L6

4102

feb 9 2016

shelat

divide&conquer

closest points

matrixmult

median

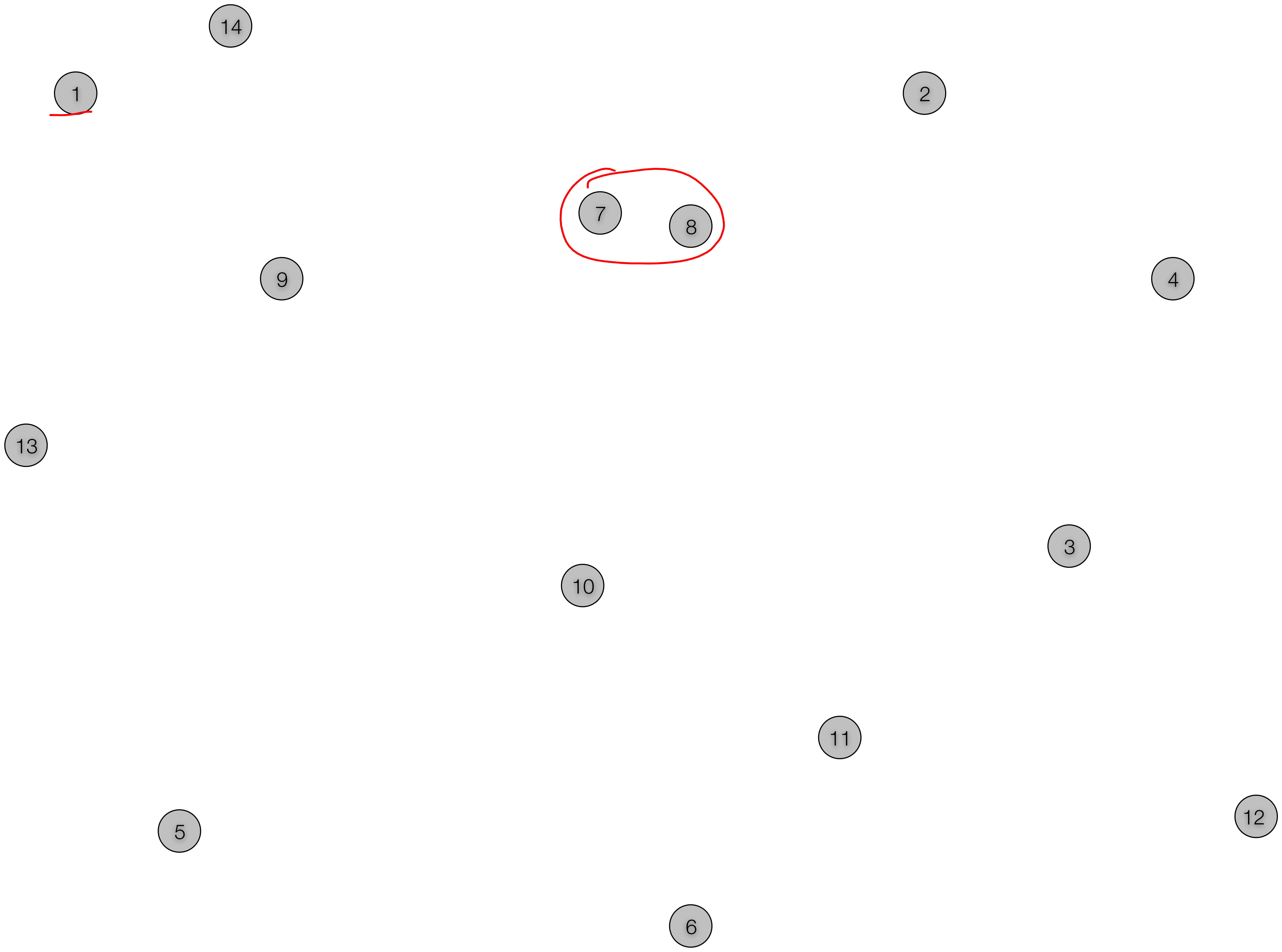
7.5	7.8	9.5	3.7	26.1

closest pair

of points

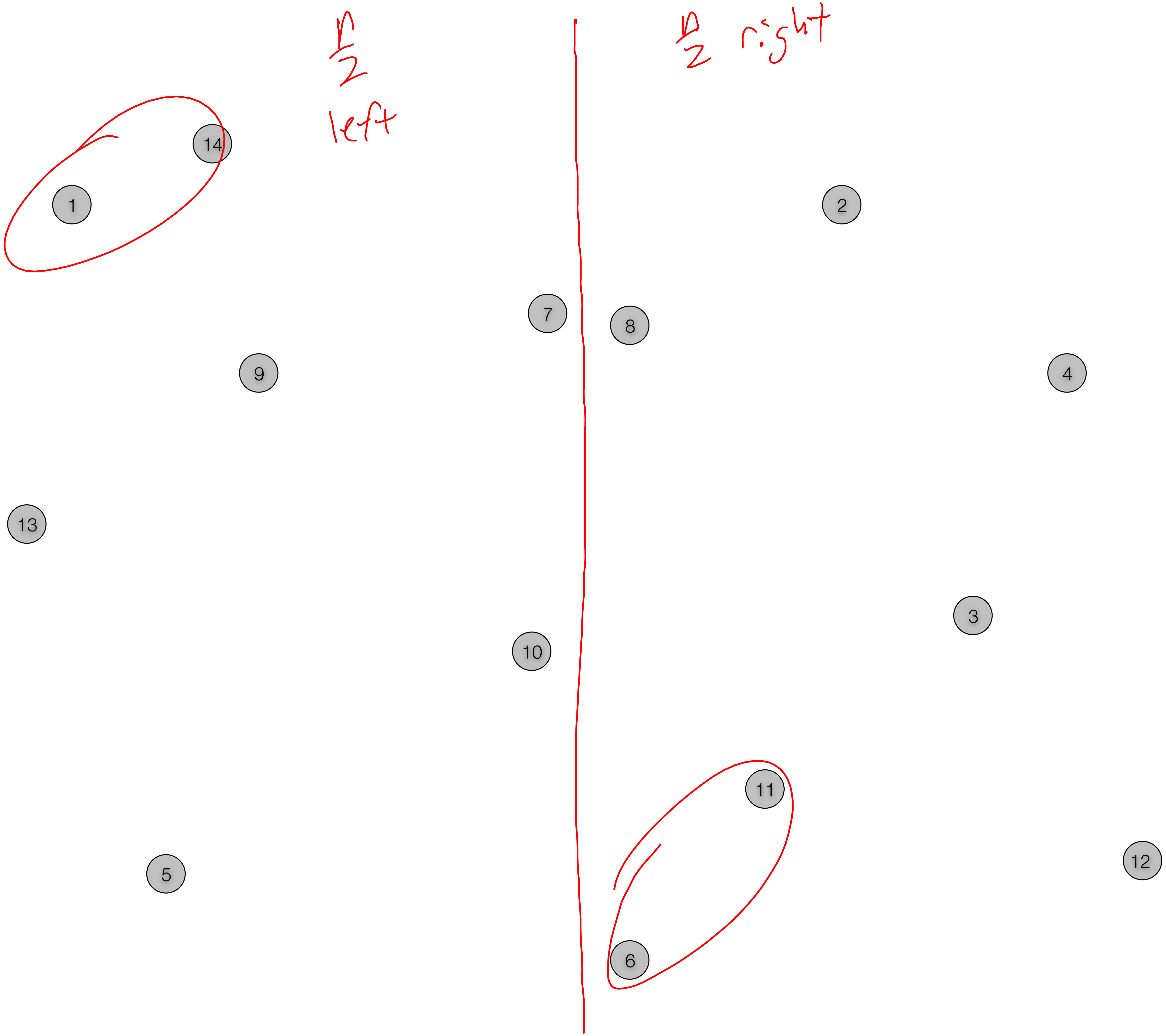
simple brute force approach takes

n points in 2D.

