16 →

written ↓

cs473 Algorithms : Lecture 15 (2022-03-10)

logistics : pset 5 due F17, pset 6 out F17, exam 1 back tonight

last lecture :

- randomized algs
  - dictionary problem [insert, lookup] [array vs linked list]
  - hashing [no (single) hash function works] [fully random] [good expected load] [costly to store]
  - universal hashing [(pseudo) random ⇒ good expected load, cheap to use]

today : randomized algs [lecture 2] [construction]

def : given $p_1=(x_1,y_1),\dots,p_n=(x_n,y_n)\in\mathbb{Z}^2$, the closest pair problem is to find $\arg\min_{i\neq j} dist((x_i,y_i),(x_j,y_j)) =: d(p_i,p_j)$ [value]

$= \sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$ [euclidean 2D dist]

convention : unit-cost arithmetic ⇒ all integers are $O(\lg n)$ bits ⇒ $|x_i|,|y_i| \le N = poly(n)$

assume : all $x_i, y_i \ge 0$ [simplifies notation] [wlog] or $p_i \in \{0,\dots,N\}^2$

thm [lecture 2] : closest pair in $O(n\lg n)$ deterministic time [nontrivial divide and conquer]

thm [today] : $O(n)$ expected randomized time [$n\lg n$ vs n] ["simpler", more flexible]

idea : data structures [hw/lecture 2] = efficient way to store data

↳ dictionaries — user data [hw due] — algorithmic data [intermediate data] [today]

Q : can we verify whether mindist $\ge \Delta$? [significant warmup]

idea : process points in order

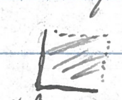
def : $\Delta_k := \min_{i,j\le k} d(p_i,p_j)$

Q [sphere] : Δ_n vs Δ ? Δ_{k-1} vs Δ_k ?

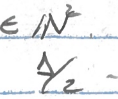
idea : coarse discretization of space

def : $\delta > 0$, $\delta \in \mathbb{R}$

a δ-subsquare is a set of $\mathbb{Z}^2$ given by $S_{\alpha,\beta} = \{(x,y) : x,y\in\mathbb{Z},\ \alpha \le x < \alpha+\delta,\ \beta \le y < \beta+\delta\}$, $\alpha,\beta\in\mathbb{R}$

eg  [open vs closed]

clm [separated points] : $\Delta_k \ge \Delta$ ⇒ each $\Delta/2$-subsquare has ≤ 1 point from $p_1,\dots,p_k$ [similar to lecture 2]

pf : eg  $d((x,y),(z,w)) = \sqrt{\underbrace{(x-z)^2}_{<(\Delta/2)^2}+\underbrace{(y-w)^2}_{<(\Delta/2)^2}}$

$\le \sqrt{\Delta^2/2} = \Delta/\sqrt{2} < \Delta$ [close points] [but $\Delta_k \ge \Delta$] ☒

[$\Delta_k \ge \Delta$ ⇒ rounding loses no info]

idea : round coordinates

def : $(a,b)\in\mathbb{N}^2$. The (a,b) $\Delta/2$-subsquare of N^2 is $S_{a\cdot\frac{\Delta}{2},\, b\cdot\frac{\Delta}{2}}$ [as above] [rat vs $\mathbb{Z}$]

the $\Delta/2$-grid is $\{S_{a,b} : 0 \le a,b \le \frac{2N}{\Delta}\} =: T_{a,b}$ [ex: assumption of rounding]

[want to associate points to subsquares they live in]

[for lecture]

def: A dictionary over $\mathcal{U}$ [universe] [often integers, can be strings, ...] is a data structure for storing a set $S \subseteq \mathcal{U}$ of keys x, along w/ associated values y.

