Tester for Nearly-Sortedness and its Applications in Databases

Arie Matsliah

Sagi Ben-Moshe
Eldar Fischer
Yaron Kanza

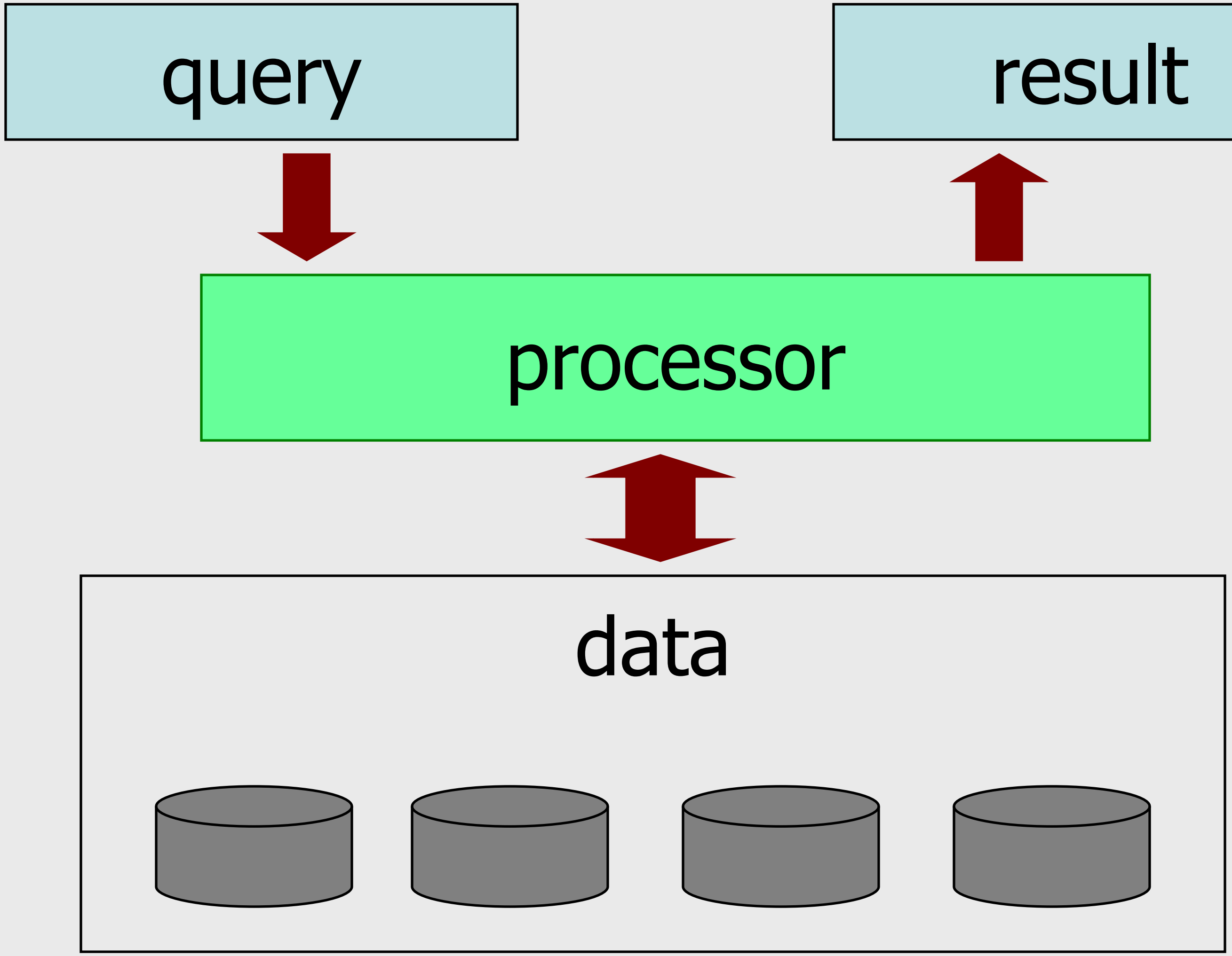
Outline

- New definition: "nearly-sorted"