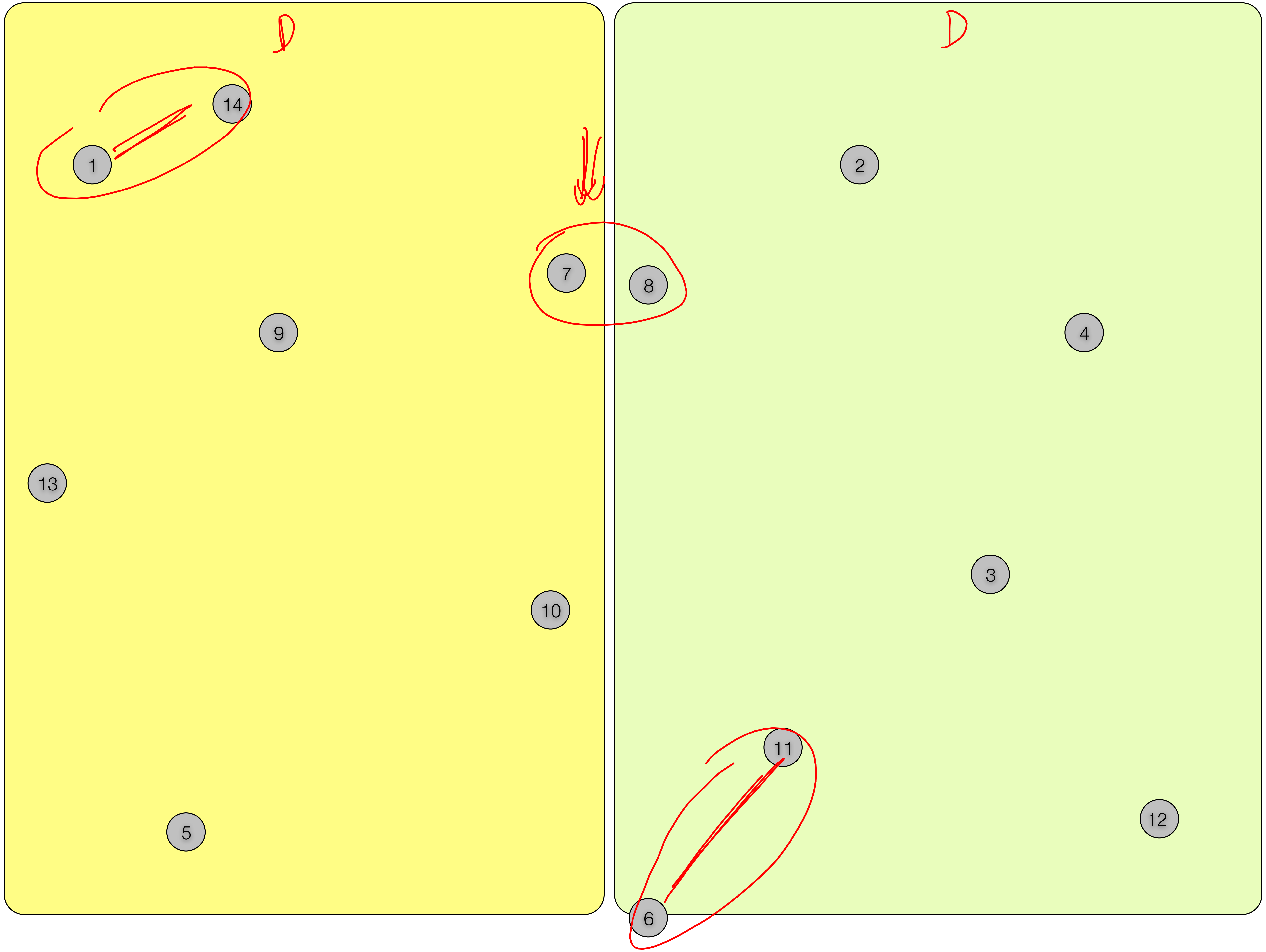


solve the large problem by

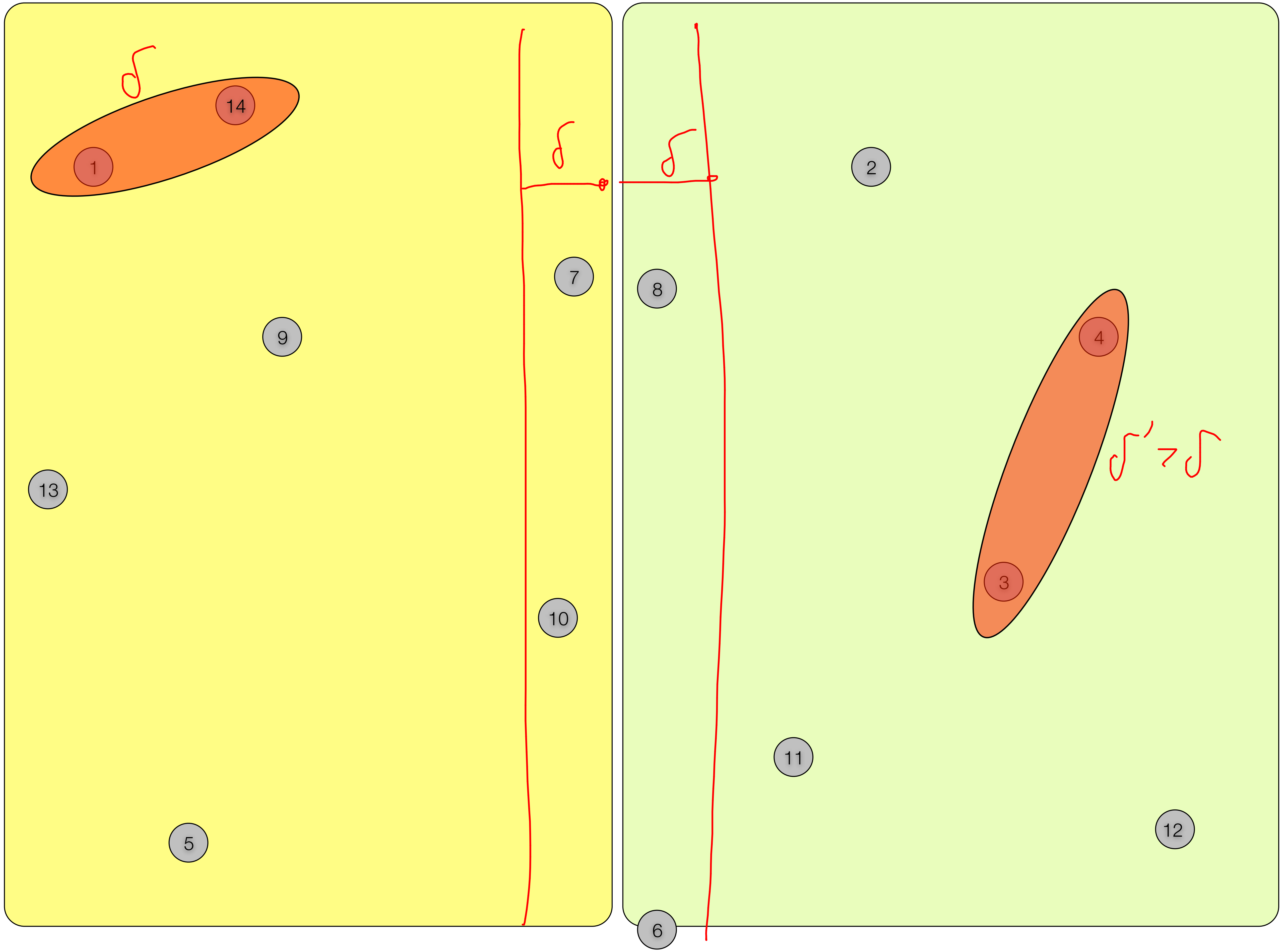
solving smaller problems
and combining solutions


