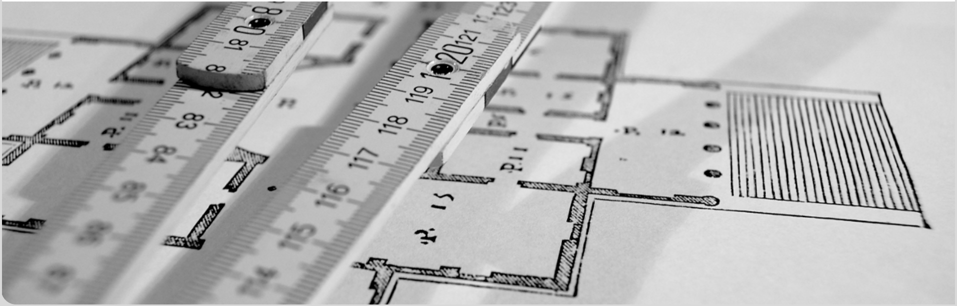
Parallel Algorithm for Closest Pair Problem

Ge Wu | July 25, 2013

INSTITUTE FOR THEORETICAL INFORMATICS



Problem Description

Closest Pair Problem

- Given n **different unordered** points $P = \{p_1, p_2, ..., p_n\}$ in **unit square**

$$p_i = (x_i, y_i) \in (0,1) \times (0,1) \subset \mathbb{R}^2$$

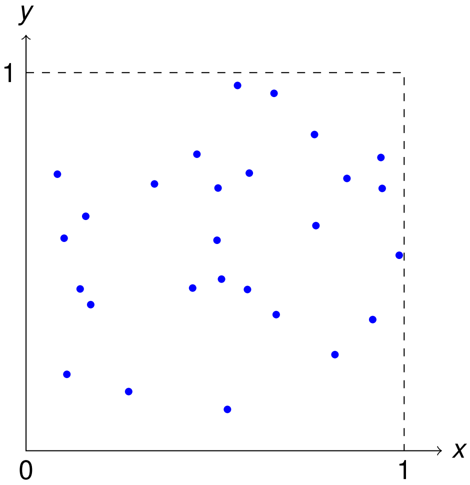
- Find the shortest euclidean distance between any two points

Background

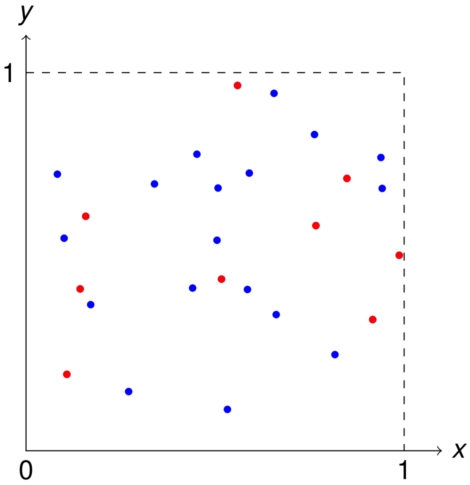
- $O(n \log n)$ lower bound in comparison tree model
- **Bentley and Shamos 1976**
  $O(n \log n)$ algorithm using divide and conquer
- **Rabin 1976**
  $O(n)$ randomized algorithm with $O(1)$ floor function
- **Fortune and Hopcroft 1978**
  $O(n \log \log n)$ deterministic algorithm with $O(1)$ floor function
- **Khuller and Matias 1995**
  Another O(n) randomized algorithm

Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute

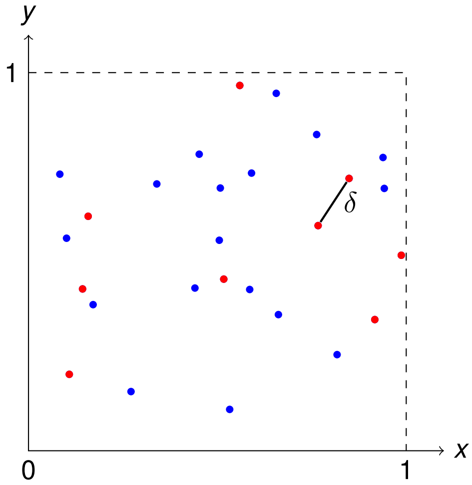


- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



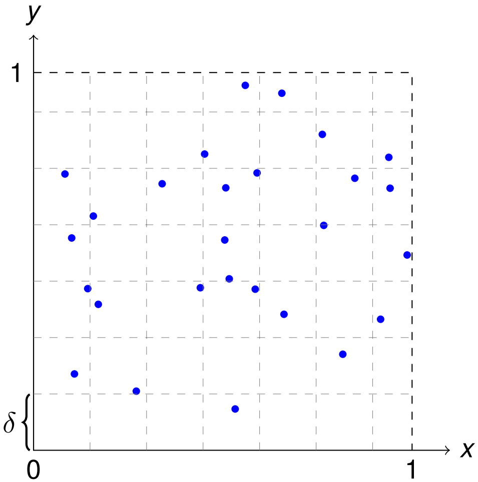
Algorithm

- Sample ($\boldsymbol{n^{\frac{2}{3}}}$ Points from P)
- Partition
- Compute



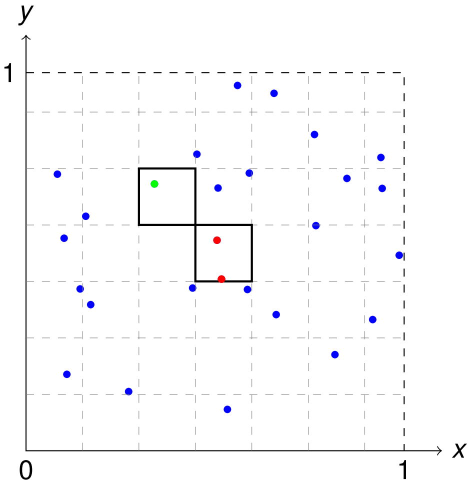
Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



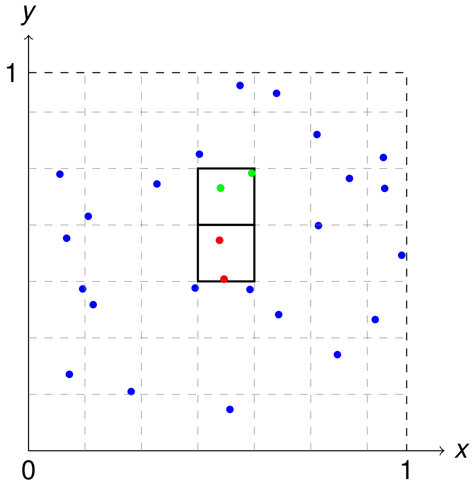
Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



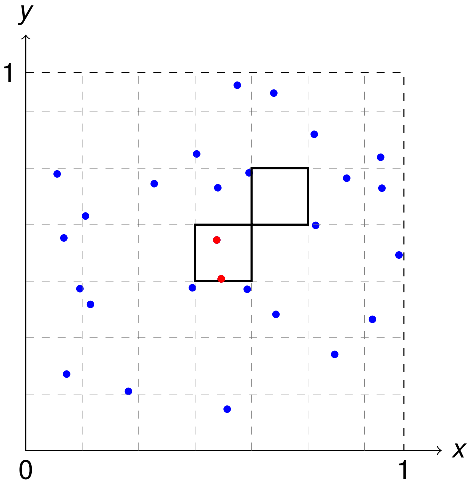
Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