$\cdot$ S vs $S_{a,b}$

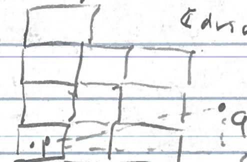
it supports: insert(x,y): add key x to S, w/ value y

[converse to subsquare] lookup(z): check if $z \in S$, if so return value y [complexity in terms of $n = |S|$]

prop: $d(p,q) \le \Delta$ $p \in T_{a,b}$ $q \in T_{c,d}$ [$\Delta/2$ grid]

$\Rightarrow$ $|a-c|, |b-d| \le 2$ [d vs d]

pf: by contraposition



a,b [pythagorean thm]

$\Delta/2$ $\Delta/2$ $\Delta/2$

$|a-c| > 2 \Rightarrow d(p,q) = \sqrt{(x-z)^2 + (y-w)^2} \ge \sqrt{(x-y)^2}$

$= |x-y|$

$|a-c|>2 \Rightarrow > \Delta$

$|b-d| > 2 \Rightarrow$ ... [analogous]

algo: [consider points in order, test to previous points]

$\cdot A$

init dictionary A over $\mathcal{U} = \{(a,b) ; 0 \le a,b \le \frac{2N}{\Delta}\}$ [$\frac{\Delta}{2}$-grid]

for $1 \le i \le n$:

$(a,b) = \left(\lfloor \frac{x_i}{\Delta/2} \rfloor, \lfloor \frac{y_i}{\Delta/2} \rfloor\right)$ [round.]

compute $\Delta'_i := \min_{\substack{p \in A[c,d] \\ |a-c| \le 2 \\ |b-d| \le 2}} d(p_i, p)$

if $\Delta'_i < \Delta$, return p_i, p, Δ'_i arg min

insert [delete] p_i into $A[a,b]$

return "$\Delta_n \ge \Delta$"

prop: suppose $\Delta_i \ge \Delta$. Then (a) algo reaches

(b) $A[a,b]$ empty at that moment [(a,b) non-in cluster]

pf: (a) algo doesn't reach only if finds $(<\Delta)$-close points within $p_1 \ldots p_i \Rightarrow \Delta_i < \Delta$

(b) p_j: $A[a,b] = p_j$ $j < i \Rightarrow p_i, p_j$ in same $\frac{\Delta}{2}$-subsquare $\Rightarrow (<\Delta)$-close $\Rightarrow$

cor: suppose $\Delta_i \ge \Delta$ then for p_i ~ makes ≤ 25 dictionary lookups [25 such (c,d)]

rand vs det [complexity?] [for lecture] **thm**: $S \subseteq \mathcal{U}$ $|S| = n$ $|\mathcal{U}| = N \le \text{poly}(n)$ — returns ≤ 25 points [each cell has ≤ 1 point by above] [hash size constant]

One can in deterministic $O(n)$ time construct a hash function family $\mathcal{H}: \mathcal{U} \to T$

- $|T| \le O(n)$
- choosing $h \in \mathcal{H}$ takes $O(1)$ space to store

takes $O(1)$ time to eval

- insertion, lookup operations are $O(1)$ expected time [all dist $\geq \Delta$] — also is correct

cor: if $\Delta_n \geq \Delta$ then — runs in $O(n)$ expected time

pf: uses dictionary on n subsquares in N^2 ∈ poly(n) size universe [thm applies]

uses — n insertions [one per point, never reinsert, as $\Delta_n \geq \Delta$]

— $\leq 25n$ lookups [$\Delta_n \geq \Delta$ each lookup ∃ single point]

⇒ $O(n)$ dictionary ops ⇒ $O(n)$ expected time

← linearity of expectation

cor: if $\Delta_{k-1} \geq \Delta$, $\Delta_k < \Delta$ [first dist $< \Delta$]

— algo is correct, returns $\Delta' = \Delta_k$

[no need to examine all points]

— runs in expected $O(k)$ time

pf: on $p_1, \ldots, p_{k-1}$ (chosen): $\Delta_{k-1} \geq \Delta$ ⇒ $\sim k$ insertion, $\leq 25k$ lookup [as above]

p_k: 25 more lookups [all subsquares still have ≤ 1 point]

$\Delta_k < \Delta \overset{[pigeonhole]}{\Rightarrow}$ will find p_i w/ $d(p_i, p_k) = \Delta_k < \Delta$ ($i<k$)

⇒ correct [find point] || and return it [correct]

$O(k)$ expected time ⇒ $\boxed{\text{no}}$ insertion [also done]

thm: $\Delta > 0$ given, will either — declare $\Delta_n \geq \Delta$ in $O(n)$ expected time