- Tolerant tests for the property of being nearly-sorted

- Applications in Databases

Database

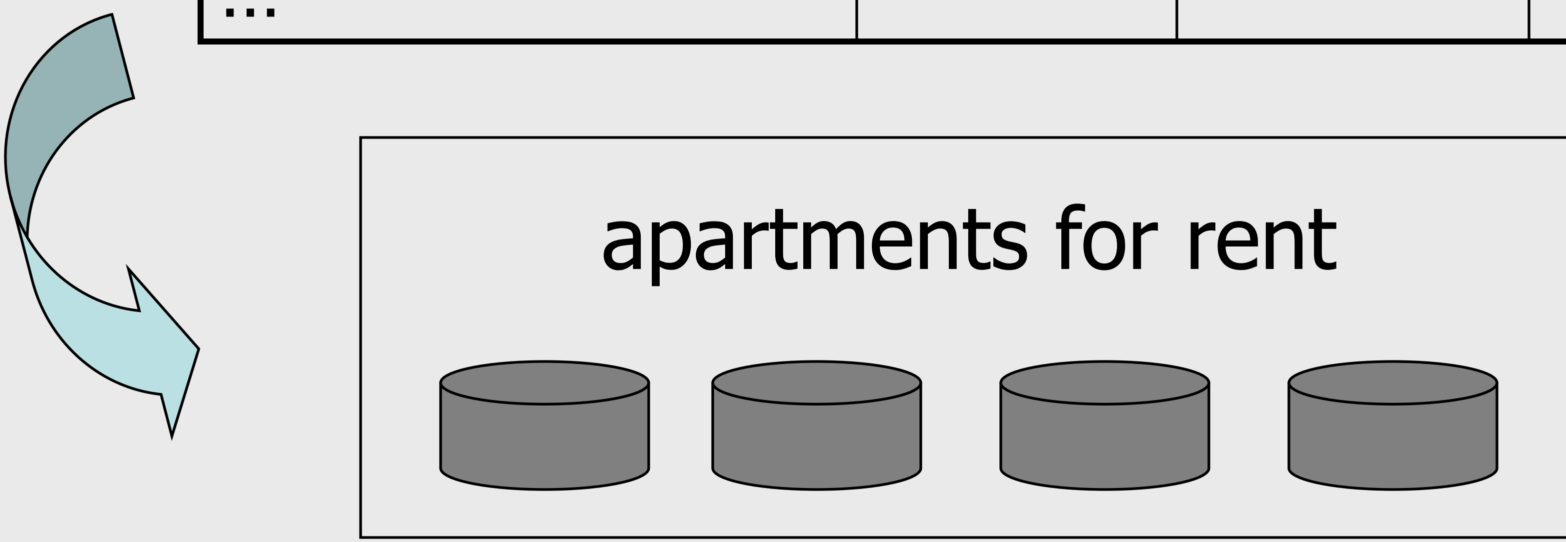


Example (data)

attributes

records

Address	Fee ($)	Size (m^2)	…
Main rd 29, …	500	70	…
First ave 3, …	850	120	…
…			
…			



Example (query)

select all apts. in NY,
having size between 70 and 80 m^2,
present sorted by monthly fee

query

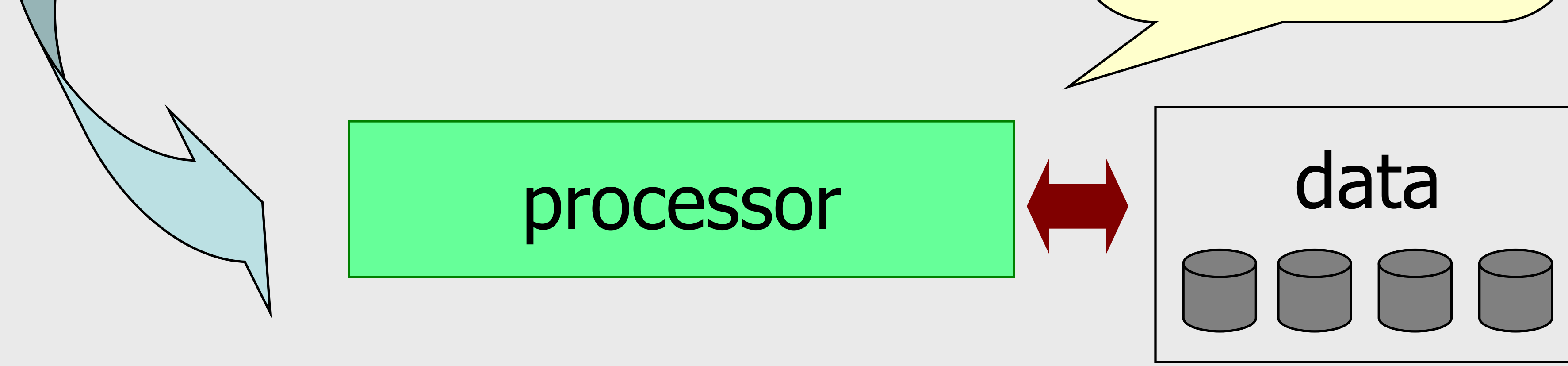
Example (processor)

parse query
…
select apts. of right size
…
sort result according to fee
…
return result

Random access is expensive

Sequential pass is cheap

In-memory computation is cheapest



Facts

- Some queries/operations can be processed more efficiently if the data is ordered (sorted acd. to some attribute)

- Additional examples: natural join, intersection, union, except, ...

- However, in many cases processor cannot assume the data is ordered

Observation (from experiments)

- Monitoring the "sort" function of a DB-management system (PostgreSQL)

- In many cases, even before sorting the data is "nearly sorted"

- Idea:
  1. test whether the data is "nearly sorted"
  2. if it is – use sorting algorithm that is tailored for nearly-sorted data

Ingredients

1. property tester for the property of being nearly-sorted

    * few queries = few random access to data
    * must be tolerant

2. efficient sorting algorithm that works if the data is nearly-sorted

    * no random access, few sequential passes
    * always correct (discovers failure)

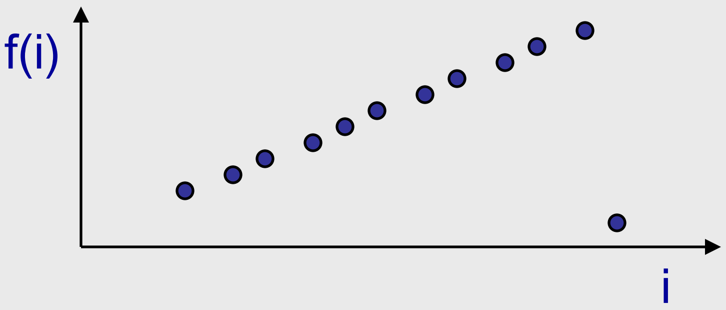
Definition: Nearly-Sorted

- $f:[n] \rightarrow R$
- R – attribute values, total order ($<, \leq, >, \geq$)

- f is sorted if for all $i<j$, $f(i) \leq f(j)$
- f is k-sorted if for all i,j: $i \leq j-k \Rightarrow f(i) \leq f(j)$
- 1-sorted $\Leftrightarrow$ sorted
- f is ε-close to being sorted if
  for some $E \subseteq [n]$, $|E| \leq \varepsilon n$: $f|_{[n]\setminus E}$ is sorted
- f is (ε,k)-nearly-sorted if
  for some $E \subseteq [n]$, $|E| \leq \varepsilon n$: $f|_{[n]\setminus E}$ is k-sorted