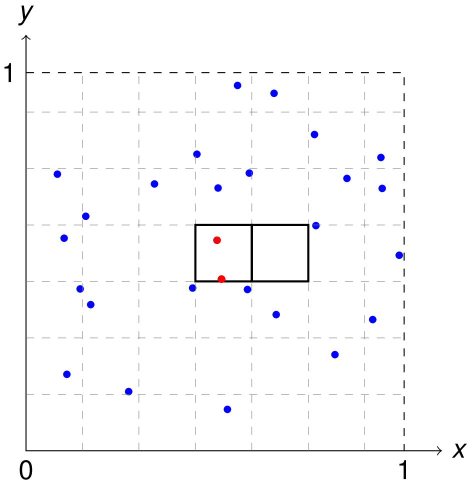
Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



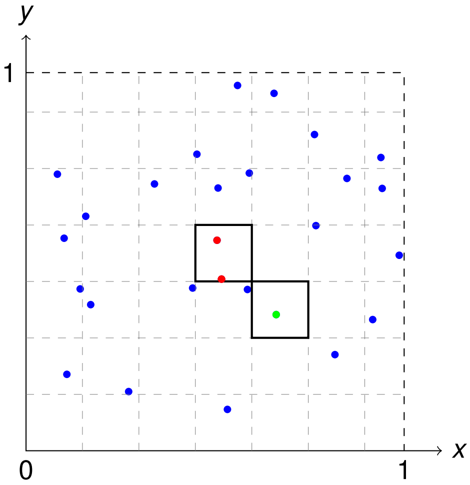
Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



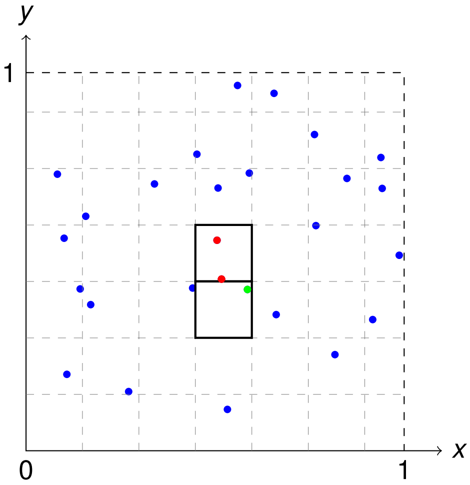
Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



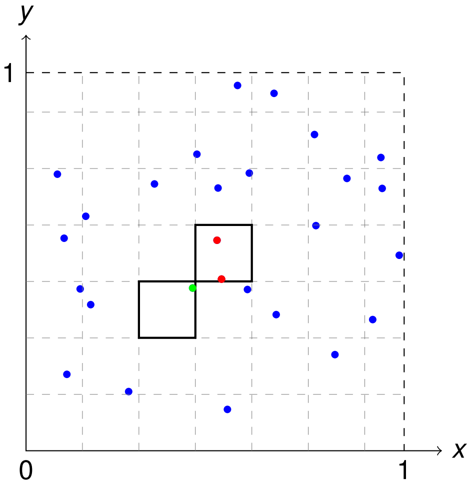
Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



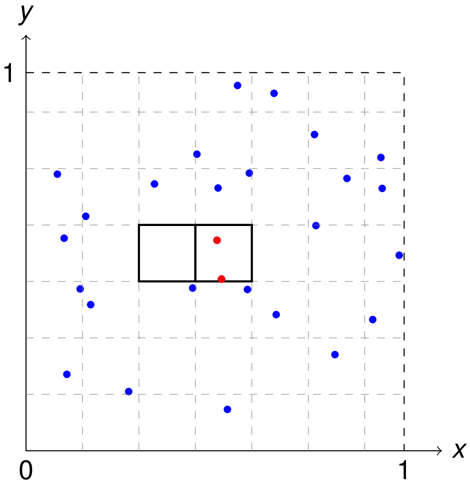
Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