— find p_i, p_k with $i < k$ in $O(k)$ expected time

— $d(p_i, p_k) = \Delta_k < \Delta$

Q: Δ unknown?

idea: try a Δ [candidate], either — finish w/ $\Delta_n \geq \Delta$

— identify $\Delta' < \Delta$

Q: how many times can Δ change? [can only go down]

A: can find examples w/ $\Theta(n)$ changes ⇒ $\Theta(n^2)$ runtime [worse than deterministic soln]

[randomness needed]

idea: process points in $\boxed{\text{random}}$ order [randomness of algo] [random order]

algo: randomly reorder $p_1, \ldots, p_n$

$(p, q) = (p_1, p_2)$

$\Delta = d(p, q)$ [candidate soln]

while

verify $\Delta_n \geq \Delta$ [existing algo]

if $\Delta_n \geq \Delta$, return $(p, q), \Delta$

else $(p, q), \Delta \leftarrow (p', q'), \Delta'$ [new min dist, argmin]

prop: algo is correct [clear]

Q: complexity?

prop: algo can only update from $\Delta \to \Delta' = \Delta_k$, $\boxed{\text{once}}$ [Δ always goes down] [so does this happen?]

def: $I_k = \begin{cases} 1 & \text{verification algo updates from } \Delta \to \Delta' = \Delta_k \\ 0 & \text{else} \end{cases}$

prop: # dict ops of algo is $O(n + \sum_{k=1}^{n} k \cdot I_k)$

pf: $O(n)$ ops final pass [must occur]

$O(k)$ ops to update $\Delta \to \Delta' = \Delta_k$, happens iff $I_k = 1$

[how to analyze]

prop: $Pr[I_k = 1] \le 2/k$

pf: $I_k = 1$ iff update $\Delta \to \Delta_k$
 was Δ_j some $j < k$

iff $\Delta_j = \Delta_{k-1} > \Delta_k$

iff $\min_{i,j<k} d(p_i, p_j) > \min_{i<k} d(p_i, p_k)$

$\Rightarrow$ random order places either point of $\arg\min_{i,j \le k} d(p_i, p_j)$ in last position of first k points

eg $p_i \; p_j - p_k$

$\Rightarrow \le 2/k$ [two bad options for shuffle] $p_i \, p_j$

rmk: $\boxed{\text{rare}}$ equality if $\boxed{\text{multiple}}$ minimizers [$\Rightarrow I_k = 0$ always] random order ☐

cor: algo runs in $O(n)$ expected time

pf: $\mathbb{E}_\sigma^{[\text{perm}]} \mathbb{E}_A^{[\text{dict}]} [\underbrace{D_{(\Delta)}}_{O(n)\text{ ops}}^{[O(n) \text{ dict ops}]} + \sum_k \underbrace{D_k}_{O(k)\text{ ops}}^{[\text{total dict}]} \cdot I_k^{[I_k \text{ rand var}]}]$

$= \mathbb{E}_\sigma [\underbrace{\mathbb{E}_A[D]}_{O(n) \text{ [linearity]}} + \sum_k \mathbb{E}_A [D_k \cdot I_k]]$

$O(n)$ ops, n total rand

$\mathbb{E}_A[D_k I_k]$: I_k non rand wrt A, A [only on σ]

$O(k)$ ops, k rand var, indep of A [expectation on results]

$= I_k \cdot O(k)$

$= \mathbb{E}_\sigma [O(n) + \sum_k O(k) \cdot I_k]$

[linearity]

$O(n) + \underbrace{\sum_k \underbrace{O(k) \cdot \underbrace{Pr[I_k = 1]}_{\le 2/k}}_{O(1)}}_{O(n)}$ [indicator]

$= O(n)$. ☐

rmk: need to be careful when applying linearity of expectation to sum of a $\boxed{\text{random}}$ number of random variables [use independence]

today: randomized algo
- closest pair
  - $O(n \lg n)$ deterministic [lecture 2 divide and conquer]
  - verification dist $\ge \Delta$ [insert points, check local subsquares via dictionary]
  - randomly order points

next lecture: randomized algo [w/ high probability of many updates]

logistics:
- pset 5 due F17
- pset 6 out F17
- exam 2 back pending