n/2
left
n/2 right
14
1
2
7
8
9
4
13
3
10
11
12
5
6

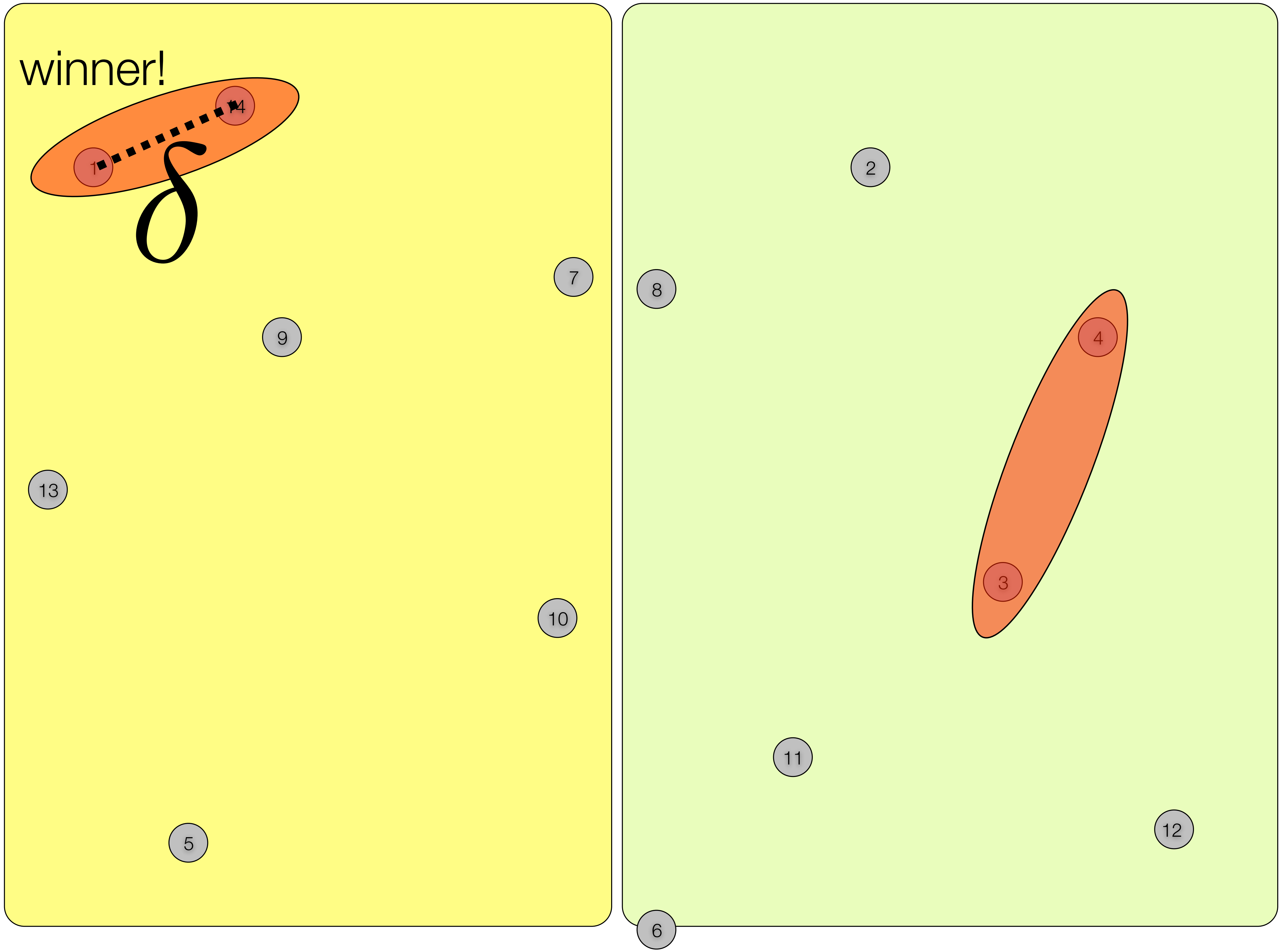
Divide & Conquer



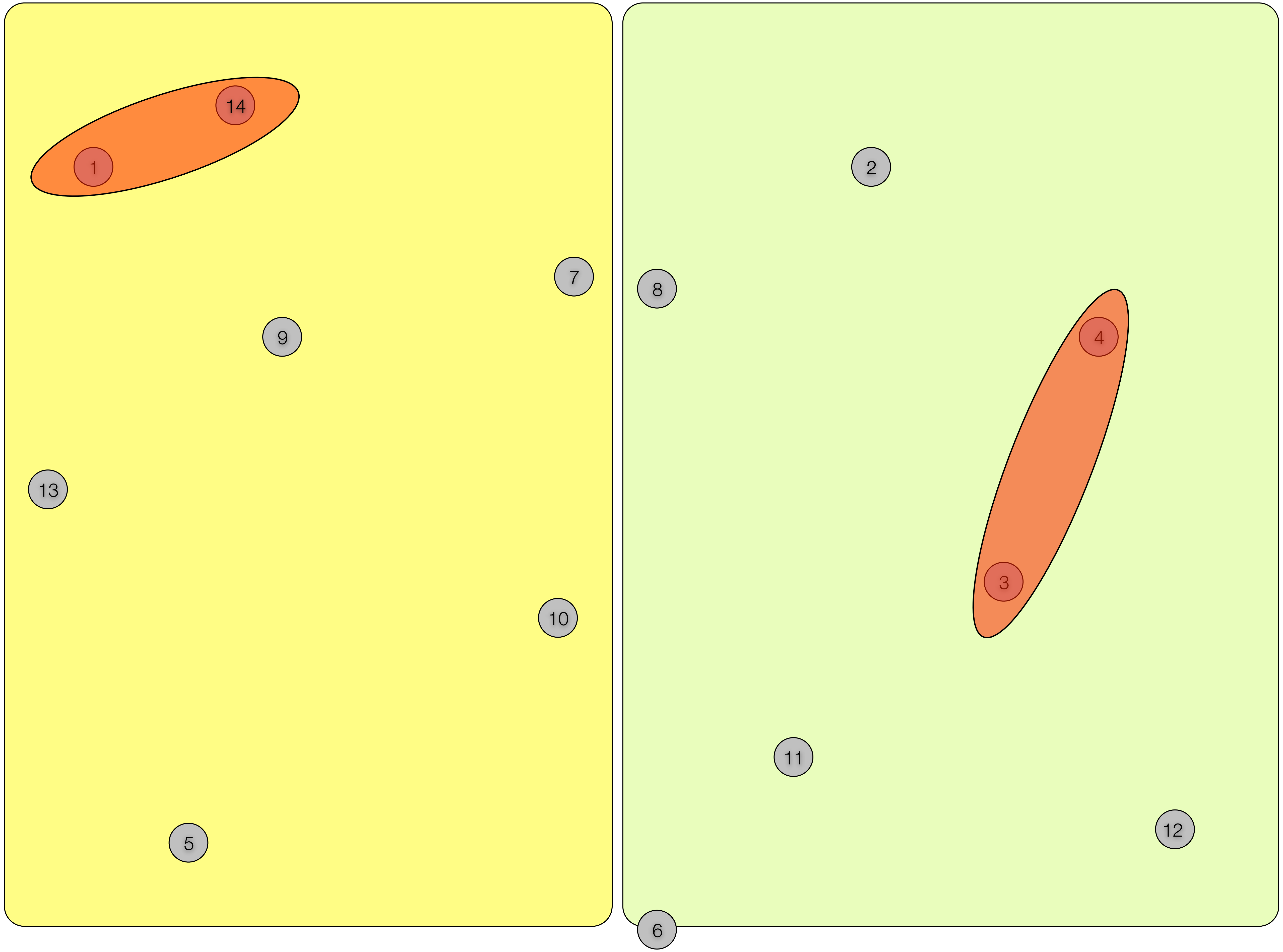
Divide & Conquer



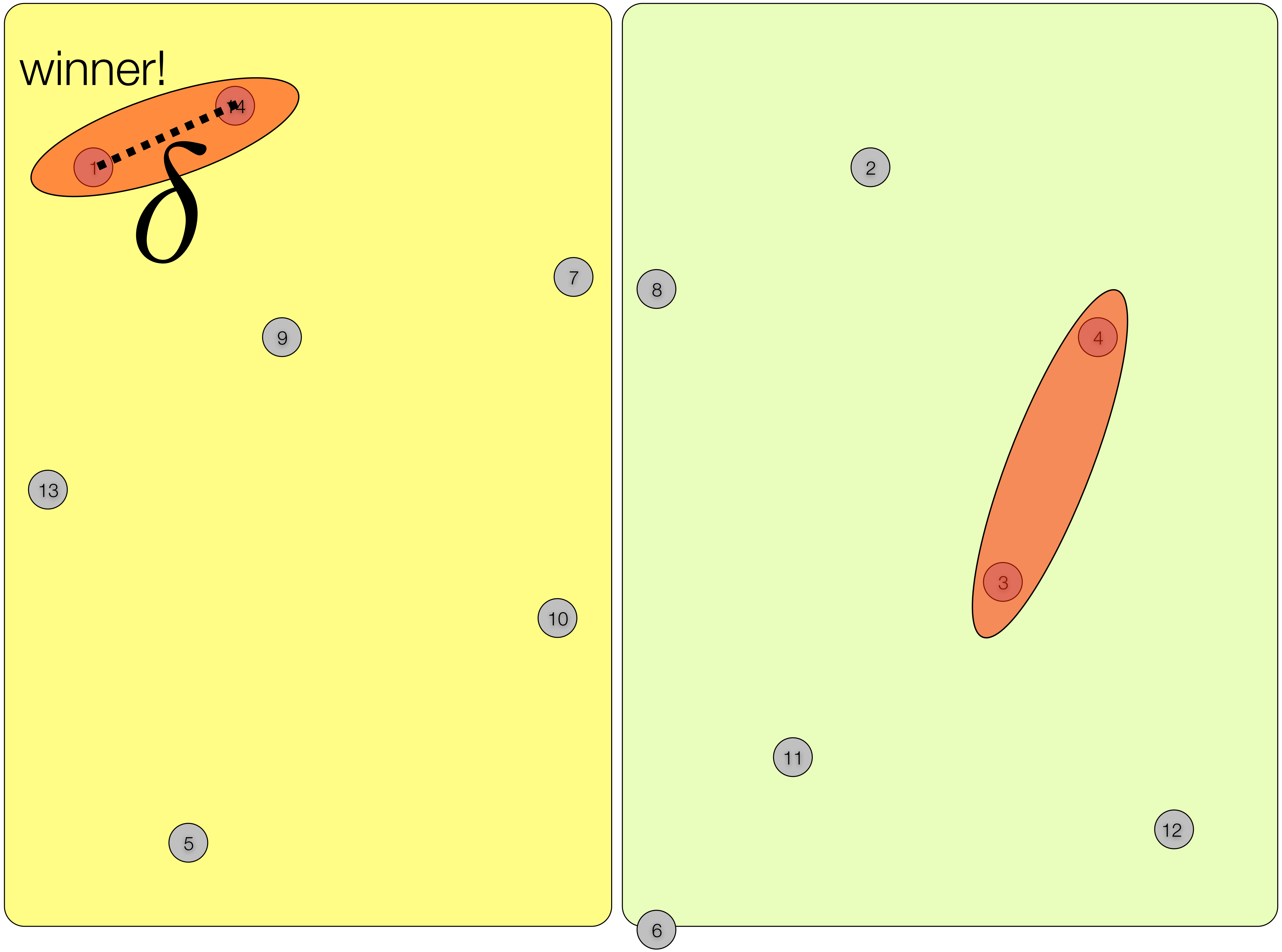
Divide & Conquer

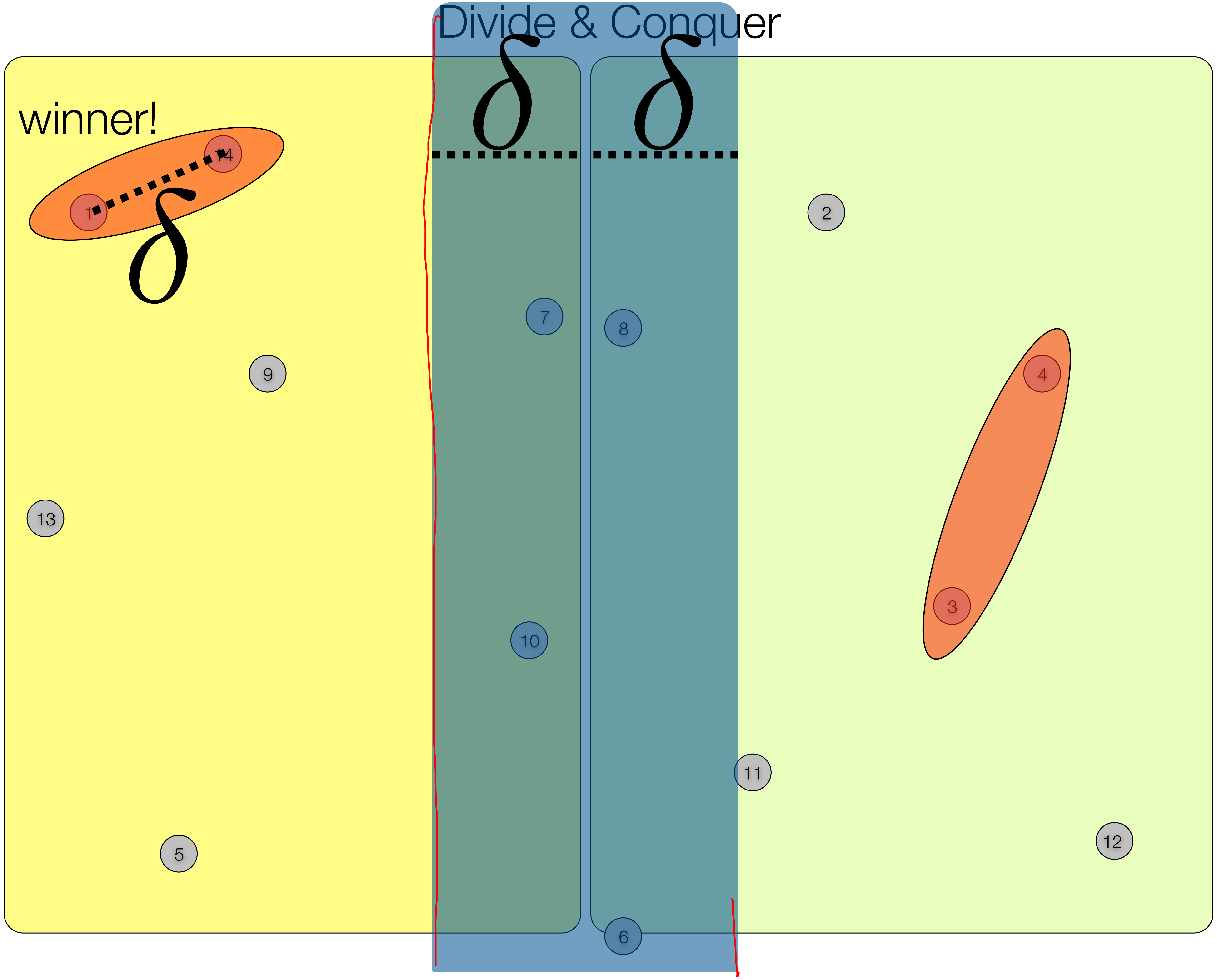