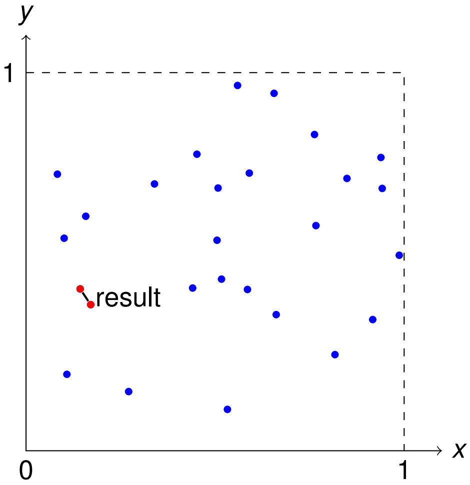
Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



Algorithm

- Sample ($n^{\frac{2}{3}}$ Points from P)
- Partition
- Compute



Sample

- Compute all pair of distances: $O((n^{\frac{2}{3}})^2) = O(n^{\frac{4}{3}})$
- Divide & Conquer: $O(n^{\frac{2}{3}} \log n^{\frac{2}{3}}) = O(n^{\frac{2}{3}} \log n) \subset O(n)$
  More samples possible: $O(\frac{n}{\log n} \log \frac{n}{\log n}) \subset O(n)$
- Other approach?

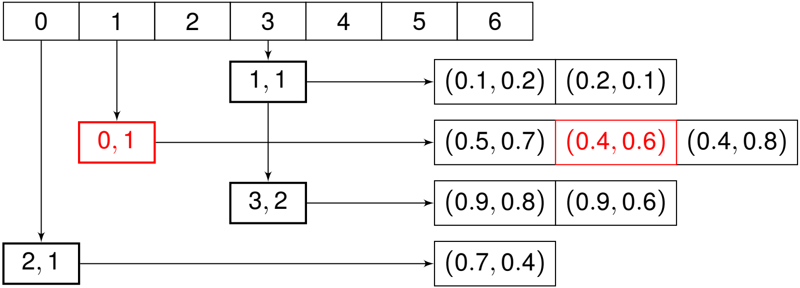
Partition

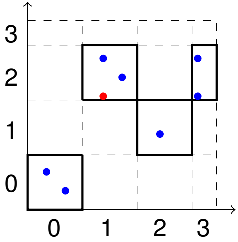
- Map points to the cells
- Divide the coordinates by cell length and **truncate** them to integer
- Partition $G = \{G_1, G_2, ..., G_k\}$ with:

$$\bigcup_{i=1}^{k} G_i = P, G_i \cap G_j = \emptyset, k <= n$$

- $O(n)$ with hashing

Partition





$$\boldsymbol{H}(cell_{x\_idx}, cell_{y\_idx}) = (cell_{x\_idx} + 2 \cdot cell_{y\_idx}) \mod 7$$

Compute

Theorem

*Let G_i with $1 \leq i \leq k$ be the **i-th** non-empty cell and c, d some positive constants, then*

$$Prob\left(\sum_{i=1}^{k} |G_i|^2 \leq c \cdot n\right) \geq 1 - \frac{1}{2^{n^d}}$$

Runtime for computing all pairs distances

$$O(\sum_{i=1}^{k} |G_i| \cdot |G_i \cup neighbor(G_i)|) = O(\sum_{i=1}^{k} |G_i|^2) = O(n)$$

Alternative for Sampling

Recursively calculate the closest distance on samples $S = \{s_1, ..., s_{n^{\frac{2}{3}}}\}$

- Sample another $n^{\frac{4}{9}}$ points S' from S
- Get closest distance on S' by calculating all pair distances: $O(n^{\frac{8}{9}})$
- Partition and compute the closest distance on S

Pseudocode

0 Cell set $H = \emptyset$, $result = \infty$

1 Choose set S of $n^{\frac{2}{3}}$ samples — $O(n^{\frac{2}{3}})$
  $\delta = min_{x,y \in S, x \neq y}(dist(x, y))$ — $O(n^{\frac{8}{9}})$

