sep 12 2013
shelat
4102
divide & conquer
closest points
matrixmult
median
closest pair

of points
simple brute force approach takes $\Theta(n^2)$
solve the large problem by solving smaller problems and combining solutions
closest left

closest on the right
Divide & Conquer

winner!
Divide & Conquer winner!
exhaustively trying each pair $\implies \Theta(n^2)$
We need is a linear time approach to detecting a Romeo-Julia winner. Θ(n) time inspecting these points.

We know the answer $< f$. 
Imaginary grid for our reasoning.
Imagine there is a grid of cubbies starting at the lowest Y point.
Each cubby can have only 1 point in it (at most)

**Lonely cubby property.**

This point will be

$$\sqrt{(\frac{\delta}{2})^2 + (\frac{\delta}{2})^2} = \frac{\sqrt{2}}{2} \cdot \delta < \delta$$
FACT: At most 1 point in each cubby
FACT: \( \leq 1 \) point per cubby
FACT: $\leq 1$ point per cubby
radius of 16 squares (one of which is Mrs. Red’s own square) larger.

⇒ there are only 17 possible cubbies for Mrs. Red’s candidate Romeo
Details: How to do step 1?

Merge sort point in X coord: 13 1 5 14 9 10 7 6 8 11 2 3 9 12
sort in Y coord:
sorted in X: 13 1 5 14 9 10 7 9 8 11 2 3 4 12
sorted in Y: 6 5 12 11 10 3 13 4 9 8 7 2 1 14
sorted in X: 13 1 5 14 9 10 7 9 8 11 2 3 4 12
sorted in Y: 6 5 12 11 10 3 13 4 9 8 7 2 1 14

Given \( S_x, S_y \) compute \( L_x, L_y \)

\[ R_x, R_y \in \Theta(n) \] to produce all \( \Theta \) of these sets.
ClosestPair(P)

- Compute Sorted-in-X list SX \( \Theta(n \log n) \)
- Compute Sorted-in-Y list SY \( \Theta(n \log n) \)

Closest(P, SX, SY) \( \Theta(n \log n) \)

Total \( \Theta(n \log n) \)
Closest(P,SX,SY)

**Base case** when |P| <= 4, brute force.

Let \( q \) be the middle entry of \( SX \)

Compute Left, Right, Lx, Ly, Rx, Ry based on \( q \)

\[
\delta_{r,j} = \min(\text{closest(L, Lx, Ly)}, \text{closest(R, Rx, Ry)})
\]

Sx = \{ Scan thru SY and choose all points that are within \( \delta \) of \( q \). \}

For each \( x \in Sx \) (in order that it was added)

Inspect the next 15 cubbies above or equal in \( y \).

If any cubby contains a point that is closer than \( \delta \), then revise \( \delta_{r,j} \)

Return \( \delta_{r,j} \)
sorted in X: 13 1 5 14 9 10 7 9 8 11 2 3 4 12
sorted in Y: 6 5 12 11 10 3 13 4 9 8 7 2 1 14
sorted in X: 13 1 5 14 9 10 7 9 8 11 2 3 4 12
sorted in Y: 6 5 12 11 10 3 13 4 9 8 7 2 1 14
Closest(P, SX, SY)

Let q be the middle-element of SX

Divide P into Left, Right according to q

delta, r, j = MIN(Closest(Left, LX, LY), Closest(Right, RX, RY))

Mohawk = { Scan SY, add pts that are delta from q.x }

For each point x in Mohawk (in order):

Compute distance to its next 15 neighbors
Update delta, r, j if any pair (x, y) is < delta

Return (delta, r, j)

Total: \( T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \)
Closest(P, SX, SY)

Let q be the middle-element of SX

Divide P into Left, Right according to q

delta, r, j = MIN(Closest(Left, LX, LY), Closest(Right, RX, RY))

Mohawk = { Scan SY, add pts that are delta from q.x }

For each point x in Mohawk (in order):

- Compute distance to its next 15 neighbors
- Update delta, r, j if any pair (x, y) is < delta

Return (delta, r, j)