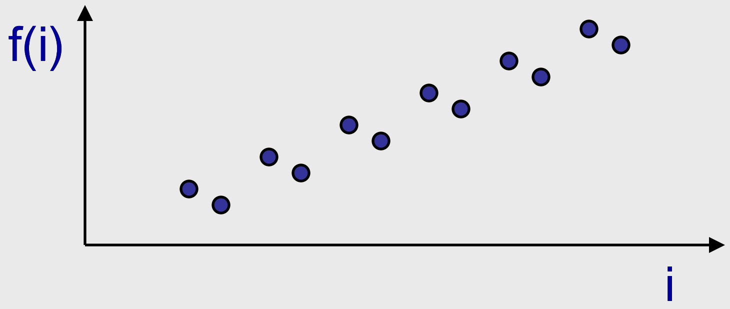
Example 1

- $1/n$-close ($\varepsilon=1/n$), n-sorted ($k=n$)



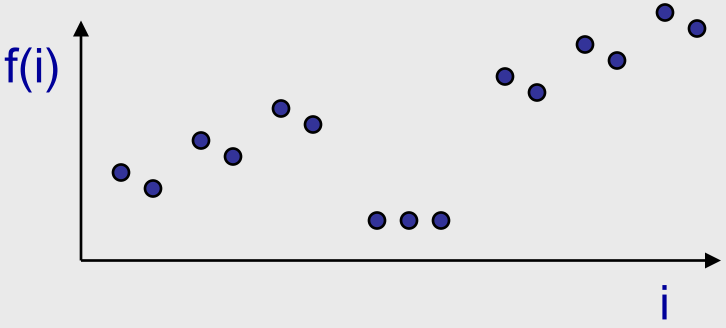
Example 2

- 1/2-close (ε=1/2), 2-sorted (k=2)



Example 3

- (1/5,2)-nearly-sorted



Next

- (Tolerant) Test for nearly-sortedness
- Algorithm for sorting nearly-sorted functions
- Experiments

Testing Nearly-Sortedness

$([\varepsilon_1,\varepsilon_2],[k_1,k_2])$-test:

- ACCEPT w.p. 2/3 if (ε_1,k_1)-nearly sorted
- REJECT w.p. 2/3 if not (ε_2,k_2)-nearly sorted

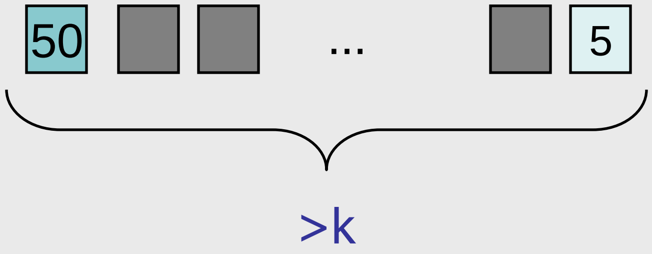
- $([0,\varepsilon],[1,1])$-test = tester for monotonicity
  #queries = $O(\log(n)/\varepsilon)$
  [BRW, DGLRRS, EKKRRV, GGLRS, FLNRRS, HK]

- $([\varepsilon,c\varepsilon],[1,1])$-test = tolerant tester for monotonicity
  #queries = $\tilde{O}(\log(n)/\varepsilon)$
  [PRR,ACCL]

- $([\varepsilon,c\varepsilon],[k,ck])$-test = tolerant test for nearly-sortedness
  #queries = $\tilde{O}(\log(n)/\varepsilon)$
  [this work]

k-Violations and (δ,k)-Active Indices

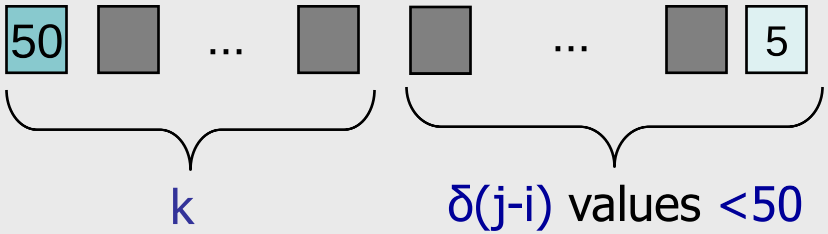
- (i,j) is a k-violation if $i \le j-k$ and $f(i)>f(j)$



k-Violations and (δ,k)-Active Indices

- (i,j) is a k-violation if $i \le j-k$ and $f(i) > f(j)$

- i is (δ,k)-active if for $\ge \delta(j-i)$ indices $h \in [i,j]$,
  (i,h) is a k-violation