2 **for** $i = 1$ **to** n **do** — $O(n)$
  $c = cell(p_i)$
  if $not\ H.findCell(c)$ **then**
    $H.addCell(\{c, idx = \{c.x, c.y\}\}$
  $H.findCell(c).addPoint(p_i)$

3 **foreach** c_1 **in** H **do** — $O(n)$
  foreach c_2 **in** $\{c_1\} \cup neighbor(c_1)$ **do**
    foreach (p_i, p_j) **in** $c_1 \times c_2$
      $result = min(result, dist(p_i, p_j))$

Runtime: $O(n)$ with high probability, worst case: $O(n^2)$

Parallelization

- PRAM, CREW model

procedure SAMPLE(**in** P, id, **out** S)
  $total \leftarrow 0$
  for $i \leftarrow \lceil \frac{n}{\#PE} \rceil \cdot id$ **to** $min(\lceil \frac{n}{\#PE} \rceil \cdot (id+1) - 1, n-1)$ **do** $\triangleright\ O(\frac{n}{p})$
    $chosen[i] \leftarrow (\textbf{Random}([0..1]) < n^{\frac{2}{3}}/n)$
    $total \leftarrow total + chosen[i]$
  end for
  $pos@id \leftarrow \textbf{PrefixSum}(total@id)$ $\triangleright\ O(\log p)$
  for $i \leftarrow L \cdot id$ **to** $min(L \cdot (id+1) - 1, n-1)$ **do** $\triangleright\ O(\frac{n}{p})$
    if $chosen[i]$ **then**
      $S[--pos] \leftarrow P[i]$
    end if
  end for
end procedure

Parallelization

- Concurrent hash table (possibly with variable size)
- Concurrent read, exclusive write (block the chain)
- At most n add operation, constant blocking time for each chain, if elements evenly distributed

procedure $\textsc{Partition}$(**in** $P, \delta,$ **out** H)
  $H \leftarrow \emptyset$
  for $i \leftarrow \lceil \frac{n}{\#PE} \rceil \cdot id$ **to** $min(\lceil \frac{n}{\#PE} \rceil \cdot (id + 1) - 1, n - 1)$ **do** $\triangleright$ $O(\frac{n}{p})$
    $cell \leftarrow (\lfloor \frac{p_i.x}{\delta} \rfloor, \lfloor \frac{p_i.y}{\delta} \rfloor)$
    if $H.$***findCell***$(cell)$ **then**
      $H.$***addCell***$(cell)$ $\triangleright$ constant times for each chain
    end if
    $H.$***findCell***$(cell).$***addPoint***(p_i)
  end for
end procedure

Parallelization

procedure COMPUTE(**in** H, k, id, **out** $result$)

  $pairs \leftarrow 0, pcnt \leftarrow 0$

  for $i \leftarrow \lceil \frac{k}{\#PE} \rceil \cdot id$ **to** $min(\lceil \frac{k}{\#PE} \rceil \cdot (id+1) - 1, k-1)$ **do** $\quad \triangleright O(\frac{k}{p})$

    $pairs \leftarrow pairs + |H[i]| \cdot |\textbf{Neighbor}(H[i])|$

  end for

  $total \leftarrow \textbf{ReduceSum}(paris@id)$ $\quad \triangleright O(\log p)$

  $pcnt \leftarrow pairs > 0?0 : min(1, \lfloor \frac{pairs}{total} \rfloor) \cdot \#PE$

  $pre\_pcnt@id \leftarrow \textbf{PrefixSum}(pcnt@id)$ $\quad \triangleright O(\log p)$

  $id' \leftarrow \textbf{BinarySearch}(id, pre\_pcnt@[0..\#PE-1])$ $\quad \triangleright O(\log p)$

  $P \leftarrow$ the $(pre\_pcnt@id' - id + 1)$-th portion of pairs from $H[\lceil \frac{k}{\#PE} \rceil \cdot id']$ to $H[min(\lceil \frac{k}{\#PE} \rceil \cdot (id+1) - 1, k-1)]$ $\quad \triangleright O(\frac{n}{p})$

