divide & conquer
closest points
matrix mult
median
|   7.5  |  7.8  |  9.5  |  3.7  |  26.1 |
closest pair
of points
simple brute force approach takes
solve the large problem by
solving smaller problems
and combining solutions
Divide & Conquer
Divide & Conquer
Divide & Conquer

winner!
Divide & Conquer
Divide & Conquer winner!
Mohawk can contain all of the input points.
Imagine there is a grid of cubbies starting at the lowest Y point.
\[ \delta > \frac{\sqrt{2}}{2} \delta \]

Insight:

Cubbies have < 1 point.
FACT: At most 1 point in each cubby
FACT: <=1 point per cubby
FACT: $\leq 1$ point per cubby
FACT: \( \leq 1 \) point per cubby
FACT: \( \leq 1 \) point per cubby
\[ \delta \]

\[ \frac{\delta}{2} \]

\[ \frac{\delta}{2} \]

\[ \leq 15 \]

\[ \leq 7 \text{ buckets} \]
Start
Check the next 15 boxes
Visit its by y-order
Check the next <15 boxes
Check the next <15 boxes
Closest(P)

Base case: if \(|P| \leq 2\), brute force.

Let \(q\) be the mid-point along \(x\) coordinates.

\(L, R = \text{split points into left & right halves according to } q\)

\(d_L = \text{closest}(L) \implies \text{let } \delta = \min(d_L, d_R)\)

\(M = \text{Mohawk}(q, \delta) \quad \text{// set of points that are w/m } \delta \text{ of } \forall x\)

for all points \(r \in M\) (sorted according to \(y\)-coordinate)

check next 15 points (in \(y\)-order) for a julia.

\(\delta = \min(\delta, d(r, ij))\)

Return \(\delta\).
Closest(P)

Base Case: If <8 points, brute force.

1. Let q be the “middle-element” of points $\Theta(n)$

2. Divide P into Left, Right according to q $\Theta(n)$

3. $\delta, r, j = \min(\text{Closest(Left)}, \text{Closest(Right)})$ $2T(n/2)$

4. Mohawk = \{ Scan P, add pts that are $\delta$ from q.x \} $\Theta(r)$

5. For each point x in Mohawk (in y-order):
   \[ \Theta(r) \]
   \[
   \begin{align*}
   \text{Compute distance to its next 15 neighbors} \\
   \text{Update delta, r, j if any pair (x, y) is < $\delta$}
   \end{align*}
   \]

6. Return $(\delta, r, j)$

   $T(n) = 2T(n/2) + \Theta(r)$ by Master's
   $\Rightarrow T(n) = \Theta(n \log n)$
**Closest(P)**

Base Case: If <8 points, brute force.

1. Let q be the "middle-element" of points
2. Divide P into Left, Right according to q
3. \( \delta, r, j = \text{MIN}(\text{Closest(Left)}, \text{Closest(Right)}) \)
4. Mohawk = \{ Scan P, add pts that are \( \delta \) from q.x \}
5. For each point x in Mohawk (in y-order):
   - Compute distance to its next 15 neighbors
   - Update \( \delta, r, j \) if any pair \((x,y)\) is < \( \delta \)
6. Return \((\delta,r,j)\)

Can be reduced to 7!
Details: How to do step 1?
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
ClosestPair(P),
Compute Sorted-in-X list SX → mergesort \( \Theta(n \log n) \)
Compute Sorted-in-Y list SY → \( \Theta(n \log n) \)
Closest(P, SX, SY) → \( \Theta(n \log n) \)

Overall solution is still \( \Theta(n \log n) \)
Closest(P, SX, SY)

Let q be the middle-element of SX

Divide P into Left, Right according to q

\[ \delta, r, j = \min(\text{Closest}(\text{Left}, LX, LY), \text{Closest}(\text{Right}, RX, RY)) \]

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 \( (\delta, r, j) \)
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!
sorted in X: 13 1 5 14 9 10 7 6 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. Scan to get LY, RY.

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)
Closest(P,SX,SY)

Let q be the middle-element of SX

Divide P into Left, Right according to q. Scan to get LY, RY.

\[ \text{delta},r,j = \text{MIN}(\text{Closest(Left, LX, LY)} \text{ Closest(Right, RX, RY)}) \]

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!
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. Scan to get LY, RY.

\[ \text{delta, r, j = MIN(Closest(Left, LX, LY) \ Closest(Right, RX, RY))} \]

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)
Closest(P, SX, SY)

Let q be the middle-element of SX
Divide P into Left, Right according to q. Scan to get LY, RY.

\[ \text{delta, r, j} = \text{MIN} \left( \text{Closest(Left, LX, LY)}, \text{Closest(Right, RX, RY)} \right) \]

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

\[ T(n) = \]
$$T(n) = 2T(n/2) + \Theta(n) = \Theta(n \log n)$$
public ClosestPair(Point2D[] points) {
    int N = points.length;
    if (N <= 1) return;

    // sort by x-coordinate (breaking ties by y-coordinate)
    Point2D[] pointsByX = new Point2D[N];
    for (int i = 0; i < N; i++)
        pointsByX[i] = points[i];
    Arrays.sort(pointsByX, Point2D.X_ORDER);

