Optimal dislocation with persistent errors in subquadratic time

Demo Talk for Student Seminar
("bad" version)

ETH Zurich

Notation

- $S^* = \{1, 2, \dots, N\}$ = sorted sequence
- S = any sequence
- $rank(x, S) := |\{y \in S|\ y < x\}|$.
- $disloc(x, S) = |x - rank(x, S)|$

Notation

- $S^* = \{1, 2, \dots, N\}$ = sorted sequence
- S = any sequence
- $rank(x, S) := |\{y \in S| \ y < x\}|$.
- $disloc(x, S) = |x - rank(x, S)|$

Definition 1 An algorithm has maximum dislocation $d = d(N)$ (with high probability) if, for any input sequence of N elements, it returns a sequence S such that $disloc(x, S) \leq d$ for all elements x with probability at least $1 - 1/poly(N)$.

Prior Work

- Braverman and Mossel (2008): max dislocation $O(\log N)$ in time $O(N3 + C)$, where $p < 1/2$ is the comparison error probability.
- Klein et al. (2011): max dislocation $O(\log N)$ in time $O(N^2)$, where $p < 1/16$ is the comparison error probability.
- Geissmann et al (2016): no (possibly randomized) algorithm can achieve maximum dislocation o(log n) with high probability.

Main Result

Theorem Recursive Window Sort returns a sequence with maximum dislocation $\kappa \log N$ with probability at least $1 - \frac{1}{N^2}$. Moreover, its running time is $\widetilde{O}(N^{\frac{3}{2}})$ and the expected total dislocation of the returned sequence is $O(n)$.

Warm Up

- Start with a *random permutation* S of the input sequence and split this sequence S into β blocks of the same size.
- Run $\textsc{Window Sort}$ on each block B_i to obtain a sequence S_i.
- Combine all the sequences S_i together into a sequence S' as follows: The first element in each S_i will be placed (in arbitrary order) in one of the first β positions of S', the second element in each S_i will be placed in a position between $\beta + 1$ and 2β in S', and so on.
- Run $\textsc{Window Sort}$ on this new sequence S'.

The Algorithm (1/3)

Algorithm NewWindowSort
(on N distinct elements)
1) Let S be a random permutation the N input elements
2) Run RecStep on S (with initial depth $d = 0$)
3) Return the resulting sequence

The Algorithm (2/3)

Algorithm RECSTEP
(on a sequence S of n_d distinct elements at depth d)
1) If $d = h$ then
  a) Run WINDOWSORT on $S' = S$ with window size n_d
  b) Return the resulting sequence
2) Else
  a) Partition S into $b_d := \frac{n_d}{\beta_d}$ blocks $B_1, B_2, \ldots, B_{b_d}$ each containing β_d elements
  b) For each *block* B_i
    i) Run RECSTEP on B_i with depth $d+1$ to obtain
      $B'_i = \langle b'_{i,1}, b'_{i,2}, \ldots, b'_{i,\beta_d} \rangle$
  c) For each $j = 1, 2, \ldots, \beta_d$ do
      $B''_j = \langle b'_{1,j}, b'_{2,j}, \ldots, b'_{b_d,j} \rangle$
  d) Let $S' = \langle s'_1, s'_2, \ldots, s'_{n_d} \rangle = \langle B''_1, B''_2, \ldots, B''_{\beta_d} \rangle$
  e) Run WINDOWSORT on S' with window size W_d
  f) Return the resulting sequence

The Algorithm (3/3)

To optimize the running time, we set the parameters as follows:

$$\beta_d \triangleq n_d^{\,1-\frac{1}{2^{h-d+1}-1}} \qquad \text{and} \qquad W_d \triangleq 4\kappa \frac{n_d}{\sqrt{\beta_d}} \log N.$$

Analysis

To optimize the running time, we set the parameters as follows:

$$\beta_d \triangleq n_d^{1-\frac{1}{2^{h-d+1}-1}} \qquad \text{and} \qquad W_d \triangleq 4\kappa \frac{n_d}{\sqrt{\beta_d}} \log N.$$

Lemma The overall running time of $\textsc{NewWindowSort}$ is $\widetilde{O}(N^{\frac{3}{2}})$.

proof Running time at depth $d = h - i$ is

$$T_i \leq c(i+1) n_d^{1+2^i/(2^{i+1}-1)} \log N$$

for some c constant.

Analysis

Base case: $i = 0 \quad (d = h)$

$$T_0 \leq cn_d^2 \leq cn_d^{1+2^0/(2^1-1)}$$

by the $\textsc{WindowSort}$ running time

Inductive step: true for $i - 1 \quad$ (depth $d + 1$)

$$T_{i-1} \leq c\, i\, n_{d+1}^{1+\frac{2^{i-1}}{(2^i-1)}} \log N = ci\beta_d^{1+\frac{2^{i-1}}{2^i-1}} \log N$$

Depth d make $b_d = \frac{n_d}{\beta_d}$ calls at level $d + 1$

and a call to $\textsc{WindowSort}$ on n_d elements with initial window size $W_d = 4\kappa \frac{n_d}{\sqrt{\beta_d}} \log N$