  $r \leftarrow \textbf{ShortestDistance}(P)$ $\quad \triangleright O(\frac{n}{p})$

  $result \leftarrow \textbf{ReduceMin}(r@id)$ $\quad \triangleright O(\log p)$

end procedure

Parallelization

0 Cell set $H = \emptyset$, $result = \infty$ ## H concurrent Hashmap

1 Choose set S of $n^{\frac{2}{3}}$ samples ## allocate p PEs — $O(\frac{n}{p} + \log p)$

$\delta = min_{x,y \in S, x \neq y}(dist(x, y))$ ## recurse once — $O(\frac{n^{\frac{8}{9}}}{p} + \log p)$

2 **for** $i = 1$ **to** n **do** ## allocate p PEs — $O(\frac{n}{p})$

$c = cell(p_i)$

if *not* $H.findCell(c)$ **then**

$H.addCell(\{c, idx = \{c.x, c.y\}\}$ ## maximal n times

$H.findCell(c).addPoint(p_i)$

3 **foreach** c_1 **in** H **do** — $O(\frac{n}{p} + \log p)$

foreach c_2 **in** $\{c_1\} \cup neighbor(c_1)$ **do**

foreach (p_i, p_j) **in** $c_1 \times c_2$

$result = min(result, dist(p_i, p_j))$

Total runtime: $O(\frac{n}{p} + \log p)$

Reference I

[BS76] Jon Louis Bentley and Michael Ian Shamos. "Divide-and-conquer in multidimensional space". In: *Proceedings of the eighth annual ACM symposium on Theory of computing*. STOC '76. Hershey, Pennsylvania, USA: ACM, 1976, pp. 220–230. DOI: 10.1145/800113.803652. URL: http://doi.acm.org/10.1145/800113.803652.

[CG89] Benny Chor and Oded Goldreich. "On the power of two-point based sampling". In: *Journal of Complexity* 5.1 (1989), pp. 96–106.

[Die+97] Martin Dietzfelbinger et al. "A reliable randomized algorithm for the closest-pair problem". In: *Journal of Algorithms* 25.1 (1997), pp. 19–51.

Reference II

[FH78] Steven Fortune and John E Hopcroft. *A note on Rabin's nearest-neighbor algorithm*. Tech. rep. Cornell University, 1978.

[Goe03] Brian Goetz. *Java theory and practice: Building a better HashMap*. 2003. URL: `http://www.ibm.com/developerworks/java/library/j-jtp08223/index.html/`.

[KM95] Samir Khuller and Yossi Matias. "A simple randomized sieve algorithm for the closest-pair problem". In: *Information and Computation* 118.1 (1995), pp. 34–37.

[KT06] J. Kleinberg and E. Tardos. "Algorithm Design". In: Pearson Education, 2006. Chap. 13 Randomized Algorithms.

[Lip09] Richard J. Lipton. *Rabin Flips a Coin*. 2009. URL: `http://rjlipton.wordpress.com/2009/03/01/rabin-flips-a-coin/` (visited on 07/15/2013).

Reference III

[Rab76] Michael Oser Rabin. “Probabilistic algorithms”. In: *Algorithms and Complexity: New Directions and Recent Results*. Ed. by Joseph Frederick Traub. Academic Press, 1976, pp. 21–39.

[Weba] URL: http://www.cse.ust.hk/tcsc/comp670r/Class_1_Notes.ppt/.

[Webb] *Closest Pair of Points Problem*. URL: http://en.wikipedia.org/wiki/Closest_pair_of_points_problem/.