Divide & Conquer

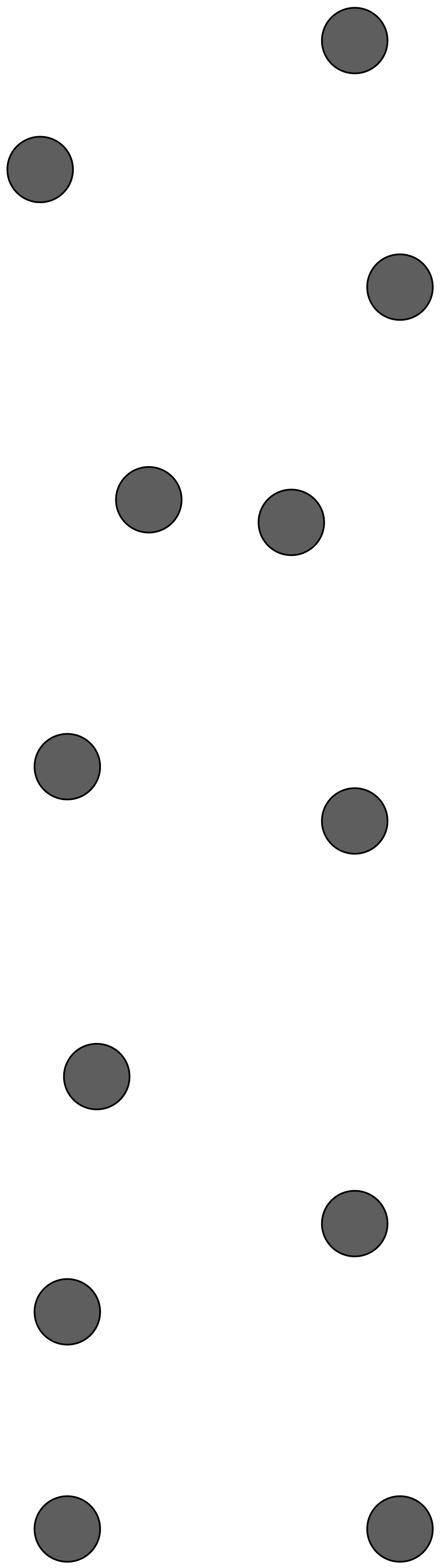


Divide & Conquer

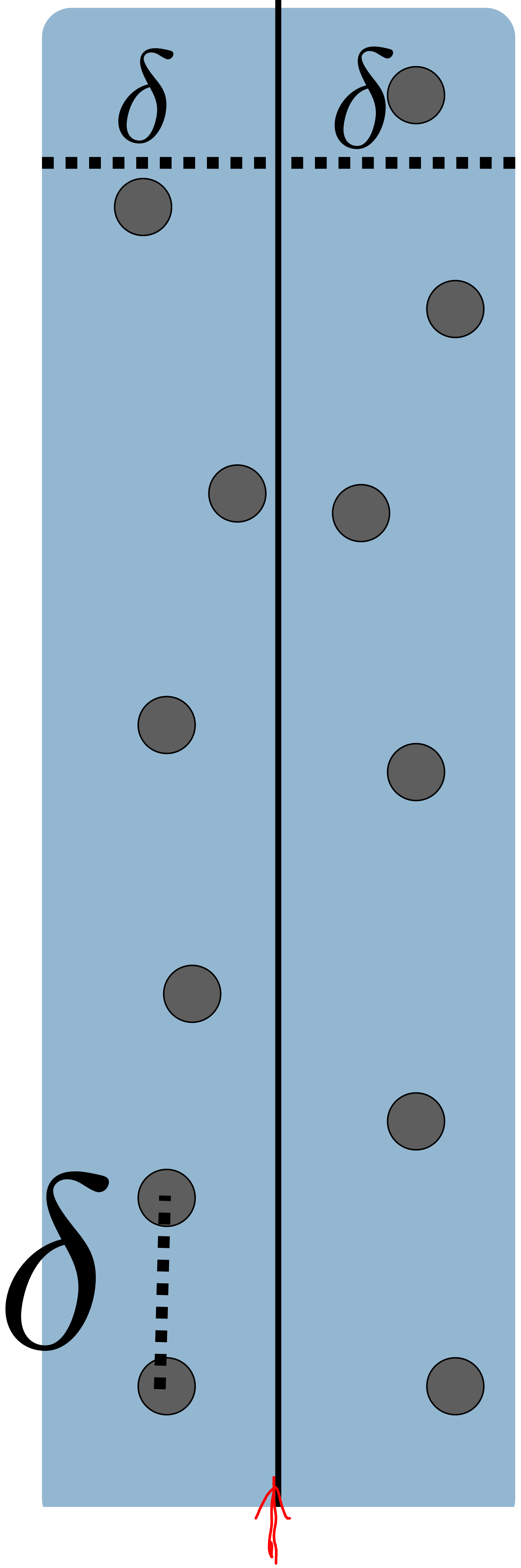


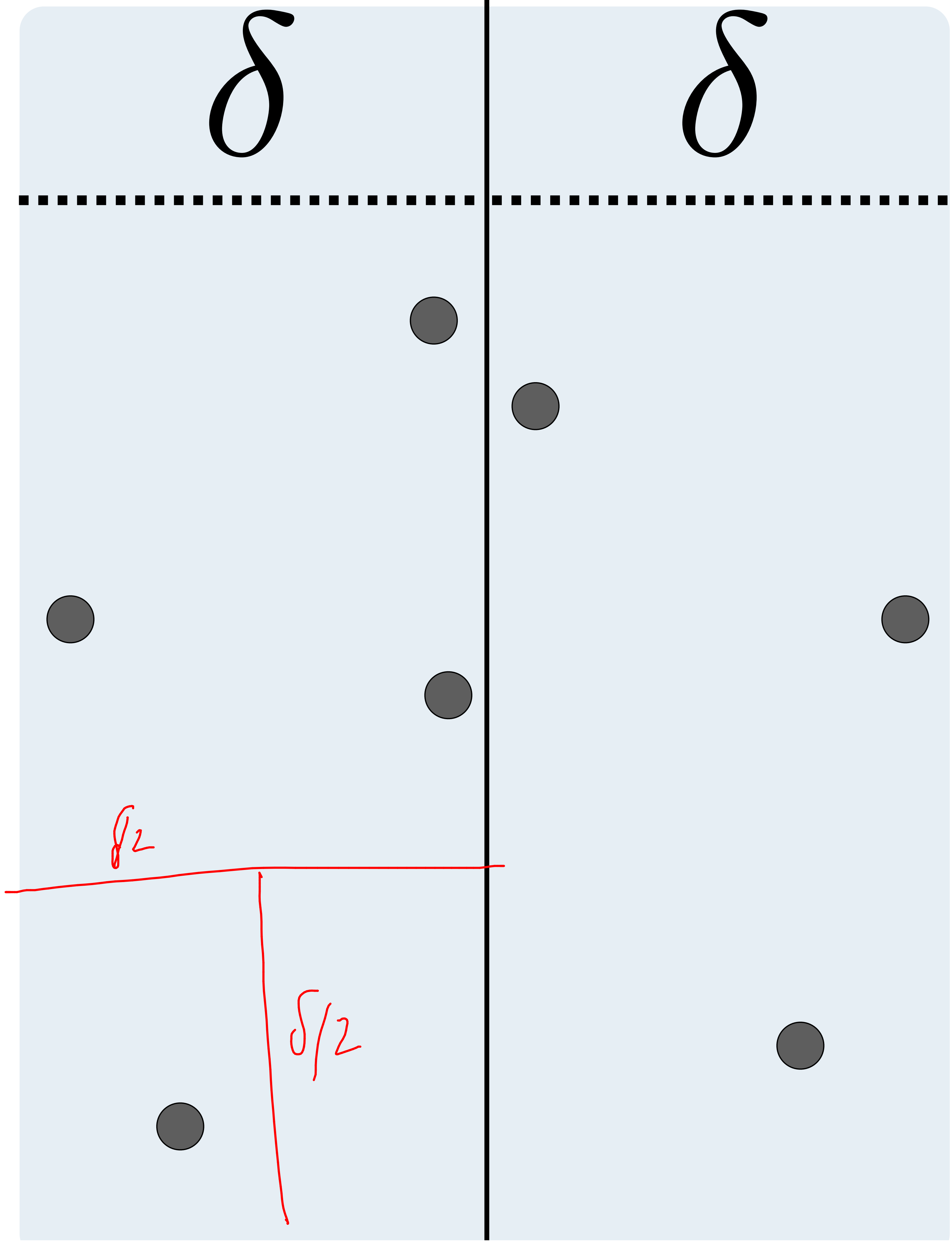


Divide & Conquer
winner!
δ
δ
δ
1
14
2
7
8
9
4
13
3
10
11
12
5
6