Can be reduced to 7!
Running time for Closest pair algorithm

\[ T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \]

\[ \Theta(n \log n) \]
\( T(n) = 2T(n/2) + \Theta(n) = \Theta(n \log n) \)
Matrix multiplication
\[
\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix} \star \begin{bmatrix}
5 & 6 \\
7 & 8
\end{bmatrix} = \begin{bmatrix}
1.5 + 2.7 & 1.6 + 2.8 \\
3.5 + 4.7 & 3.6 + 4.8
\end{bmatrix}
\]
\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix} \star \begin{bmatrix}
5 & 6 \\
7 & 8 \\
\end{bmatrix} = \begin{bmatrix}
5 + 14 & 6 + 16 \\
15 + 28 & 18 + 32 \\
\end{bmatrix} = \begin{bmatrix}
19 & 22 \\
43 & 50 \\
\end{bmatrix}
\]
\[
\begin{bmatrix}
    a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
    a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
    \vdots \\
    a_{n,1} & a_{n,2} & \cdots & a_{n,n}
\end{bmatrix}
\begin{bmatrix}
    b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\
    b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\
    \vdots \\
    b_{n,1} & b_{n,2} & \cdots & b_{n,n}
\end{bmatrix}
= \begin{bmatrix}
    c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\
    c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\
    \vdots \\
    c_{n,1} & c_{n,2} & \cdots & c_{n,n}
\end{bmatrix}
\]

\[C_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}\]

- each \(C_{i,j}\) requires \(\Theta(n)\)

- there are \(n^2\) such elements, so naive algo is \(\Theta(n^3)\)
\[ a_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j} \]
If parts that are
$\mathcal{Y}_2 \times \mathcal{W}_2$
\[
\begin{bmatrix}
\frac{A}{C} & B \\
\frac{E}{G} & F
\end{bmatrix}
\begin{bmatrix}
\frac{E}{G} & F \\
\frac{G}{H} & H
\end{bmatrix} =
\begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH
\end{bmatrix}
\]

\[T(n) = \Theta\left(\frac{n^3}{2}\right) + \Theta(n^2) = \Theta\left(n^{\log_2 8}\right) = \Theta(n^3)\]

\text{time for an \ n x n matrix multiplication}
\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\begin{bmatrix}
E & F \\
G & H
\end{bmatrix}
= \begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH
\end{bmatrix}
\]
\[
\begin{bmatrix}
A & B \\
C & D \\
\end{bmatrix}
\begin{bmatrix}
E & F \\
G & H \\
\end{bmatrix}
= 
\begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH \\
\end{bmatrix}
\]

\[T(n) = 8T(n/2) + \Theta(n^2)\]

\[\Theta(n^3)\]
\[
\begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH
\end{bmatrix}
\rightarrow P_1 + P_2
\]
\[\quad (AF - AH) + AH + BH \]

\[P_1 = A(F - H)\]
\[P_2 = (A + B)H\]
\[P_3 = (C + D)E\]
\[P_4 = D(G - E)\]
\[P_5 = (A + D)(E + H)\]
\[P_6 = (B - D)(G + H)\]
\[P_7 = (A - C)(E + F)\]

Each is an \(\frac{n^2}{2}\)-matrix mult
\[
\begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH
\end{bmatrix}
\]

\[P_1 = A(F - H)\]
\[P_2 = (A + B)H\]
\[P_3 = (C + D)E\]
\[P_4 = D(G - E)\]
\[P_5 = (A + D)(E + H)\]
\[P_6 = (B - D)(G + H)\]
\[P_7 = (A - C)(E + F)\]
\[
\begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH
\end{bmatrix}
\]

\[P_1 = A(F - H)\]

\[P_2 = (A + B)H\]

\[P_3 = (C + D)E\]

\[P_4 = D(G - E)\]

\[P_5 = (A + D)(E + H)\]

\[P_6 = (B - D)(G + H)\]

\[P_7 = (A - C)(E + F)\]
\[ R = \begin{bmatrix} \frac{AE + BG}{T = P_3 + P_4} & \frac{AF + BH}{U = P_5 + P_1 - P_3 - P_7} \\ \end{bmatrix} = P_1 + P_2 \]

\[ P_1 = A(F - H) \]
\[ P_2 = (A + B)H \]
\[ P_3 = (C + D)E \]
\[ P_4 = D(G - E) \]
\[ P_5 = (A + D)(E + H) \]
\[ P_6 = (B - D)(G + H) \]
\[ P_7 = (A - C)(E + F) \]