Proof Sketch

Karlsruhe Institute of Technology

Theorem

*Let G_i with $1 \leq i \leq k$ be the **i-th** non-empty cell and c, d some positive constants, then*

$$Prob\left(\sum_{i=1}^{k} |G_i|^2 \leq c \cdot n\right) \geq 1 - \frac{1}{2^{n^d}}$$

Proof Sketch

Sampling Lemma

Let $G = \{G_1, G_2, ..., G_k\}$ be a partition of set P, for which $N(D) \geq n$, where

$$N(G) = \sum_{i=1}^{k} \frac{|G_i| \cdot (|G_i| - 1)}{2}$$

If $n^{\frac{2}{3}}$ pairwise different elements are drawn at random from P then the probability, that two elements will be chosen from the same G_i, is at least $1 - 2^{-n^c}$ for some positive constant c

Proof see

- **Rabin 1976**
- **Dietzfelbinger et al. 1997**

  Estimate the probability using Chebyshev's inequality

Proof Sketch

Lemma 2

Let G_δ be a grid with gap δ and $G_{2\delta}$ another grid with gap 2δ, which is obtained by ignoring every second line of G_δ, then

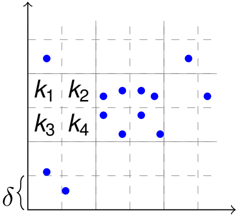
$$N(G_{2\delta}) \leq 4N(G_\delta) + \frac{3}{2}n$$

Let $k = \sum_{i=1}^{4} k_i$

$f(x) = x(x-1)$ convex $\Rightarrow f(\frac{1}{4}k) \leq \frac{1}{4}\sum_{i=1}^{4} f(k_i)$

$$\begin{aligned}\tfrac{1}{2}k(k-1) &= 8 \cdot \tfrac{1}{4}k(\tfrac{1}{4}k - 1) + \tfrac{3}{2}k \\ &\leq 8 \cdot \tfrac{1}{4}\sum_{i=1}^{4} k_i * (k_i - 1) + \tfrac{3}{2}k\end{aligned}$$

$\Rightarrow N(G_{2\delta}) \leq 8 \cdot \frac{1}{2}N(G_\delta) + \frac{3}{2}n$



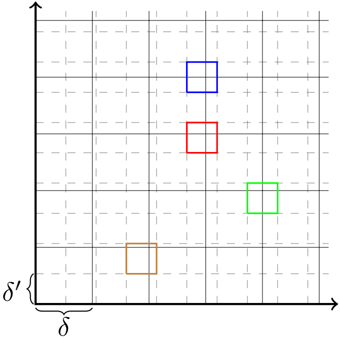
Proof Sketch

Lemma 3

For any grid G_δ, $G_{\delta'}$ with $\delta' \leq \delta$ the following applies

$$N(G_{\delta'}) \leq 16N(G_\delta) + 6n$$

$$\begin{aligned} N(G_{\delta'}) &\leq \sum_{i=1}^{4} N(G_{2\delta}^{i}) \\ &\leq \sum_{i=1}^{4}(4N(G_\delta) + \tfrac{3}{2}n) \\ &= 16N(G_\delta) + 6n \end{aligned}$$



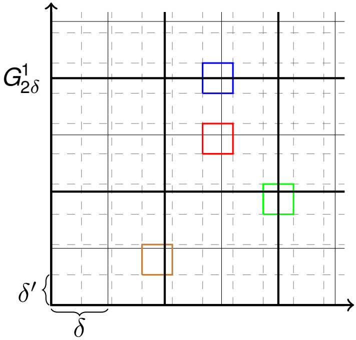
Proof Sketch

Lemma 3

For any grid G_δ, $G_{\delta'}$ with $\delta' \leq \delta$ the following applies

$$N(G_{\delta'}) \leq 16N(G_\delta) + 6n$$

$$\begin{aligned} N(G_{\delta'}) &\leq \sum_{i=1}^{4} N(G_{2\delta}^{i}) \\ &\leq \sum_{i=1}^{4} (4N(G_\delta) + \tfrac{3}{2}n) \\ &= 16N(G_\delta) + 6n \end{aligned}$$