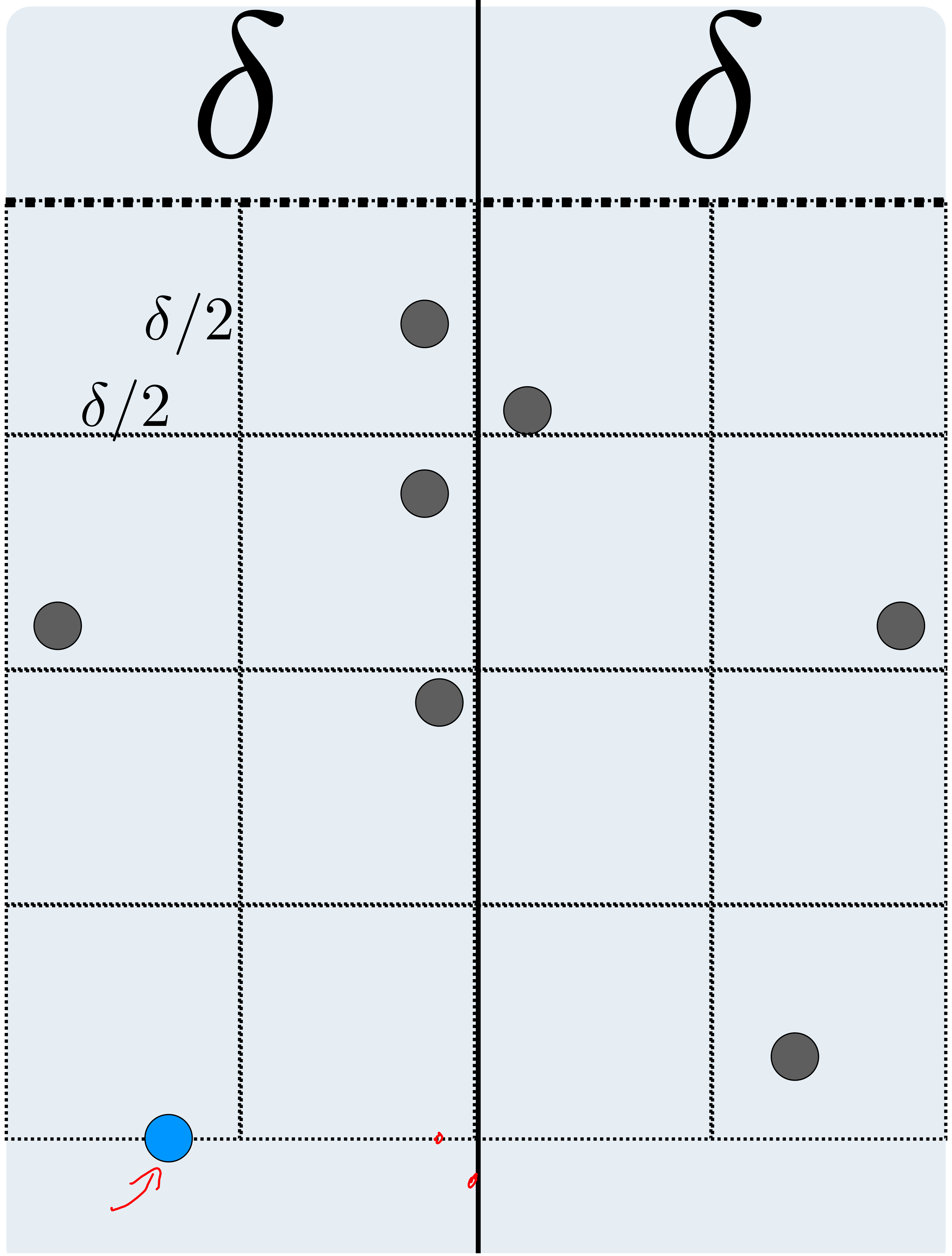
Mohawk can contain all of the input points.

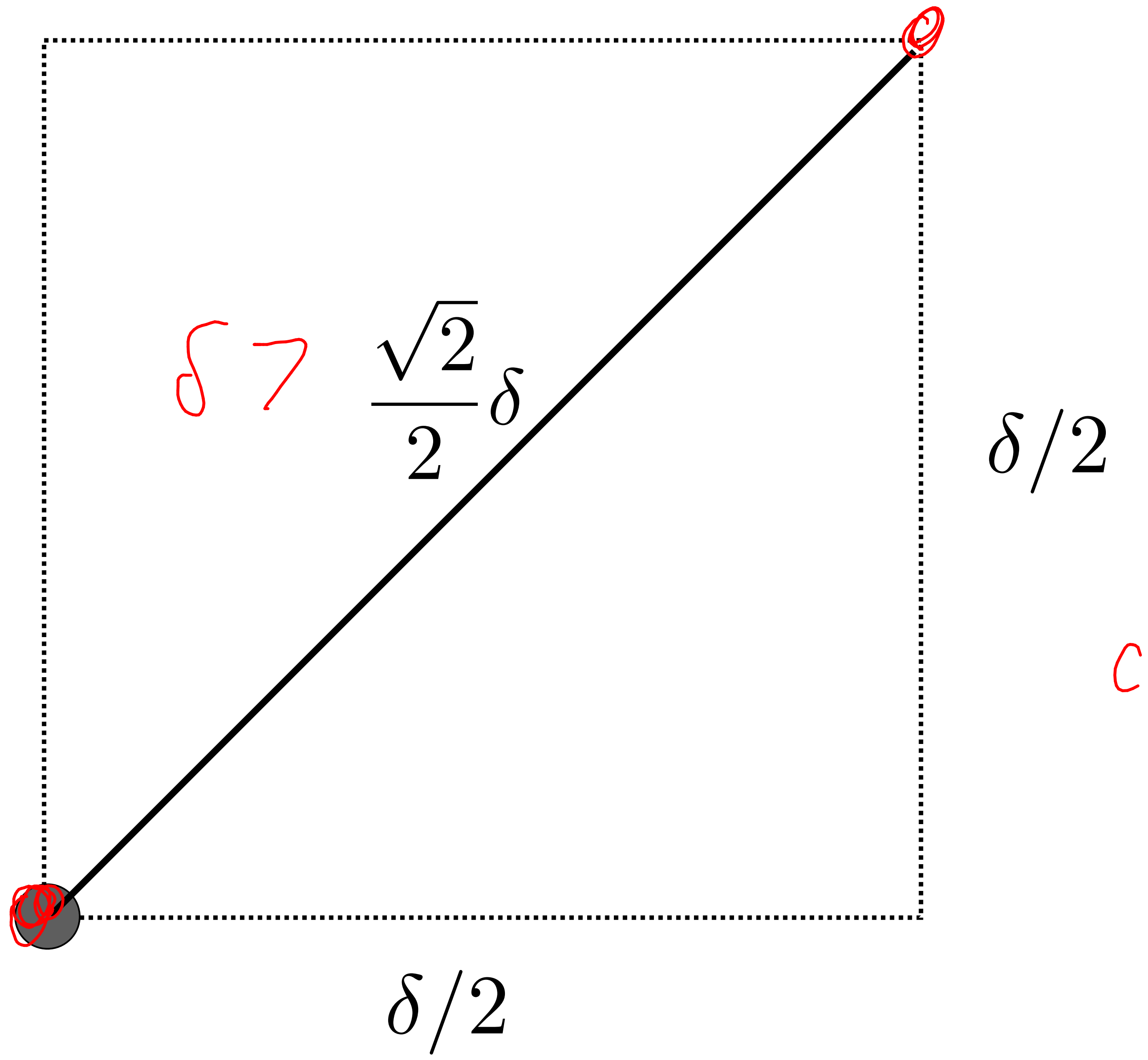

δ
δ
δ



δ
δ
δ/2
δ/2

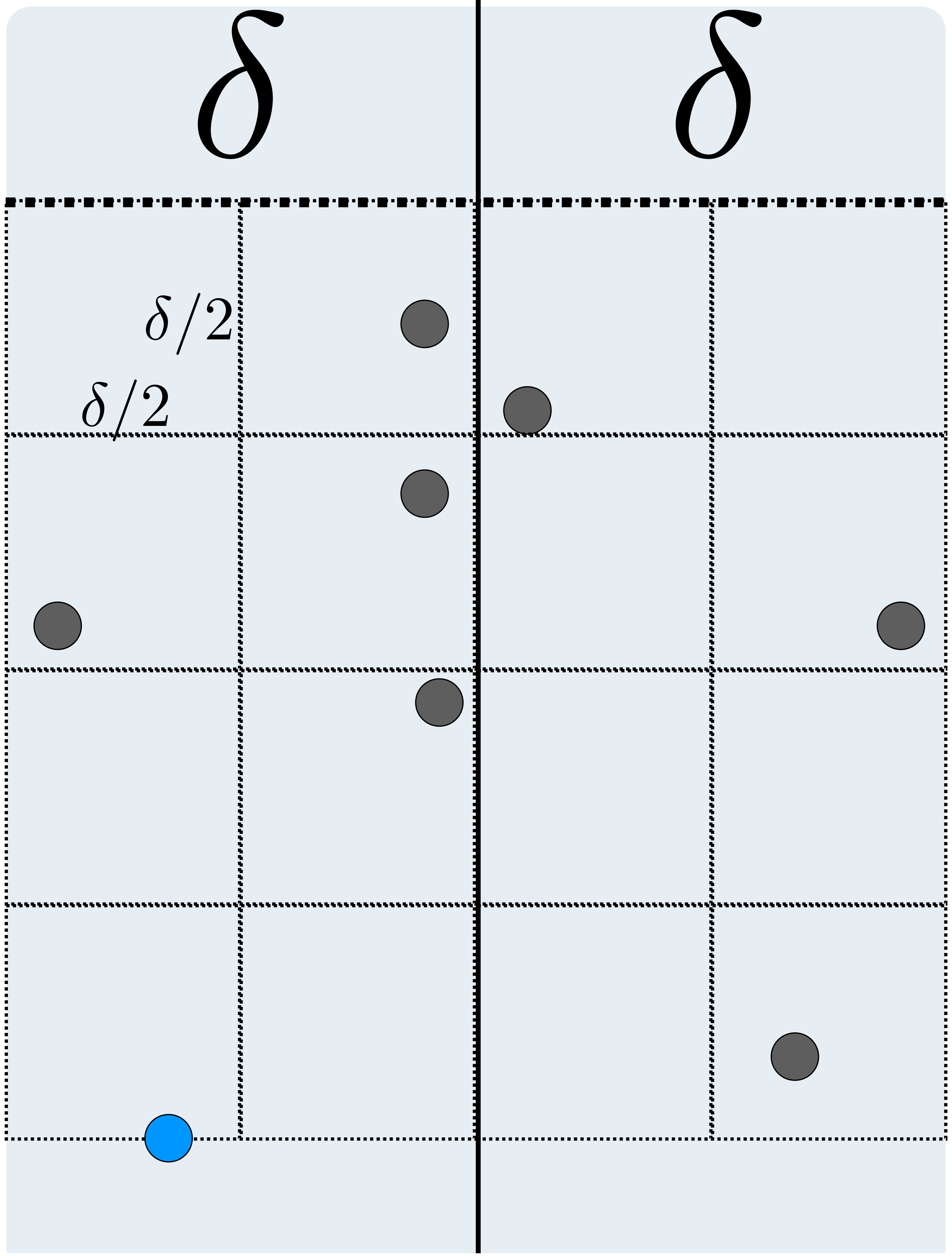
Imagine there is a grid of cubbies starting at the lowest Y point



