Next: Fusion Tree (Fredman, Willard ’90)

Static, $O(n)$ space
query time $O\left(\frac{\log n}{\log \log n}\right)$

beats $\log n$, regardless of $U$!!

basic idea - deg-b search tree, with $b \approx \sqrt{w}$

how to search among $b$ numbers in $O(1)$ time?

Compress $b$ numbers in one word

\[ \begin{align*}
\{8, 10, 14, 15\} &= \{0100, 0110, 1110, 1111\} \\
q_0 &= 00011 \\
z &= 0010
\end{align*} \]

idea - trie

\[ \log b \]

\[ b \text{ leaves} \]

\[ O(b) \text{ nodes} \]

Can encode compressed trie in a word

\[ b \log \log U \leq b \log w \leq \log w = o(w) \]
How to query q:

- Follow path, & get element \( z \)
- \( \text{index of} \ \hat{z} \) may be wrong
- by nonstandard word op \( \text{in } O(1) \text{ time} \)
- find first bit where \( q \) & \( z \) differ
- \( \text{in } O(1) \text{ time} \)
- get index of ans
- by another nonstandard word op \( \text{in } O(1) \text{ time} \)

Can simulate everything with standard word ops (bitwise and/or, add, multiplication)

but very messy!

---

**Combine vEB & Fusion Trees**

\( O(n) \) space

query time \( O( \min \{ \log \log U, \frac{\log n}{\log w} \} ) \)

\[ \leq O( \min \{ \log w, \frac{\log n}{\log w} \} ) \]

\[ \leq O( \sqrt{\log n} ) \]

**Remark**

- can use Beame–Fich to improve query time

\( O( \min \{ \frac{\log \log U}{\log \log \log U}, \frac{\log n}{\log w} \} ) \)

\[ = O( \min \{ \frac{\log w}{\log \log w}, \frac{\log n}{\log w} \} ) \]
\[
S(n) = n^8 S(n^{1/8}) + O\left(\frac{n^{81/8}}{\log n}\right)
\]

\[
\Rightarrow O(n)
\]

\[
Q(n) = Q(n^{1/8}) + O\left(\frac{\log n}{\log \log n}\right)
\]

\[
\Rightarrow O\left(\frac{\log n}{\log \log n}\right)
\]

**Rmk** - this is optimal in terms of $\alpha$ because of matching lower bd in "call-probe model"

**Rmk** - for priority queues: can do better

Han-Thorup '02: $O\left(\frac{\log \log n}{\log n}\right)$ expected time

\[
\Rightarrow \text{can sort $n$ integers in } O\left(n \frac{\log \log n}{\log n}\right) \text{ expected time}
\]
Problem: Orthogonal Range Searching

Given \( n \) pts in 2D, build data structure to answer queries:
- axis-aligned
- given a rectangle \( q \), find pts inside \( q \).

Diff. version:
- report-one,
- report-all,
- count,
- sum of weights,
- \( \text{min weight} \leq \text{range min weight} \),
- \( \text{rank} \) - can consider other ranges
  e.g. disks, triangles,
  \( \ldots \)

1D:

Insert/delete: \( O(\log n) \) by BST

Space: \( O(n) \)
Query: \( O(\log n) \) for report-one, count
\( O(\log n + k) \) for report-all

2D? Special case: 3-sided rectangle
reduces to 1D range min query

\[
\Rightarrow O(n) \text{ space} \\
\text{query time } O(\log n + 1)
\]

for report-one

\[
2 \text{ find } \min \text{ search in } x
\]

for report-all: \(O(\log n + k)\) by repeatedly find min & recurse left & right