Radix sort
not comparison based
assumes array elements are 6-bit integers

Basic idea: slides
steps: • count (histogram)
    • accumulate
    • move

In general: split 6 bits into chunks

For example: \( b_i = b/m = k \) (constant radix \( k \))

\[ \begin{array}{c}
\text{A} \rightarrow \text{C} \rightarrow \text{D} \\
\text{count} \rightarrow \text{acc.} \rightarrow \text{move} \\
\text{size } 2^{b_0} = 2^k \\
\text{size } N
\end{array} \]

Analysis:
- count: \( O(N) \)
- acc.: \( O(2^k) \)
- move: \( O(N) \)
- iterations: \( O(mN) \)

Locality:
- Temporal: no
- Spatial: yes, dual 2^k "live" address regions

\( \Rightarrow \) can cause TLB misses
Translation Lookaside Buffer (TLB)

Virtual memory

Physical memory

Virtual address (32 bits)

Physical address (32 bits)

TLB: stores (v,p) pairs

Intuitively: The size of the TLB determines how many different memory regions a program can operate on simultaneously and efficiently.

Core 2 Duo:

- L1 virtually addressed
- L2 physically addressed

$$|\text{TLB}| = 32 = 2^5$$

CC-Radix Sort (CC = Cache Conscious)

Basic idea:

- Sort by MSB first -> temporal locality
- Choose radix $< c_d$, $c_d \approx |\text{TLB}|$
- Repeat until array fits into L2 cache
- Then use standard radix sort
Adaptive Sorting

8 Aides

Selected

Nawaz

count

2

acc

Unsure

first digit = 00000

10000 = 00000