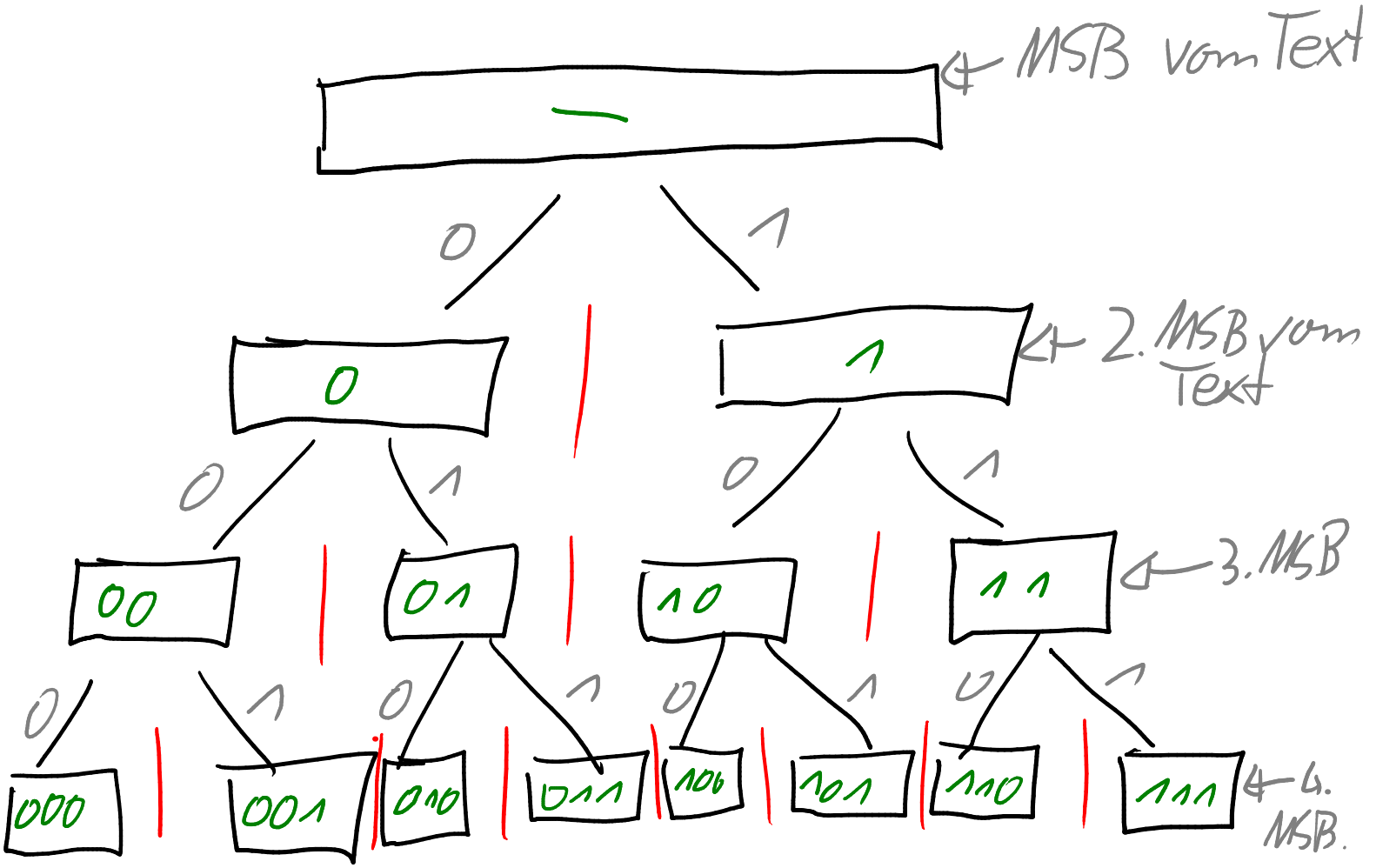
https://www.sciencedirect.com/science/article/pii/S1570866713000610