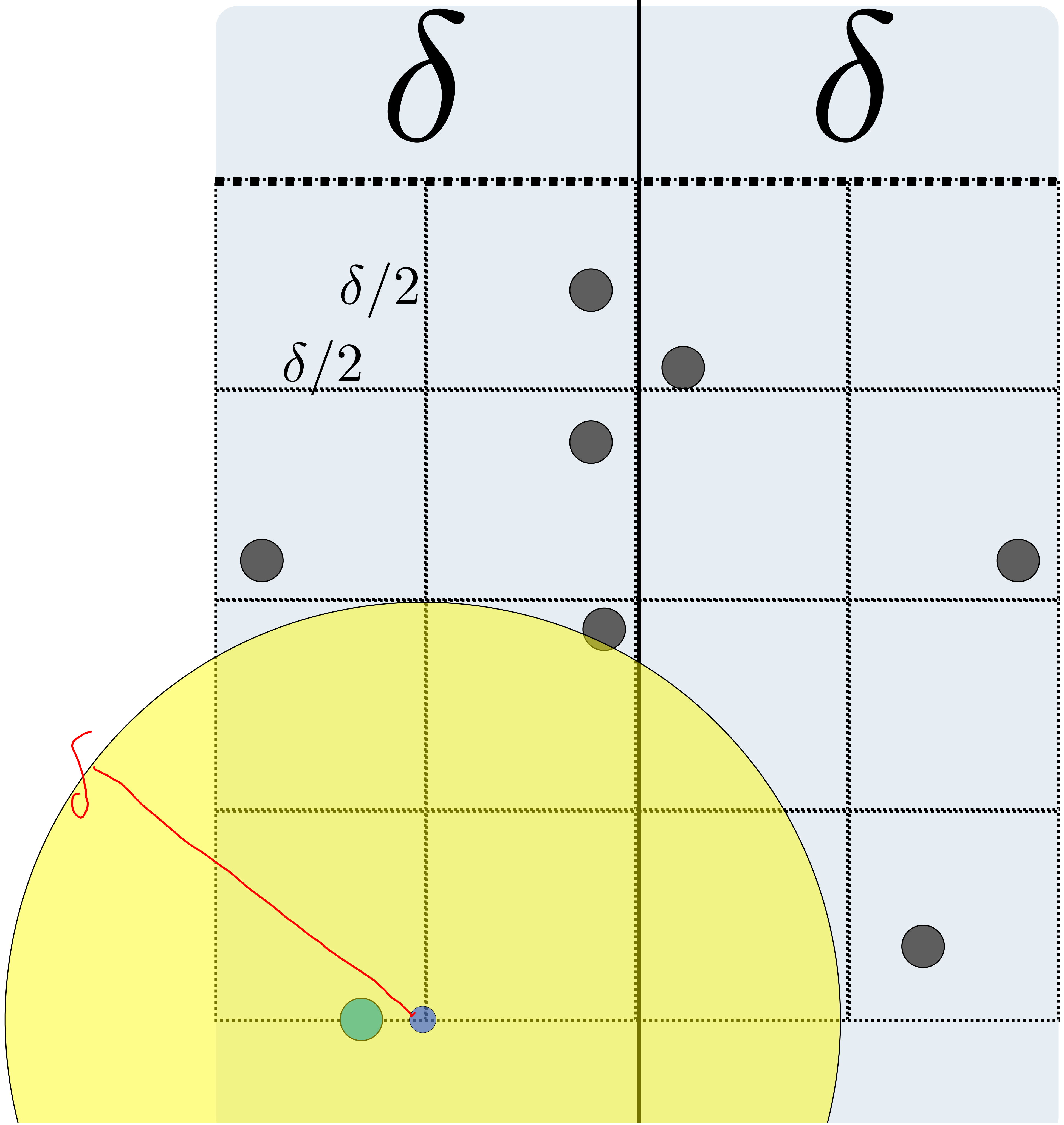


② insight.

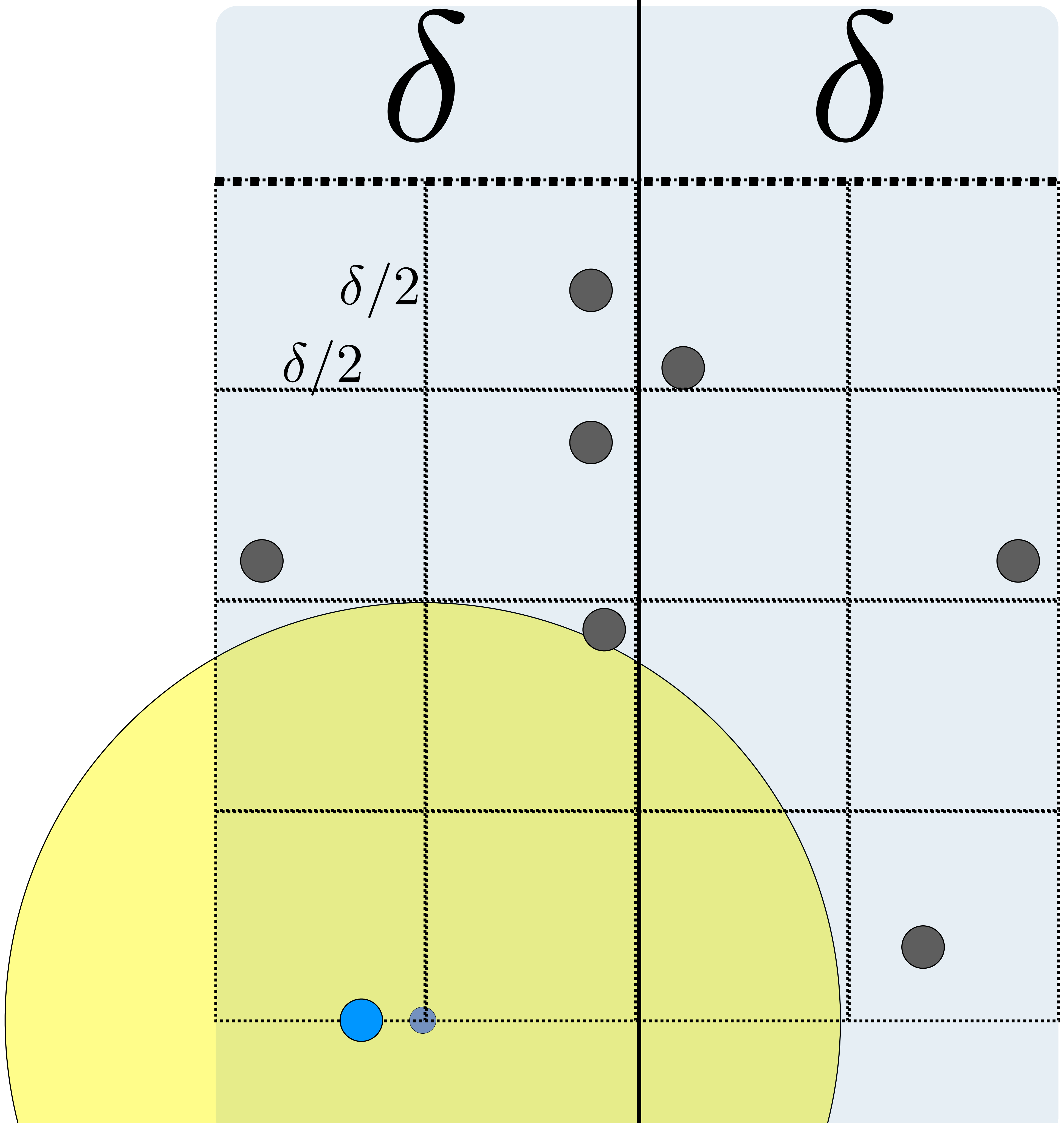
cubbies have ≤ 1 point.