    // check for coincident points
    for (int i = 0; i < N-1; i++) {
        if (pointsByX[i].equals(pointsByX[i+1])) {
            bestDistance = 0.0;
            best1 = pointsByX[i];
            best2 = pointsByX[i+1];
            return;
        }
    }

    // sort by y-coordinate (but not yet sorted)
    Point2D[] pointsByY = new Point2D[N];
    for (int i = 0; i < N; i++)
        pointsByY[i] = pointsByX[i];

    // auxiliary array
    Point2D[] aux = new Point2D[N];

    closest(pointsByX, pointsByY, aux, 0, N-1);
}

int mid = lo + (hi - lo) / 2;
Point2D median = pointsByX[mid];

// compute closest pair with both endpoints in left subarray or both in right subarray
double delta1 = closest(pointsByX, pointsByY, aux, lo, mid);
double delta2 = closest(pointsByX, pointsByY, aux, mid+1, hi);
double delta = Math.min(delta1, delta2);

// merge back so that pointsByY[lo..hi] are sorted by y-coordinate
merge(pointsByY, aux, lo, mid, hi);

// aux[0..M-1] = sequence of points closer than delta, sorted by y-coordinate
int M = 0;
for (int i = lo; i <= hi; i++) {
    if (Math.abs(pointsByY[i].x() - median.x()) < delta)
        aux[M++] = pointsByY[i];
}

// compare each point to its neighbors with y-coordinate closer than delta
for (int i = 0; i < M; i++) {
    // a geometric packing argument shows that this loop iterates at most 7 times
    for (int j = i+1; (j < M) && (aux[j].y() - aux[i].y() < delta); j++) {
        double distance = aux[i].distanceTo(aux[j]);
        if (distance < delta) {
            delta = distance;
            if (distance < bestDistance) {
                bestDistance = delta;
                best1 = aux[i];
                best2 = aux[j];
                // StdOut.println("better distance = " + delta + " from " + best1 + " to " + best2);
            }
        }
    }
}

return delta;

arbitrage
input: array of $n$ numbers

1, ..., $n$

goal: to find index $i,j$ s.t. $i < j$ which

$$\max \{ i \geq j \mid A[i,j] - A[i,j] \}$$

We want an $\Theta(n \log n)$ algorithm!
first attempt

arbit(A[1...n])


→ (i,j) ← Arbit(A[i+1,...,n])
→ (i′,j′) ← Arbit(A[i+2,...,n])

r* ← min(A[i+1,...,n])
j* ← max(A[i+1,...,n])

Return max { (i,j) (i′,j′) (r,j) }  l mean

A[j′]−A[i′]
first attempt

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

base case if $|A| \leq 2$

\( \text{lg} = \text{arbit}(\text{left}(A)) \rightarrow T(\frac{n}{2}) \)

\( \text{rg} = \text{arbit}(\text{right}(A)) \rightarrow T(\frac{n}{2}) \)

\( \text{minl} = \min(\text{left}(A)) \rightarrow \Theta(n) \)

\( \text{maxr} = \max(\text{right}(A)) \rightarrow \Theta(n) \)

return \( \max\{\text{maxr} - \text{minl}, \text{lg}, \text{rg}\} \)

\( T(n) = 2T(\frac{n}{2}) + \Theta(n) \rightarrow \Theta(n \log n) \) solution.
first attempt: time $\Theta(n \log n)$

```
arbit(A[1...n])
  base case if $|A| \leq 2$
  lg = arbit(left(A))
  rg = arbit(right(A))
  minl = min(left(A))
  maxr = max(right(A))
  return max{maxr-minl, lg, rg}
```
better approach
Can we find a solution that has $T(n) = 2T(n/2) + O(1)$?
Can we find a solution that has $T(n) = 2T(n/2) + O(1)$?

\[
\text{minl} = \min\left(\text{left}(A)\right) \in \Theta(n)
\]

\[
\text{maxr} = \max\left(\text{right}(A)\right)
\]

return max\{maxr-minl, lg, rg\}
second attempt

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

\text{base case if } |A| \leq 2

\begin{align*}
(l_g, \text{min}_G, \text{max}_G) & \leftarrow \text{Arbit}(A_{[1..\lceil n/2 \rceil]}) \\
(r_g, \text{min}_R, \text{max}_R) & \leftarrow \text{Arbit}(A_{[\lceil n/2 \rceil+1..n]})
\end{align*}

\text{Return } \left( \max \{l_g, r_g, \text{max}_R - \text{min}_G, \ldots \text{maj} G \text{ min}_R \text{ min}_G \ldots \text{maj} \text{ max}_G \text{ max}_R \text{ max}_G \ldots \right)

\text{returns } (\text{best trade}, \text{min}, \text{max}) \text{ max of } A

T(n) = 2T(\frac{n}{2}) + \Theta(1)
second attempt

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

base case if \(|A| \leq 2\), ...