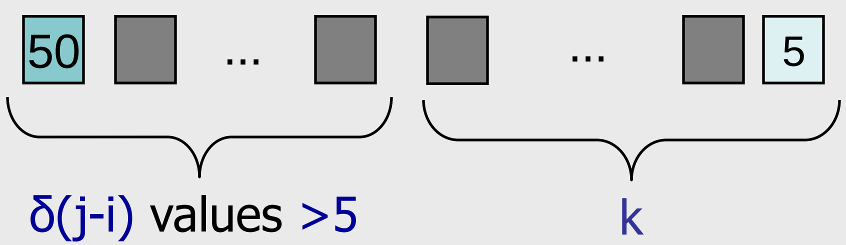


k-Violations and (δ,k)-Active Indices

- (i,j) is a k-violation if $i \leq j-k$ and $f(i)>f(j)$

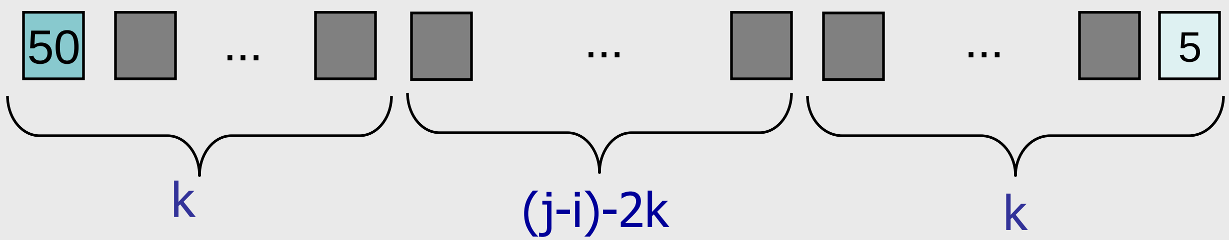
- i is (δ,k)-active if for $\geq \delta(j-i)$ indices $h \in [i,j]$,
  (i,h) is a k-violation

- j is (δ,k)-active if for $\geq \delta(j-i)$ indices $h \in [i,j]$,
  (h,j) is a k-violation



k-Violations and (δ,k)-Active Indices

- either i or j must be (½-k/(j-i),k)-active



➔ if f is not (ε,k)-nearly sorted,

(½-k/(j-i),k)-actives ≥ εn

Towards tolerant testing based on [ACCL]

Lemma

- if f is (ε,k)-nearly sorted then

  $\#\ (1/4,k)$-actives $\leq 5\varepsilon n$

- if f is not $(6\varepsilon,6k)$-nearly sorted then

  $\#\ (1/3,k)$-actives $\geq 6\varepsilon n$

([ε,cε],[k,ck])-test

```
counter=0
repeat T=O(1/ε) times:
        pick i∈[n]
        If i is 1/3-active then counter++
if (counter/T >5.5ε)
        REJECT
else
        ACCEPT
```

Problem: how to check if i is 1/3-active?

Activity-testing algorithm

input: i,δ

- if i is $(1/3,k)$-active,
  output YES w.p. $\geq 1-\delta$
- if i is not $(1/4,k)$-active,
  output NO w.p. $\geq 1-\delta$

query complexity: $\tilde{O}(\log(1/\delta)\log(n))$

works by approximating the number of violations (by sampling) within neighborhoods of increasing size

$([\varepsilon,6\varepsilon],[k,6k])$-tolerant test

```
count=0
repeat T=O(1/ε) times:
        pick i∈[n]
        if AT(i,1/T) = YES then count++
if (count/T >5.5ε)
        REJECT
else
        ACCEPT
```

query complexity: $\tilde{O}(\log(n)/\varepsilon)$

- if f is (ε,k)-nearly sorted
  ➔ fraction of $(1/4,k)$-actives $\leq 5\varepsilon$
  "➔" ACCEPT w.p. $\geq 2/3$

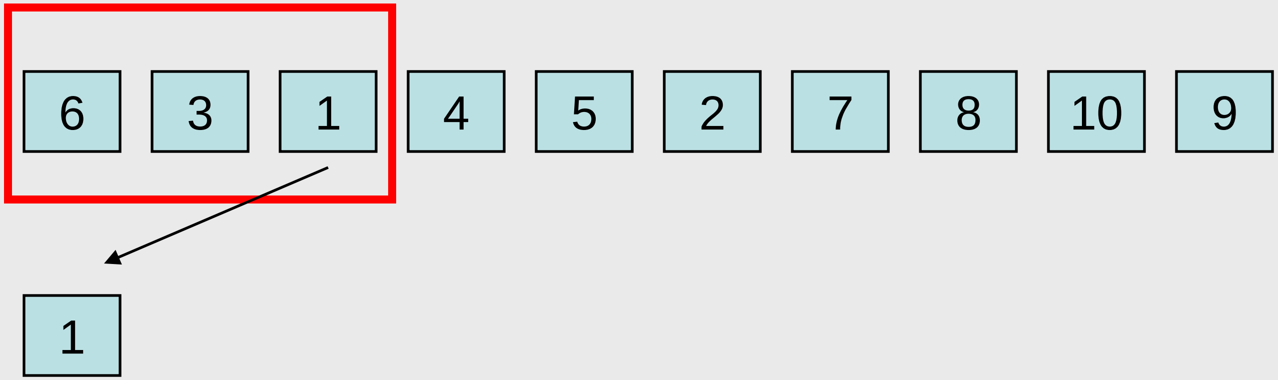
- if f is not $(6\varepsilon,6k)$-nearly sorted
  ➔ fraction of $(1/3,k)$-actives $\geq 6\varepsilon$
  "➔" REJECT w.p. $\geq 2/3$

Sorting Nearly-Sorted relations

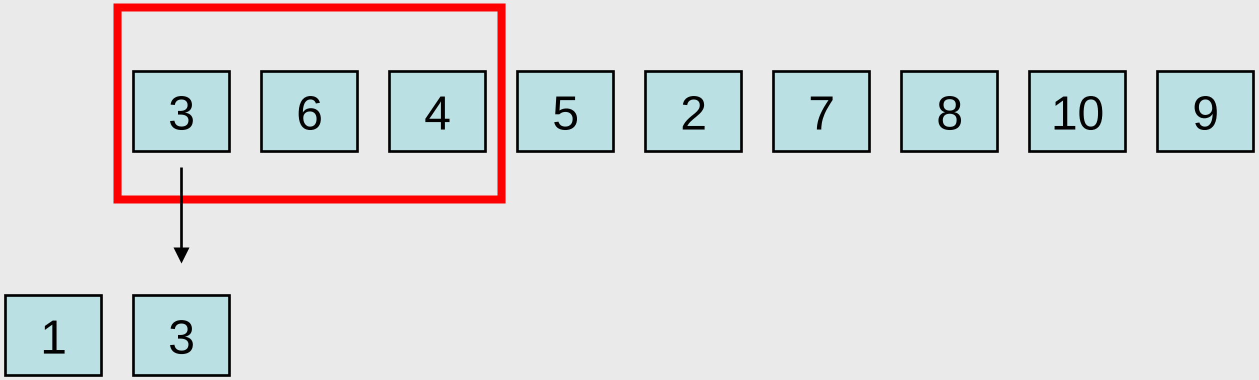
- Use the Replacement-Selection algorithm