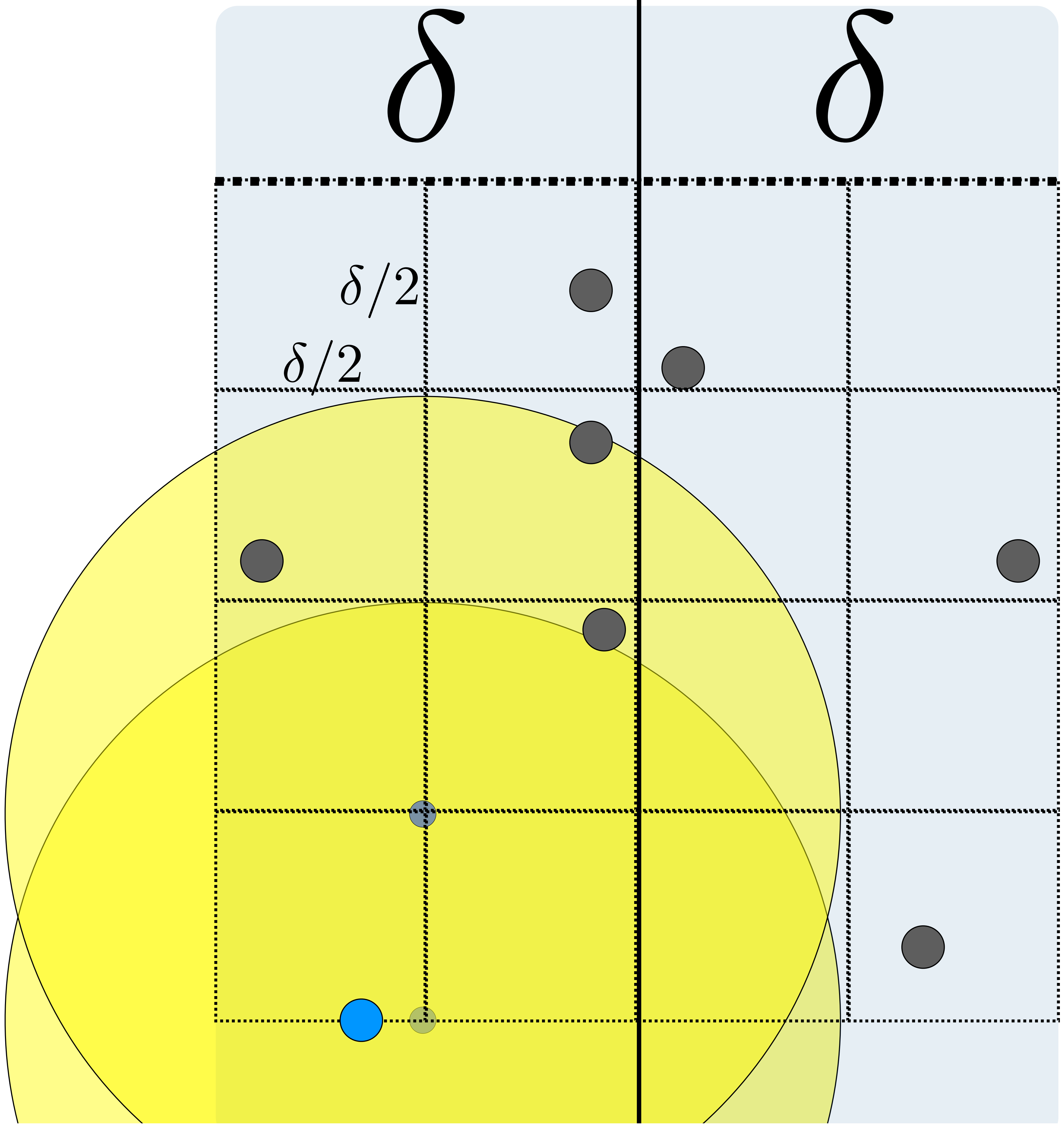
FACT: At most 1 point in each cubby



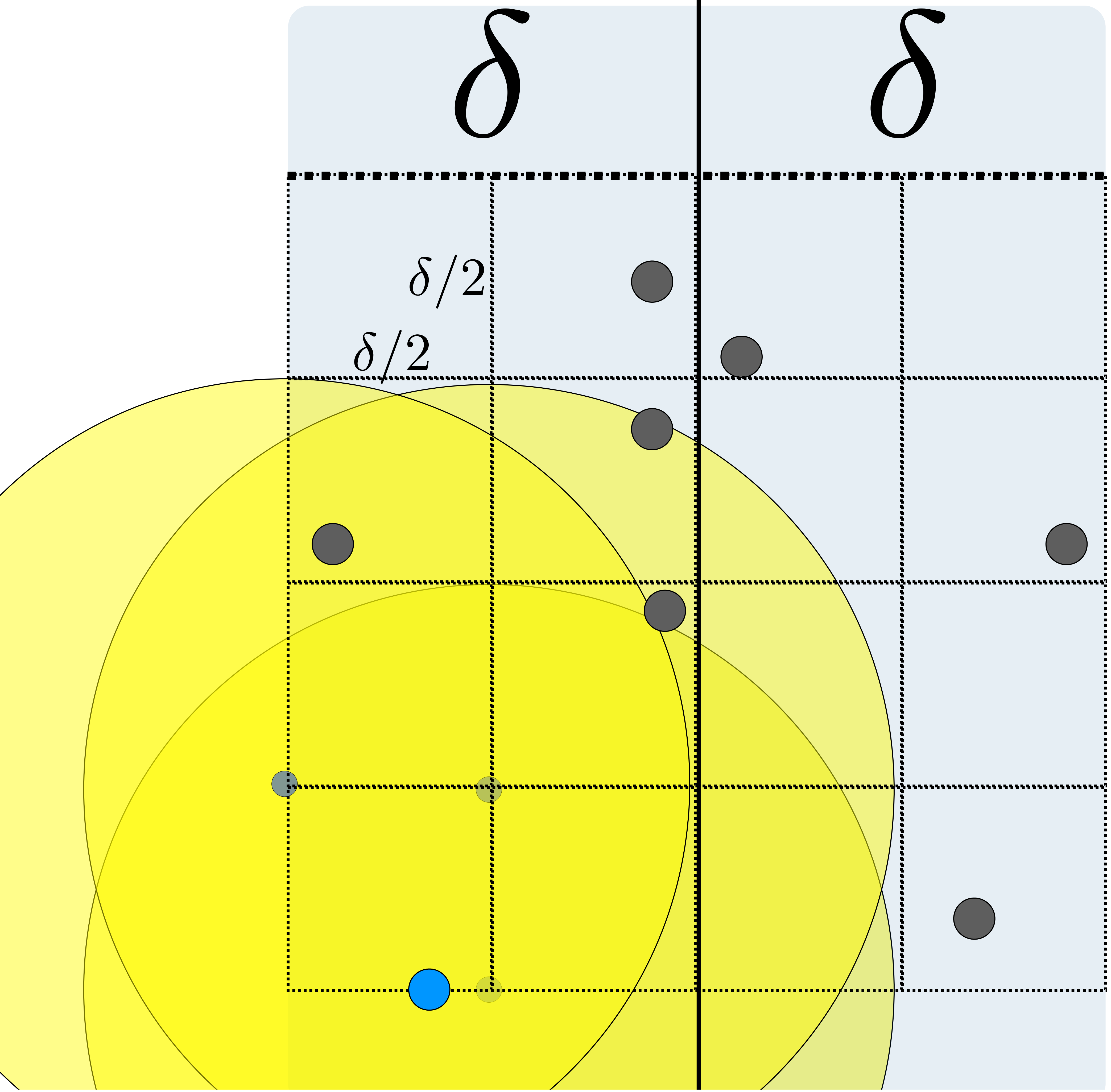
FACT: <=1 point per cubby



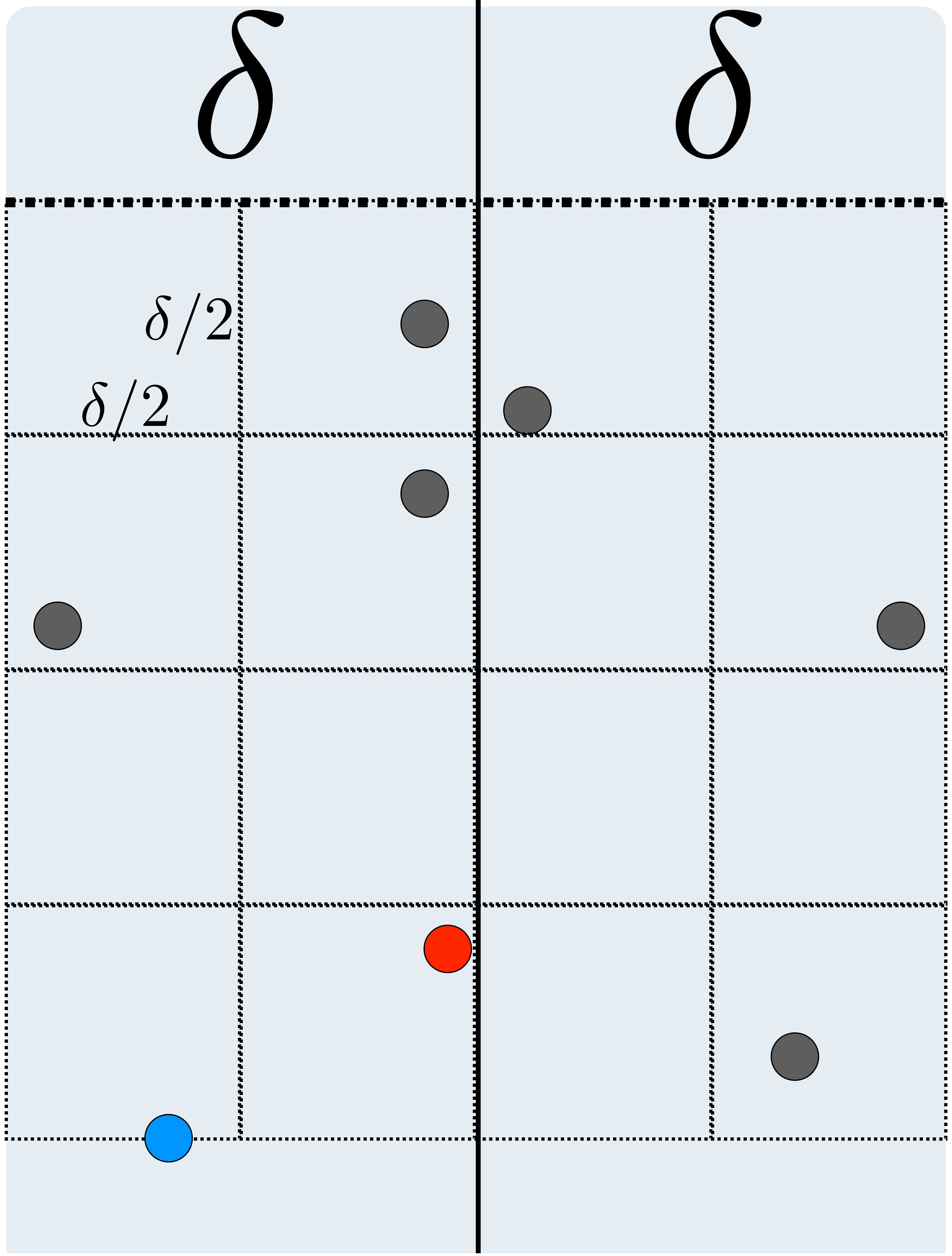
FACT: <=1 point per cubby

