Sorting in $O(n)$ for $w \geq \lg^{2+\varepsilon} n$ ($\lg \lg n$)

[Andersson et al. 1998]

6.897
L13.2
Mar. 17, 2005

High-level idea:

- break each integer into $\lg^{\varepsilon} n$ equal-size chunks
- replace each chunk by $O(\lg n)$-bit index (order preserving)

$\Rightarrow$ n integers effectively $O(\lg^{1+\varepsilon} n)$ bits long

Tool: packed sorting

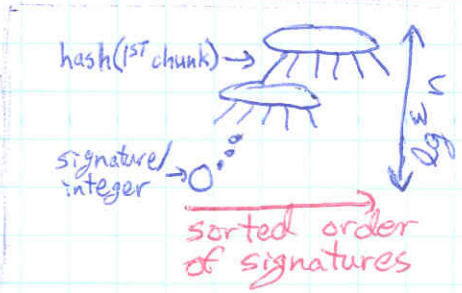
sort n b-bit integers in $O(n)$ time if $w = \Omega(b \lg n \lg\lg n)$

Signature sort:

- replace each chunk in an integer with $O(\lg n)$-bit hash $\rightarrow$ signature
- packed sort on signatures
- build height-$\lg^{\varepsilon} n$ tree of signatures
  - only store branching nodes (each node stores its "effective depth")
  - to add signature i to tree on $1, \ldots, i-1$:
    - find $|\text{lcp}(\text{sig. } i-1, \text{sig. } i)|$ in $O(1)$ (first 1 bit in XOR)
    - walk up rightmost path to node or edge of appropriate depth
      - add new child
      - subdivide edge
    - change length of walk-$O(1)$ to decrease in rightmost path length

  $\Rightarrow O(n)$ time

  (Cartesian tree construction algorithm of [Gabow, Bentley, Tarjan—STOC 1984])



- goal: convert to tree of integers with leaves in desired sorted order by re-ordering subtrees of each node (hash order $\rightarrow$ chunk order)
- attach node ID & current edge ID ($O(\lg n)$ bits) to chunk value ($w/\lg^{\varepsilon} n$ bits) along each edge
- recursively sort these $O(n)$ chunk values of $\sim w/\lg^{\varepsilon} n$ bits

  $\Rightarrow$ $O(1)$ recursions before packed sorting applies ($O(1/\varepsilon)$)
- ~~resort~~ actually sort by (node ID, hash value) (or resort via radix) $\Rightarrow$ information about a node is consecutive in array
- scan to find permutation of child subtrees for each node
- in-order traversal to extract leaf order