Analysis

$$T_i \le \frac{n_d}{\beta_d} \cdot ci\beta_d^{1+\frac{2^{i-1}}{2^i-1}} \log N + 4\kappa c' \frac{n_d^2}{\sqrt{\beta_d}} \log N =$$

$$c\left(in_d\beta_d^{\frac{2^{i-1}}{2^i-1}} + n_d^{\frac{3}{2}+\frac{1}{2^{i+2}-2}} \right) \log N =$$

$$c\left(in_d^{1+\frac{2^i}{2^{i+1}-1}} + n_d^{1+\frac{2^i}{2^{i+1}-1}} \right) \log N = c(i+1)n_d^{1+\frac{2^i}{2^{i+1}-1}} \cdot \log N$$

Analysis

$$T_i \le \frac{n_d}{\beta_d} \cdot ci\beta_d^{1+\frac{2^i-1}{2^i-1}} \log N + 4\kappa c' \frac{n_d^2}{\sqrt{\beta_d}} \log N =$$

$$c\left(in_d\beta_d^{\frac{2^i-1}{2^i-1}} + n_d^{\frac{3}{2}+\frac{1}{2^{i+2}-2}} \right) \log N =$$

$$c\left(in_d^{1+\frac{2^i}{2^{i+1}-1}} + n_d^{1+\frac{2^i}{2^{i+1}-1}} \right) \log N = c(i+1)n_d^{1+\frac{2^i}{2^{i+1}-1}} \cdot \log N$$

Running time at depth $d = h - i$ is

$$T_i \le c(i+1)n_d^{1+2^i/(2^{i+1}-1)} \log N$$

for some c constant.

Analysis

For $i = h = \log N$ (depth $d = 0$)

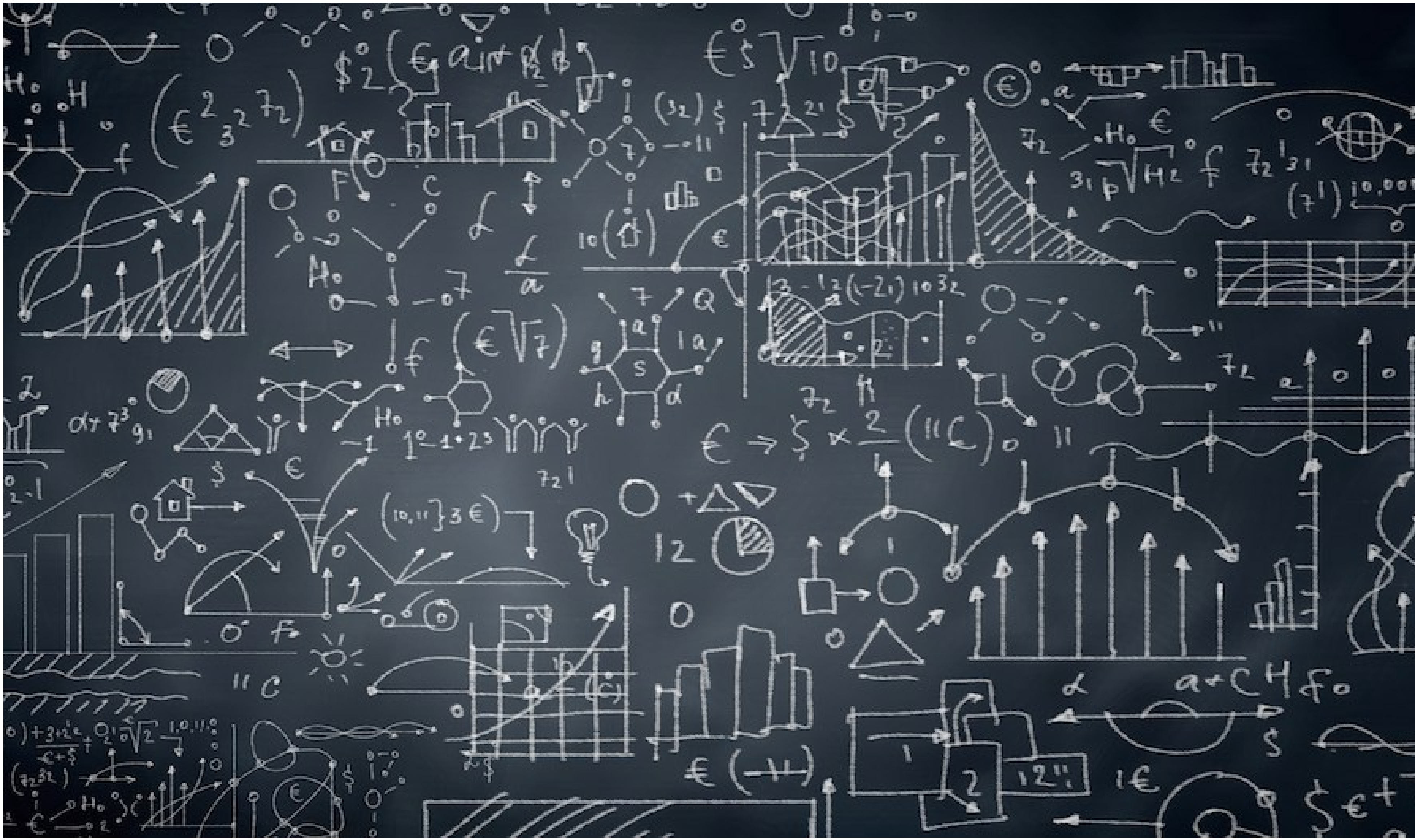
$$T_h \leq c n_0^{1+N/(2^{i+1}-1)} \log N$$

Running time at depth $d = h - i$ is

$$T_i \leq c(i+1) n_d^{1+2^i/(2^{i+1}-1)} \log N$$

for some c constant.

Is It Clear?



Thank you