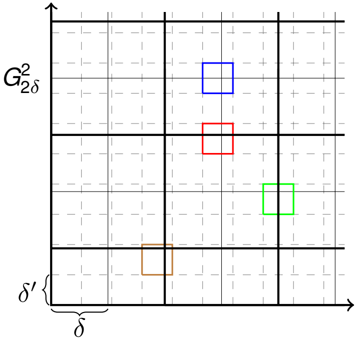
Proof Sketch

Lemma 3

For any grid G_δ, $G_{\delta'}$ with $\delta' \le \delta$ the following applies

$$N(G_{\delta'}) \le 16N(G_\delta) + 6n$$

$$\begin{aligned} N(G_{\delta'}) &\le \sum_{i=1}^{4} N(G^i_{2\delta}) \\ &\le \sum_{i=1}^{4} (4N(G_\delta) + \tfrac{3}{2}n) \\ &= 16N(G_\delta) + 6n \end{aligned}$$



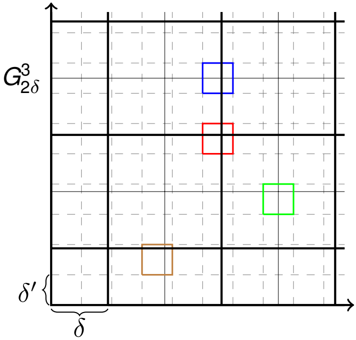
Proof Sketch

Lemma 3

For any grid G_δ, $G_{\delta'}$ with $\delta' \leq \delta$ the following applies

$$N(G_{\delta'}) \leq 16N(G_\delta) + 6n$$

$$\begin{aligned} N(G_{\delta'}) &\leq \textstyle\sum_{i=1}^{4} N(G_{2\delta}^{i}) \\ &\leq \textstyle\sum_{i=1}^{4}(4N(G_\delta) + \frac{3}{2}n) \\ &= 16N(G_\delta) + 6n \end{aligned}$$



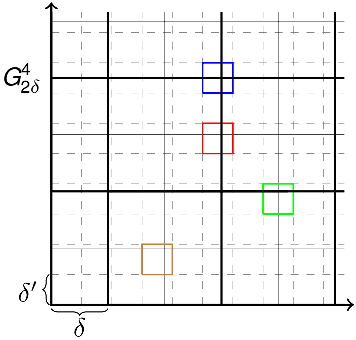
Proof Sketch

Lemma 3

For any grid G_δ, $G_{\delta'}$ with $\delta' \leq \delta$ the following applies

$$N(G_{\delta'}) \leq 16N(G_\delta) + 6n$$

$$\begin{aligned} N(G_{\delta'}) &\leq \sum_{i=1}^{4} N(G_{2\delta}^{i}) \\ &\leq \sum_{i=1}^{4} (4N(G_\delta) + \tfrac{3}{2}n) \\ &= 16N(G_\delta) + 6n \end{aligned}$$



Proof Sketch

- Two of $n^{\frac{2}{3}}$ samples will be very likely in the same cell, if $N(G) \geq n$
- $N(G_{2\delta}) \leq 4N(G_\delta) + \frac{3}{2}n$
- $N(G_{\delta'}) \leq 16N(G_\delta) + 6n \qquad$ if $\delta' \leq \delta$

Proof.

Let δ^* be the grid gap, which makes $n \leq N(G_{\delta^*}) < 5.5n$. (It exists!)
If $n^{\frac{2}{3}}$ samples are randomly chosen, two of them will be in one cell with high probability. Let δ be the distance between them with $\delta < 2\delta^*$, then

$$\begin{aligned} N(G_\delta) &\leq 16N(G_{2\delta^*}) + 6n \\ &\leq 16(4N(G_{\delta^*}) + \frac{3}{2}n) + 6n \in O(n) \end{aligned}$$

□