- Thm: if f is (ε,k)-nearly-sorted, then RS with $M=\varepsilon n+k$, sorts f in two passes

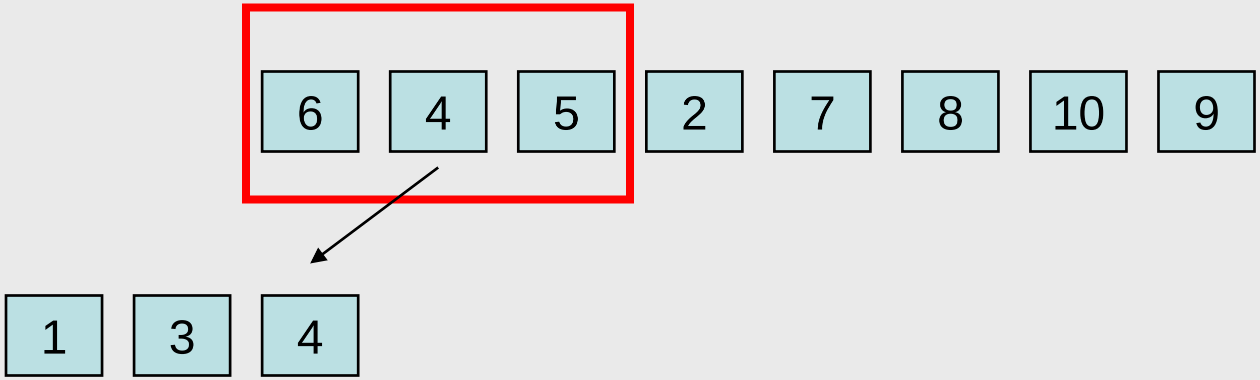
Replacement-Selection



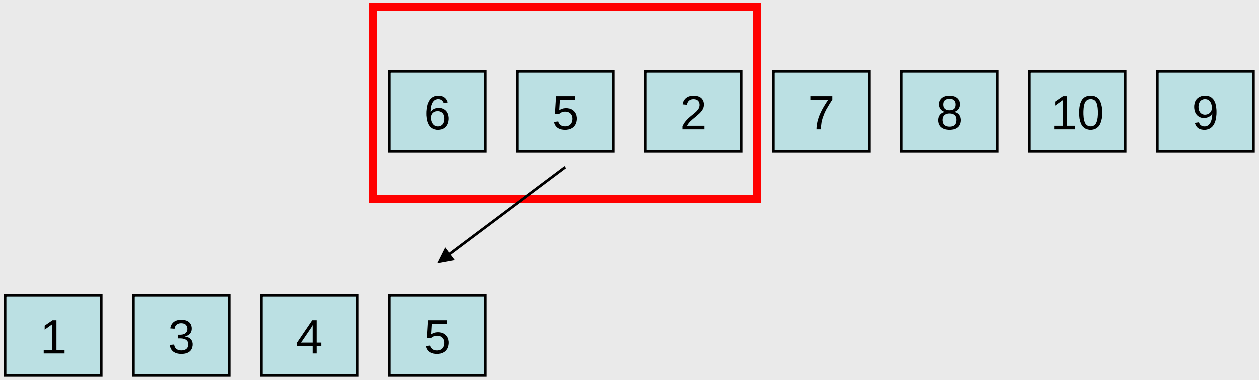
Replacement-Selection



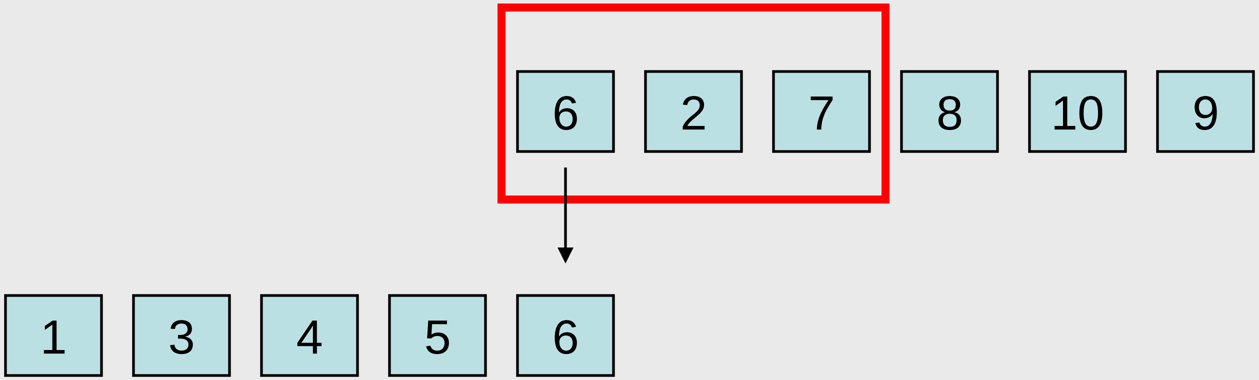
Replacement-Selection



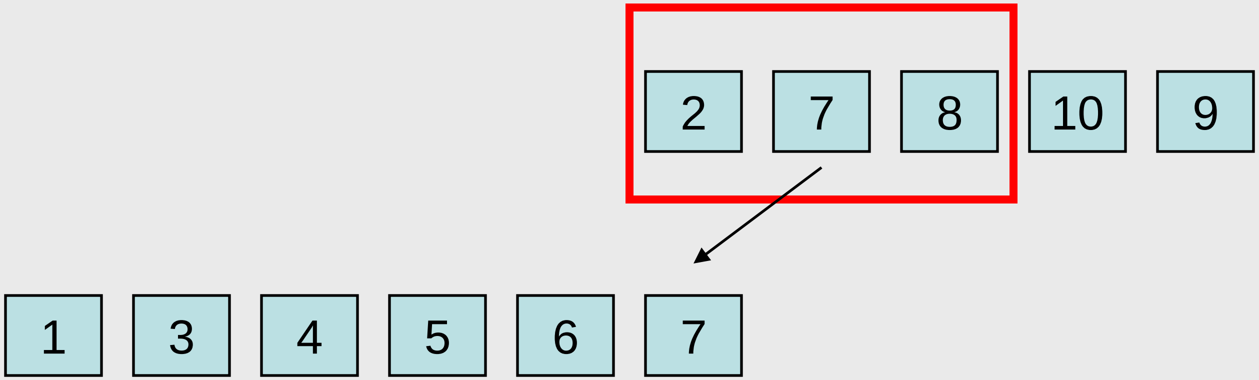