\[(lg, minl, maxl) = \text{arbit}(\text{left}(A))\]
\[(rg, minr, maxr) = \text{arbit}(\text{right}(A))\]

return \( \max\{\maxr - \minl, lg, rg\}, \min\{minl, minr\}, \max\{maxl, maxr\} \)
\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix} \star \begin{bmatrix}
5 & 6 \\
7 & 8 \\
\end{bmatrix} =
\]
\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix} \star \begin{bmatrix}
5 & 6 \\
7 & 8 \\
\end{bmatrix} = \begin{bmatrix}
5 + 14 & 6 + 16 \\
15 + 28 & 18 + 32 \\
\end{bmatrix} = \begin{bmatrix}
19 & 22 \\
43 & 50 \\
\end{bmatrix}
\]
\[
\begin{bmatrix}
  a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
  a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{n,1} & a_{n,2} & \cdots & a_{n,n}
\end{bmatrix}
\begin{bmatrix}
  b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\
  b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{n,1} & b_{n,2} & \cdots & b_{n,n}
\end{bmatrix}
= \begin{bmatrix}
  c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\
  c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_{n,1} & c_{n,2} & \cdots & c_{n,n}
\end{bmatrix}
\]
\[ a_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j} \]
\[
\begin{bmatrix}
a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n,1} & a_{n,2} & \cdots & a_{n,n}
\end{bmatrix}
\begin{bmatrix}
b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\
b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
b_{n,1} & b_{n,2} & \cdots & b_{n,n}
\end{bmatrix}
= 
\begin{bmatrix}
c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\
c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
c_{n,1} & c_{n,2} & \cdots & c_{n,n}
\end{bmatrix}
\]
\[
\begin{bmatrix}
A & B \\
C & D \\
\end{bmatrix}
\begin{bmatrix}
E & F \\
G & H \\
\end{bmatrix}
\]
\[
\begin{bmatrix}
A & B \\
C & D \\
\end{bmatrix}
\begin{bmatrix}
E & F \\
G & H \\
\end{bmatrix}
= 
\begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH \\
\end{bmatrix}
\]
\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\begin{bmatrix}
E & F \\
G & H
\end{bmatrix}
= 
\begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH
\end{bmatrix}
\]

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

\[\Theta(n^3)\]
\[
= \begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH
\end{bmatrix}
\]

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

\[
T = P_3 + P_4
\]

\[
U = P_5 + P_1 - P_3 - P_7
\]

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

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

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

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

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

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

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

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

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

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

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

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

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

\[
P_7 = (A - C)(E + F')
\]
\[ R = \begin{bmatrix} \begin{align*} A E + B G & \quad AF + BH & \quad S \\ C E + D G & \quad CF + DH \\ T = P_3 + P_4 & \quad U = P_5 + P_1 - P_3 - P_7 \end{align*} \end{bmatrix} = P_1 + P_2 \]

\[ P_1 = A(F - H) \]
\[ P_2 = (A + B)H \]
\[ M(n) = 7M(n/2) + 18n^2 \]
\[ = \Theta(n^{\log_2 7}) \]
\[ P_3 = (C + D)E \]
\[ P_4 = D(G - E) \]
\[ P_5 = (A + D)(E + H) \]
\[ P_6 = (B - D)(G + H) \]
\[ P_7 = (A - C)(E + F) \]
taking this idea further

3x3 matrices
1978 victor pan method

70x70 matrix using 143640
mul.ts

what is the recurrence:
Median
problem: given a list of \( n \) elements, find the element of rank \( n/2 \). (half are larger, half are smaller)
Problem: given a list of $n$ elements, find the element of rank $n/2$. (half are larger, half are smaller)

Can generalize to $i$

First solution: sort and pluck.

$O(n \log n)$
problem: given a list of n elements, find the element of rank i.
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
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 \((i, A[1, \ldots, n])\)
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 \(((i - p - 1), A[p + 1, \ldots, n])\)
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 $> i$
Else
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?
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 \(((i - p - 1), A[p + 1, \ldots, n])\)
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 \(((i - p - 1), A[p + 1, \ldots, n])\)

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

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

a good partition element

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

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

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

\[ P(n) = S\left(\lceil n/5 \rceil\right) + O(n) \]
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

\[
3 \left( \left\lfloor \frac{1}{2} \left\lfloor \frac{n}{5} \right\rfloor \right\rfloor - 2 \right) \\
\geq \frac{3n}{10} - 6
\]

this implies there are at most \( \frac{7n}{10} + 6 \) numbers larger than smaller

larger than \( \star \) /smaller

\[
\frac{3n}{10} - 6
\]
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
else
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 = \(\text{FindPartitionValue}(A, n)\)
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 \(((i-p-1), A[p+1, \ldots, n])\)

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

- handle base case for small list
- else pivot = \text{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 \(((i - p - 1), A[p + 1, \ldots, n])\)

\[
S(n) = S(\lfloor n/5 \rfloor) + O(n) + S(7n/10 + 6)
\]

\(\Theta(n)\)
arbitrage
input: array of $n$ numbers

1     ....     n

goal:
first attempt
first attempt

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

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

\texttt{arbit\+(A[1\ldots n])}

\hspace{1em} 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}