Wavelet-Tree-Konstruktion in der Praxis

- Bei WTs hängt vieles von der Alphabetgröße ab
  ↳ Alphabet reduzieren, in einem reduzierten Alphabet kommen alle Zeichen im Text vor
  z.B.: Bei ASCII, enthält viele Steuerungssymbole im unteren Bereich. Die kommen in den wenigsten Texten vor.
  Oder DNA mit $\sigma = 4$ und A,C,G,T die im ASCII 1 Byte pro

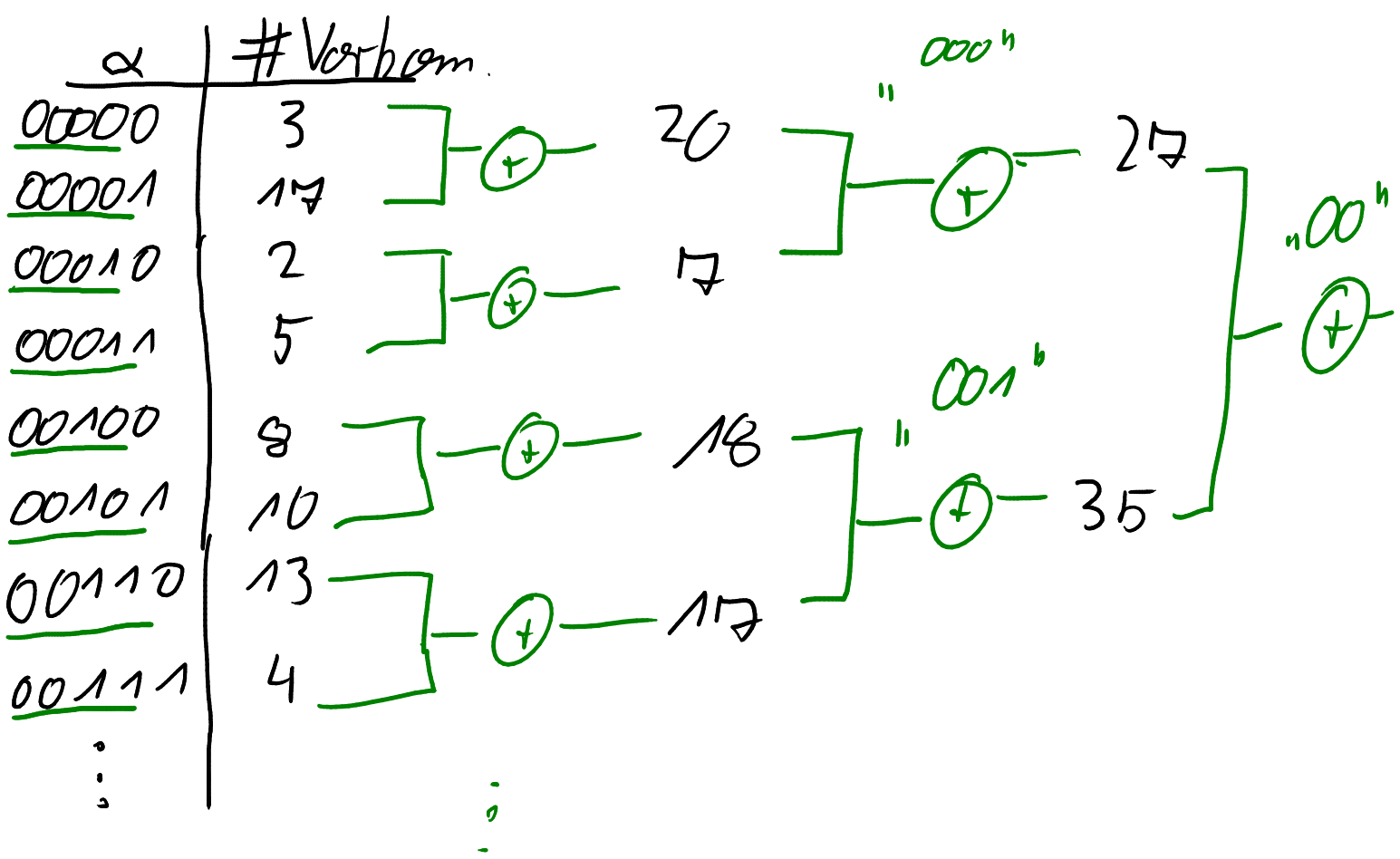
Zeichen brauchen

- Erste Level vom WT sind die MSBs in Textreihenfolge
- Anschließend konstruieren wir den WT von unten nach oben



Betrachte Bitpräfix um Grenzen der einzelnen Kinder zu berechnen.