Replacement-Selection



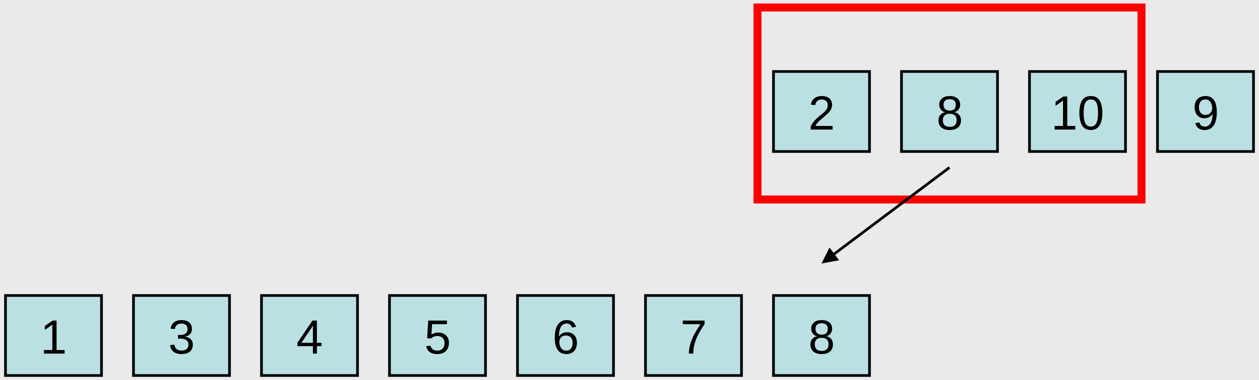
Replacement-Selection



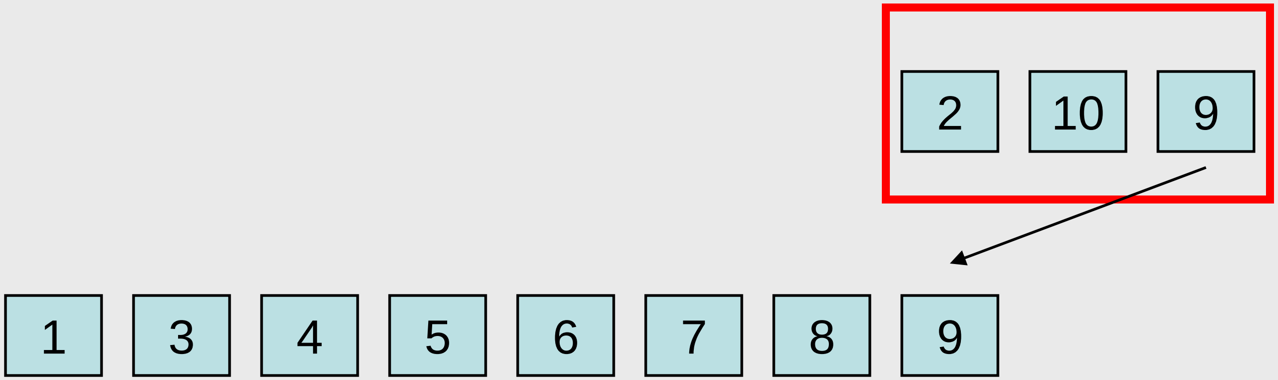
Replacement-Selection



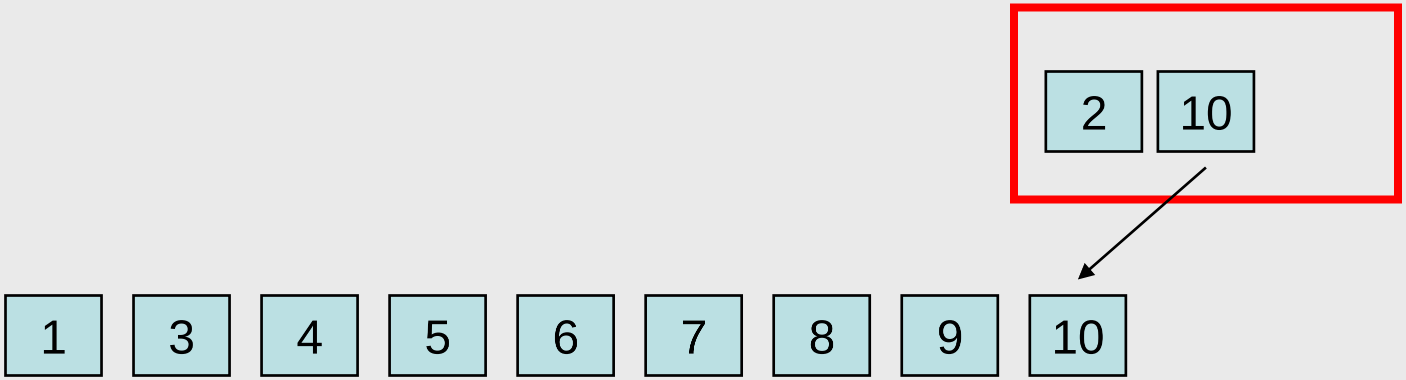
Replacement-Selection



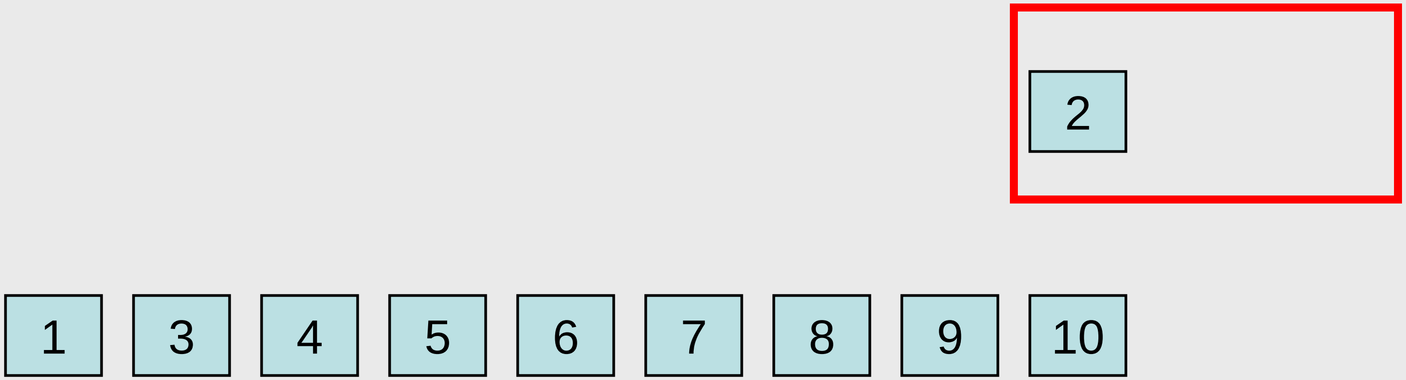
Replacement-Selection