\[ T(n) = \frac{7}{7} T \left( \frac{n}{2} \right) + 18 \Theta(n^2) \]
\[ \Theta(n^2) \sim n^{2.81} \]
\[
\begin{align*}
R &= P_5 + P_4 - P_2 + P_6 \\
T &= P_3 + P_4 \\
U &= P_5 + P_1 - P_3 - P_7 \\
\end{align*}
\]

\[
\begin{align*}
P_1 &= A(F - H) \\
P_2 &= (A + B)H \\
P_3 &= (C + D)E \\
P_4 &= D(G - E) \\
P_5 &= (A + D)(E + H) \\
P_6 &= (B - D)(G + H) \\
P_7 &= (A - C)(E + F) \\
\end{align*}
\]

\[
M(n) = 7M(n/2) + 18n^2
\]

\[
= \Theta(n^{\log_2 7}) \sim 2.81^n
\]
taking this idea further

3x3 matrices

\[ T(n) = 2T(n/3) + \Theta(n^2) \]

\[
= \Theta(n \log_3 2^3) = \sim n^{2.77}
\]
1978 victor pan method

70x70 matrix using 143640 mults

what is the recurrence:

\[ T(n) = 143640 T(n/70) + \Theta(n^2) \]

\[ \Theta(n \log_7 143640) \approx 2.995n \]
Median
problem: given a list of \( n \) elements, find the element of rank \( n/2 \). (half are larger, half are smaller)

\[
\begin{align*}
\text{can sort the list} \\
\tilde{O}(n \log n)
\end{align*}
\]
problem: given a list of $n$ elements, find the element of rank $n/2$. (half are larger, half are smaller)

first solution: sort and pluck.

$O(n \log n)$
problem: given a list of $n$ elements, find the element of rank $i$. 

ABSTRACTION: find the $i^{th}$ element in BAYU

\[ i = \frac{n}{2} \] corresponds to MEDIAN.
problem: given a list of \( n \) elements, find the element of rank \( i \).

key insight: we do not have to “fully” sort. semi sort can suffice.
pick first element
partition list about this one
see where we stand
review: how to partition a list
review: how to partition a list

GOAL: start with THIS LIST and END with THAT LIST

less than greater than
review: how to partition a list
review: how to partition a list

1. First element that is larger than

2. Swap w/ back ptr
review: how to partition a list
review: how to partition a list
review: how to partition a list

partitioning a list about an element takes linear time.
select \( (i, A[1, \ldots, n]) \)

1. partition using the first element
2. if rank \( i < \text{rank } p \), select \( (i, A[1, \ldots, p]) \)
3. if rank \( i > \text{rank } p \), select \( (i-p, A[p, \ldots, n]) \)
4. if rank \( i = \text{rank } p \) = return \( p \).
select \((i, A[1, \ldots, n])\)

- handle base case.
- partition list about first element
  - if pivot is position \(i\), return pivot
  - else if pivot is in position > \(i\)
    - select \((i, A[1, \ldots, p - 1])\)
  - else
    - select \(((i - p - 1), A[p + 1, \ldots, n])\)

Let's imagine that each partition splits the array into 2 parts of size \(n/2\).

\[
T(n) = T\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n)
\]
select \((i, A[1, \ldots, n])\)  Assume our partition always splits list into two equal parts

handle base case.
partition list about first element
if pivot is position \(i\), return pivot
else if pivot is in position \(> i\)  select \((i, A[1, \ldots, p - 1])\)
else  select \(((i - p - 1), A[p + 1, \ldots, n])\)
Assume our partition always splits list into two equal parts.

- **Handle base case.**
- **Partition list about first element.**
  - If pivot is position $i$, return pivot.
  - Else if pivot is in position $> i$, select $(i, A[1, \ldots, p - 1])$.
  - Else select $((i - p - 1), A[p + 1, \ldots, n])$.

$$T(n) = T(n/2) + O(n)$$

$$\Theta(n)$$
problem: what if we always pick bad partitions?

problem: what if we always pick bad partitions?
problem: what if we always pick bad partitions?
problem: what if we always pick bad partitions?
problem: what if we always pick bad partitions?