Dazu betrachten wir das Histogram.



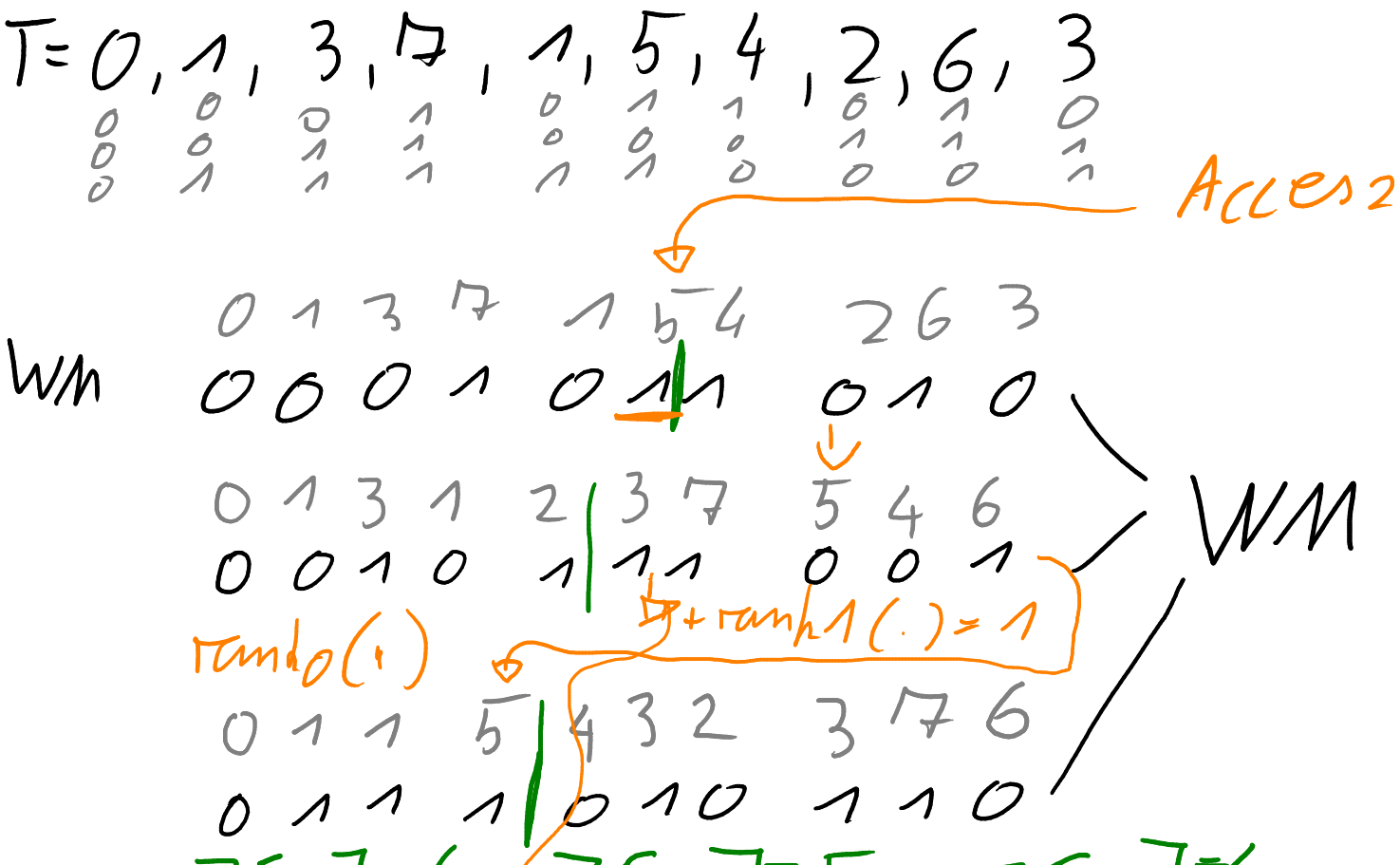
- Wir betrachten Level-wise WTs
  - Wir haben ein Bitvektor pro Level und nicht einen pro Kind.
  - In der Praxis brauchen die Pointer auf Kinder für große Alphabete viel Platz