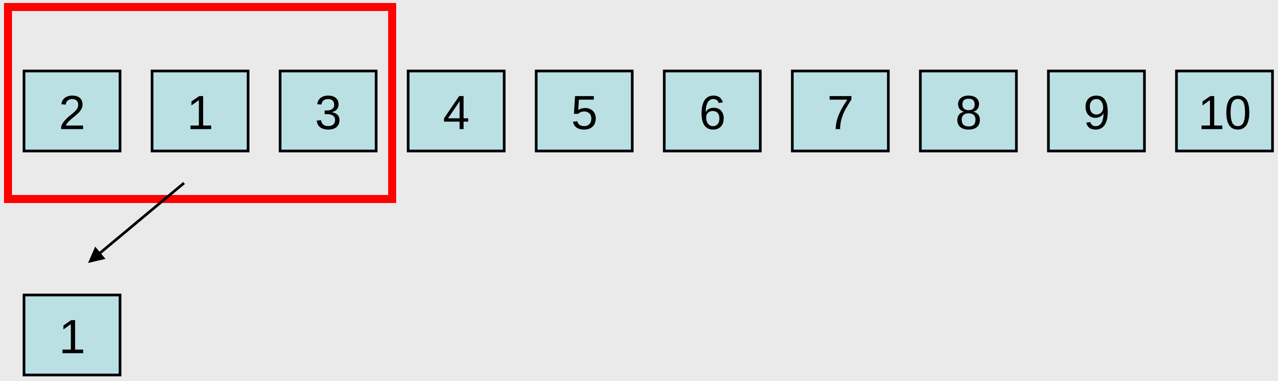
Replacement-Selection



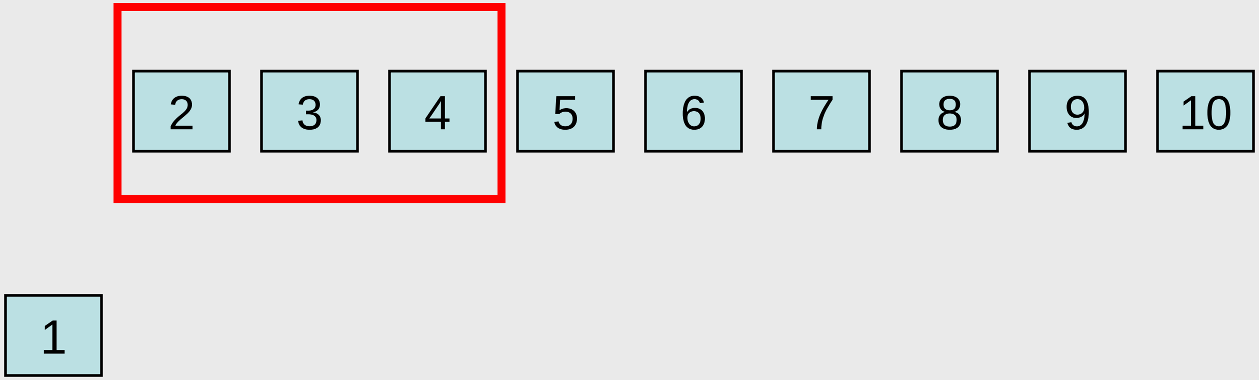
Replacement-Selection



Replacement-Selection



Replacement-Selection



If #marked ≤ M, the data is sorted after two passes

Sorting Nearly-Sorted relations

Lemma: #marked $\leq M$ $(=\varepsilon n+k)$

Proof:

let $E_1,\ldots,E_t \subseteq [n]$ be "bad" subsets of size $\leq \varepsilon n$

let D be their intersection

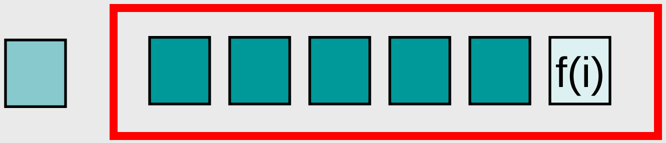
clearly $D \leq \varepsilon n$

Claim: If i is marked, then $i \in D$ ➔ #marked $\leq \varepsilon n$

Proof:

for $i \leq \varepsilon n+k$, no index is marked

let $i > \varepsilon n+k$ be marked, assume i not in D ➔ i not in E_h for some h



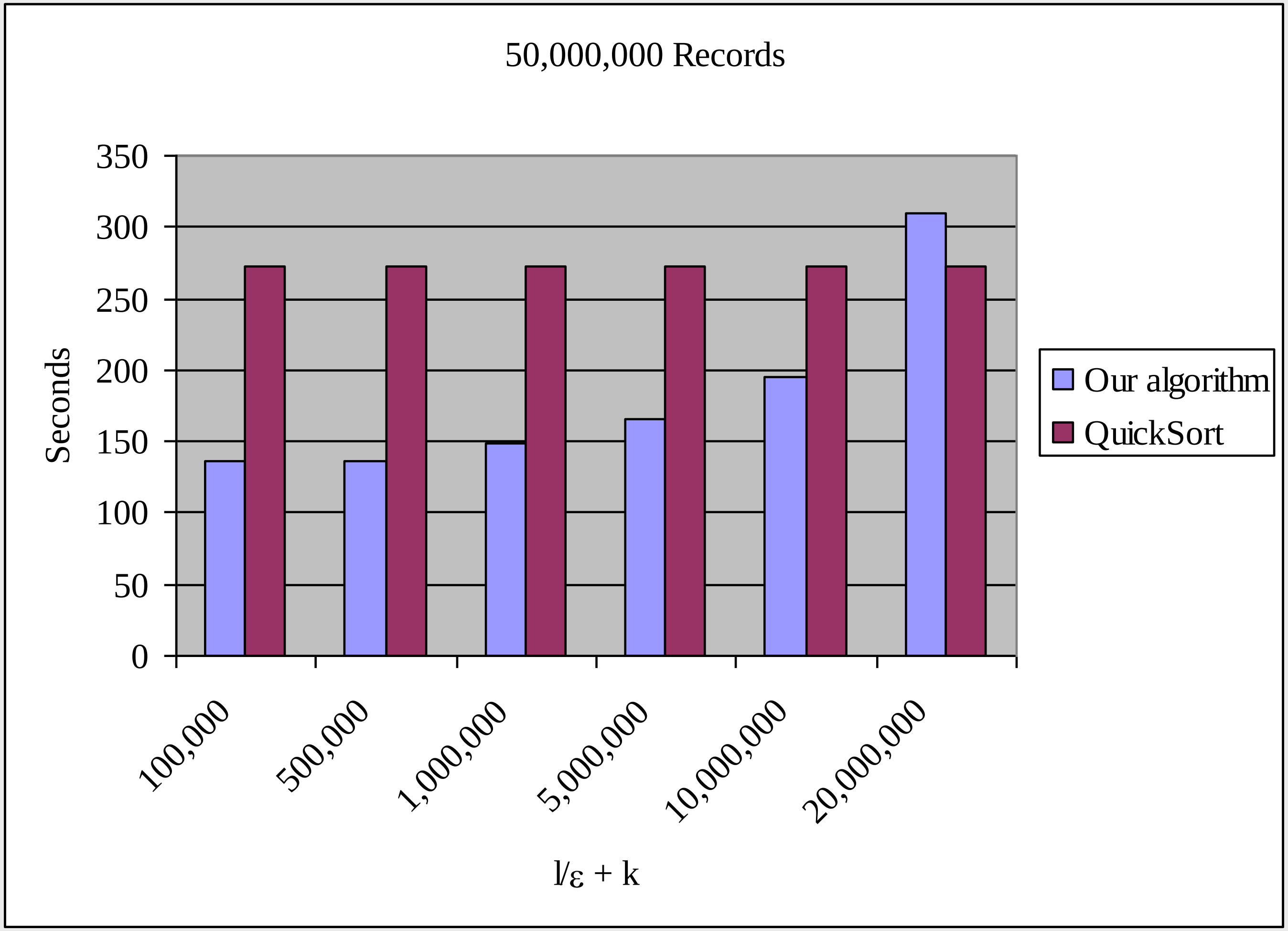
$>\varepsilon n$ k-violations

➔ $E_h > \varepsilon n$ (contradiction)

Experiments

- monitoring the "sort" function of PostgreSQL
- data was $(1/\sqrt{n}, \sqrt{n})$-nearly sorted in most cases
- testing with parameters compatible with currently typical memory size is faster than making one pass
- in-memory sorting $(6/\sqrt{n}, 6\sqrt{n}])$-nearly sorted data with RS is >2 times faster than standard quicksort
- more elaborate tests pending…

Experiments



Thank you