Problem: Search for integers in \([U]\)

- What we know: \(O(\log n)\) query/update, \(O(n)\) space
- \[O(1)\] query time,
  \(O(U)\) space, \(O(U)\) update time

Van Emde Boas '75: \(O(\log \log U)\) query time & update time (expected), \(O(n)\) space

**VEB Trees**

**Idea**: Multi-way D&C

- Divide \([U]\) into \(\sqrt{U}\) chunks of length \(\sqrt{U}\)
  - \(0\) \(\sqrt{U}\) \(\sqrt{U}\) \(\sqrt{U}\) \(\ldots\) \(\sqrt{U}\) \(U\)

- Store min & max for each chunk \(\leq\) \(\sqrt{n}\) size \(\sqrt{U}\)
- Recurse inside each chunk \(\leq\) excluding min & max
  - \(D = \{(\text{indices of all nonempty chunks}) \leq \sqrt{U}\}\)
  - Store \(D\) in dictionary/hashing
  - Recurse in \(D\).

**Query\((q)\)**:

- \(i = \text{index of chunk containing } q\)
  - \(O(1)\) time by int division
- \(O(1)\) time by hashing
  - If \(i \in D\), recurse inside chunk (also check min/max)
  - If \(i \notin D\),
If \( i \notin D \),
recurse in \( D \)
& return max of pred chunk

\[
Q(U) = Q(\sqrt{U}) + O(1)
\]

let \( U = 2^l \),
\[
\sqrt{U} = 2^\frac{l}{2}
\]

\[
Q'(l) = Q'(\frac{l}{2}) + O(1)
\]
\[
\Rightarrow O(\log l) = O(\log \log U)
\]

**Insert(\( x \)):**

\( i = \) index of chunk containing \( x \)

If \( i \in D \),
recurse inside chunk & update min/max

if \( i \notin D \),
insert \( i \) to \( D \),
set chunk's min/max to \( x \).

\[
I(U) = I(\sqrt{U}) + O(1)
\]

\[
\Rightarrow O(\log \log U)
\]

**Delete:** Similar

**Space:** \( O(\ n \log \log U ) \) space
but can be reduced to $O(n)$

**Trick:** divide into $O(n^b)$ blocks of $O(b)$ elems

Store min/max per block in UEB tree

$\Rightarrow$ space $O(n + \frac{n}{b} \log\log U) \Rightarrow O(n)$

Query/time update $O(\log\log U + b) \Rightarrow O(\log\log U)$

linear search

choose $b = \log\log U$

**Alternative version:**

![Diagram](image)

**Remark:**

Static:
Beame, Fich '02 $O\left(\frac{\log\log U}{\log\log\log U}\right)$ query time

but $O(n^{\log_2})$ space

matching lower bds
Rmk on Model of Computation: Word RAM

- RAM on w-bit word
- assume standard ops on w-bit words
  \[ \text{in } O(1) \text{ time} \]
  (add, mult, div, bitwise and/or)
- assume \( w \geq \log n \) (indices, ptrs fit in word)
- assume \( w \geq \log U \) (input numbers fit in word)

Next: Fusion Tree (Fredman, Willard’90)

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

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

\[ \{8, 10, 14, 15\} = \{0100, 0110, 1110, 1111\}\]

Idea - trie

\[\text{Compressed trie}\]