- Wavevet-Trees haben auch Nachteile (gerade bei großen Alphabeten)
  - Entweder wir müssen uns merken wo die Intervalle anfangen oder Pointer auf die Kinder speichern ODER wir brauchen sehr viele binäre rank/select-Anfragen

Diese Anfragen/Pointer sind notwendig um sich im Baum zurechtzufinden

- Alternative zu WTs ist die Wavelet-Matrix
  - Erlaubt die gleichen Anfragen wie ein WT in der gleichen asymptotischen Zeit
  - Braucht aber weniger binäre rank/select-Anfragen

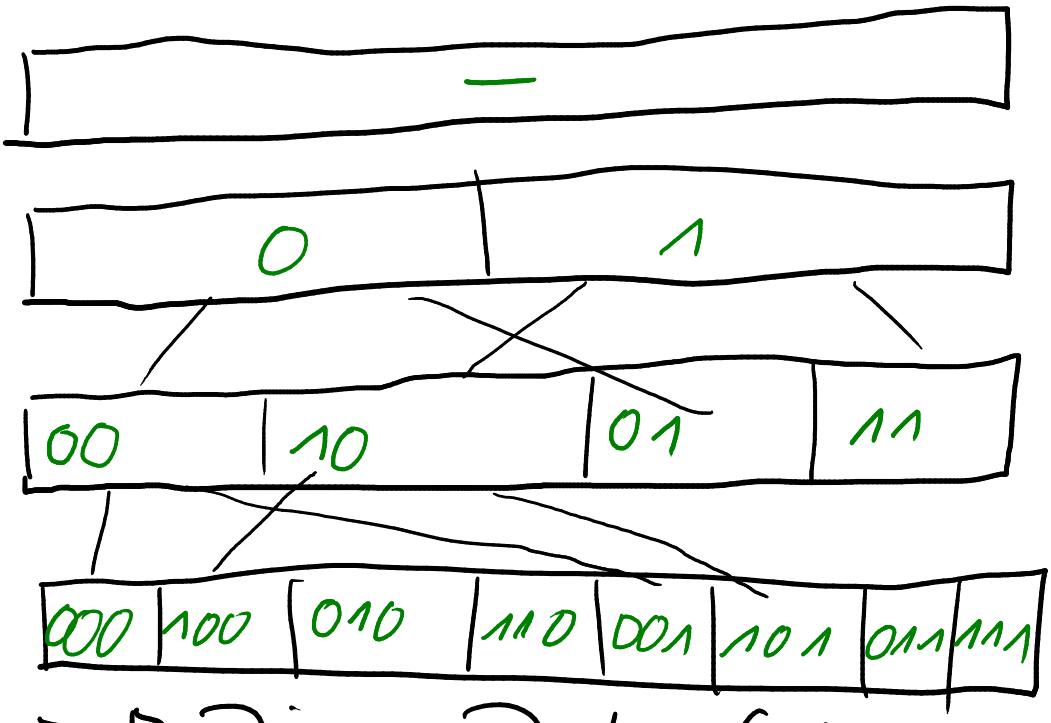
Wie sieht ein WM aus?



„Sortiere stabil nach dem aktuell betrachteten Bit"

- Wir merken uns noch die Anzahl der Oen pro Level
- Somit haben wir auf jedem Level nur 2 Intervalle und müssen die Grenzen nicht aufwendig berechnen oder abspeichern

- Aber auch in der WM gibt es „Bereiche" in denen bestimmte Bitpräfixe vorkommen.



→ Diese Reihenfolge der Bitpräfixe heißt bit-reversal-Permutation

- Wir können den gleichen Trick wie beim WT zum Bestimmen der Bereichsgrenzen anwenden.