\[ O(n^2) \]
select \((i, A[1, \ldots, n])\)

handle base case.
partition list about first element
if pivot is position \(i\), return pivot
else if pivot is in position \(> i\)
else
select \((i, A[1, \ldots, p - 1])\)
else select \(((i - p - 1), A[p + 1, \ldots, n])\)
\textbf{select } (i, A[1, \ldots, n])

handle base case.
partition list about first element
if pivot is position \(i\), return pivot
else if pivot is in position \(> i\) \hspace{1em} \text{select } (i, A[1, \ldots, p - 1])
else \hspace{1em} \text{select } ((i - p - 1), A[p + 1, \ldots, n])

\[ T(n) = T(n - 1) + O(n) \]

\[ \Theta(n^2) \]
Needed:

a good partition element

\[ \text{partition} (A[1, \ldots, n]) \]
Needed:

a good partition element

partition \((A[1, \ldots, n])\) produce an element where 30% smaller, 30% larger
solution:
bootstrap

image: gucci
image: d&g
image: mark nason
partition \((A[1, \ldots, n])\)
partition \((A[1, \ldots, n])\)
partition \((A[1, \ldots, n])\)
partition \((A[1,\ldots,n])\)

median of each group

form a smaller list

\(B[1,\ldots,\lceil n/5 \rceil]\)

select \((\lceil n/5 \rceil/2, B[1,\ldots,\lceil n/5 \rceil])\)

use the median of this smaller list as the partition element
partition ($A[1, \ldots, n]$)
partition \((A[1, \ldots, n])\)

divide list into groups of 5 elements
find median of each small list
gather all medians
call select(...) on this sublist to find median
return the result
partition \((A[1, \ldots, n])\)

- divide list into groups of 5 elements
- find median of each small list
- gather all medians
- call select(...) on this sublist to find median
- return the result

\[
P(n) = S(\lceil n/5 \rceil) + O(n)
\]
a nice property of our partition
a nice property of our partition
a nice property of our partition
SWITCH TO A BIGGER EXAMPLE
a nice property of our partition
a nice property of our partition

$$3 \left( \left\lceil \frac{1}{2} \left\lfloor \frac{n}{5} \right\rfloor \right\rceil - 2 \right)$$

$$\geq \frac{3n}{10} - 6$$

this implies there are at most \( \frac{7n}{10} + 6 \) numbers larger than \( / \) smaller \( / \)
a nice property of our partition
\[ \leq \frac{7n}{10} + 6 \]
select \( (i, A[1, \ldots, n]) \)
select \((i, A[1, \ldots, n])\)

  handle base case for small list
  else pivot = FindPartitionValue(A,n)
  partition list about pivot
  if pivot is position \(i\), return pivot
  else if pivot is in position > \(i\)  select \((i, A[1, \ldots, p - 1])\)
  else select \(((i - p - 1), A[p + 1, \ldots, n])\)
FindPartition \((A[1, \ldots, n])\)

- divide list into groups of 5 elements
- find median of each small list
- gather all medians
- call select(...) on this sublist to find median
- return the result

\[
P(n) = S(\lceil n/5 \rceil) + O(n)
\]
select \((i, A[1, \ldots, n])\)

handle base case for small list
else pivot = FindPartitionValue(A, n)
partition list about pivot
if pivot is position \(i\), return pivot
else if pivot is in position > \(i\)  select \((i, A[1, \ldots, p - 1])\)
else select \(((i - p - 1), A[p + 1, \ldots, n])\)

\[ S(n) = S(\lceil n/5 \rceil) + O(n) + S(7n/10 + 6) \]
select \((i, A[1, \ldots, n])\)

handle base case for small list

else pivot = FindPartitionValue(A,n)

partition list about pivot

if pivot is position \(i\), return pivot

else if pivot is in position > \(i\)

else

select \((i, A[1, \ldots, p - 1])\)

else select \(((i - p - 1), A[p + 1, \ldots, n])\)

\[
S(n) = S(\lceil n/5 \rceil) + O(n) + S(7n/10 + 6) + \Theta(n)
\]
arbitrage
input: array of \( n \) numbers

\[
\begin{array}{ccccccc}
1 & \ldots & \ldots & \ldots & \ldots & \ldots & n
\end{array}
\]

goal:
first attempt
first attempt

\[
\text{arbit}(A[1...n])
\]
first attempt

arb(A[1...n])
base case if |A|=1
lg = arb(left(A))
rg = arb(right(A))
minl = min(left(A))
maxr = max(right(A))
return max{maxr-minl,lg,rg}
better approach
second attempt

arbit+(A[1...n])

    base case if |A|=1
second attempt

arbit+(A[1...n])

    base case if |A|=1
    (lg,minl,max) = arbit(left(A))
    (rg,mi,maxr) = arbit(right(A))

    return max{maxr-minl,lg,rg}