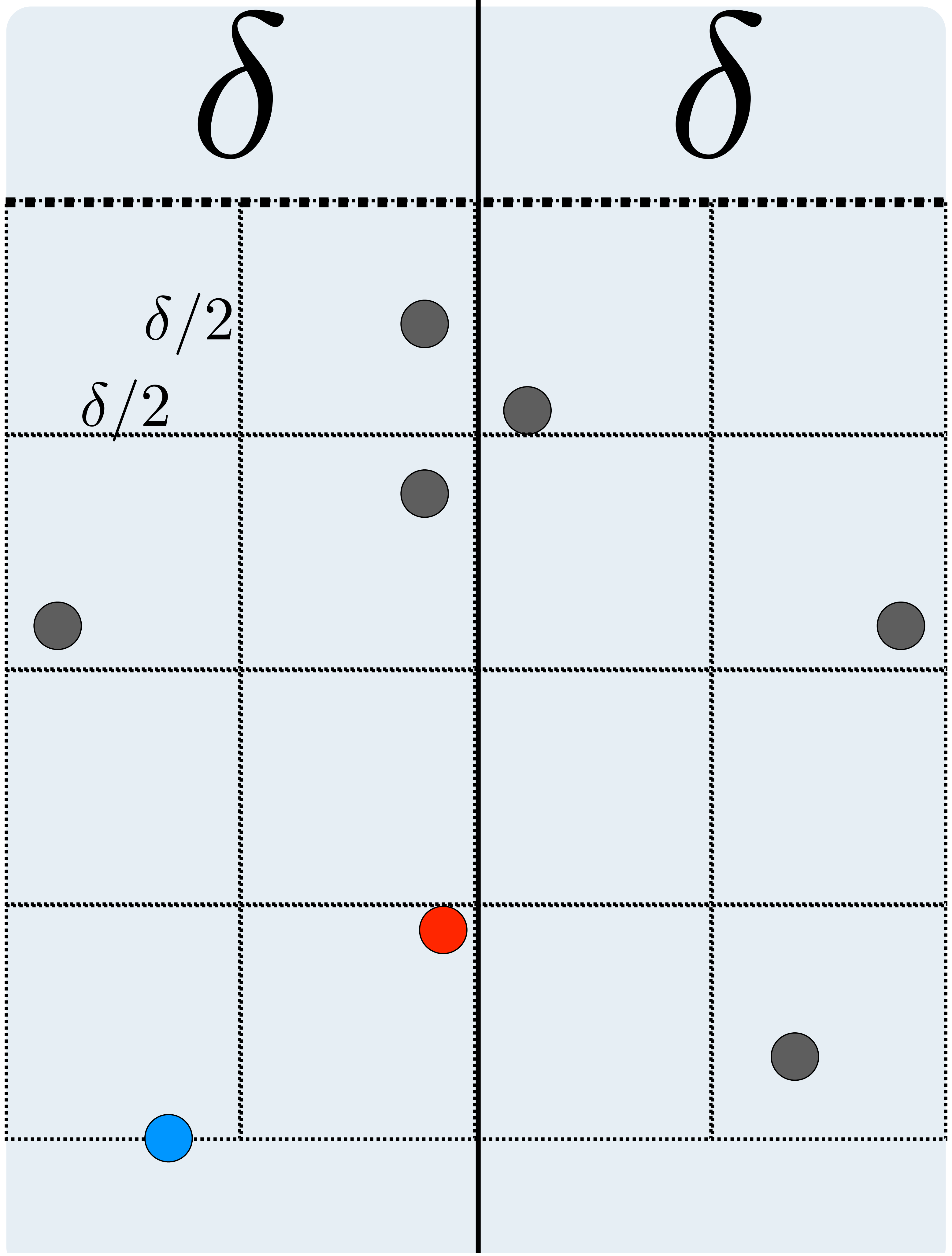


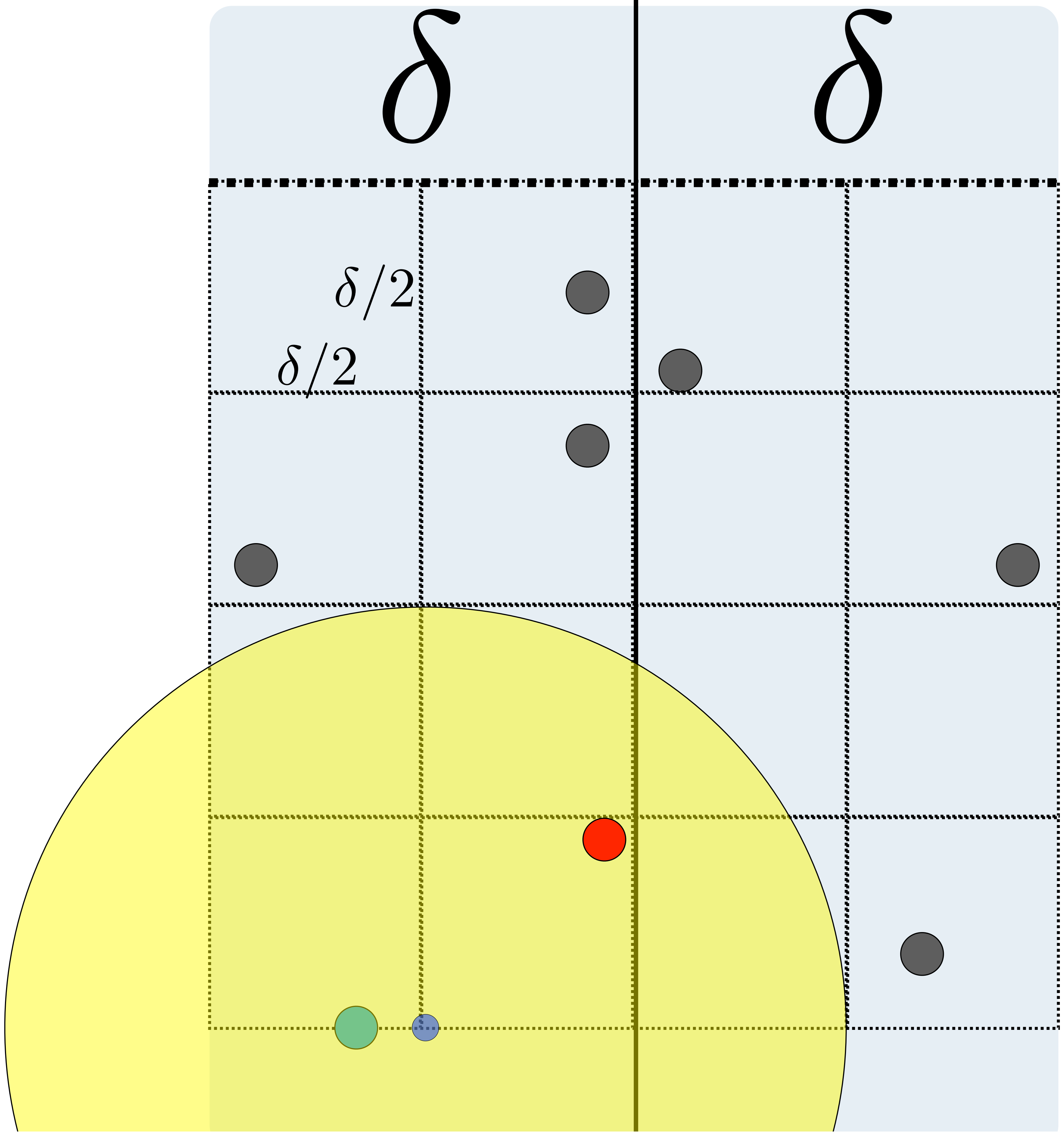
FACT: <=1 point per cubby



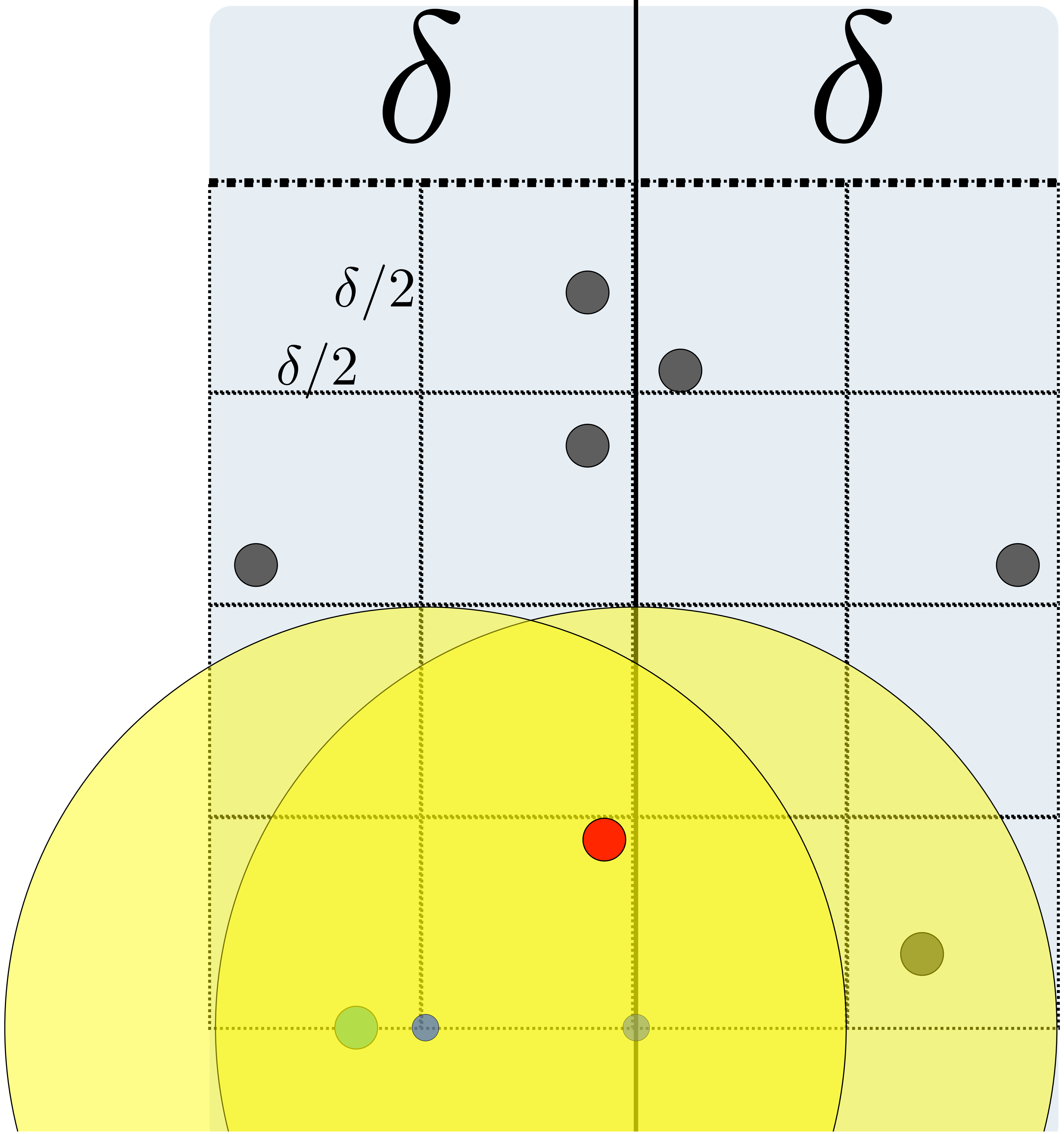
FACT: <=1 point per cubby


δ
δ
$\delta/2$
$\delta/2$

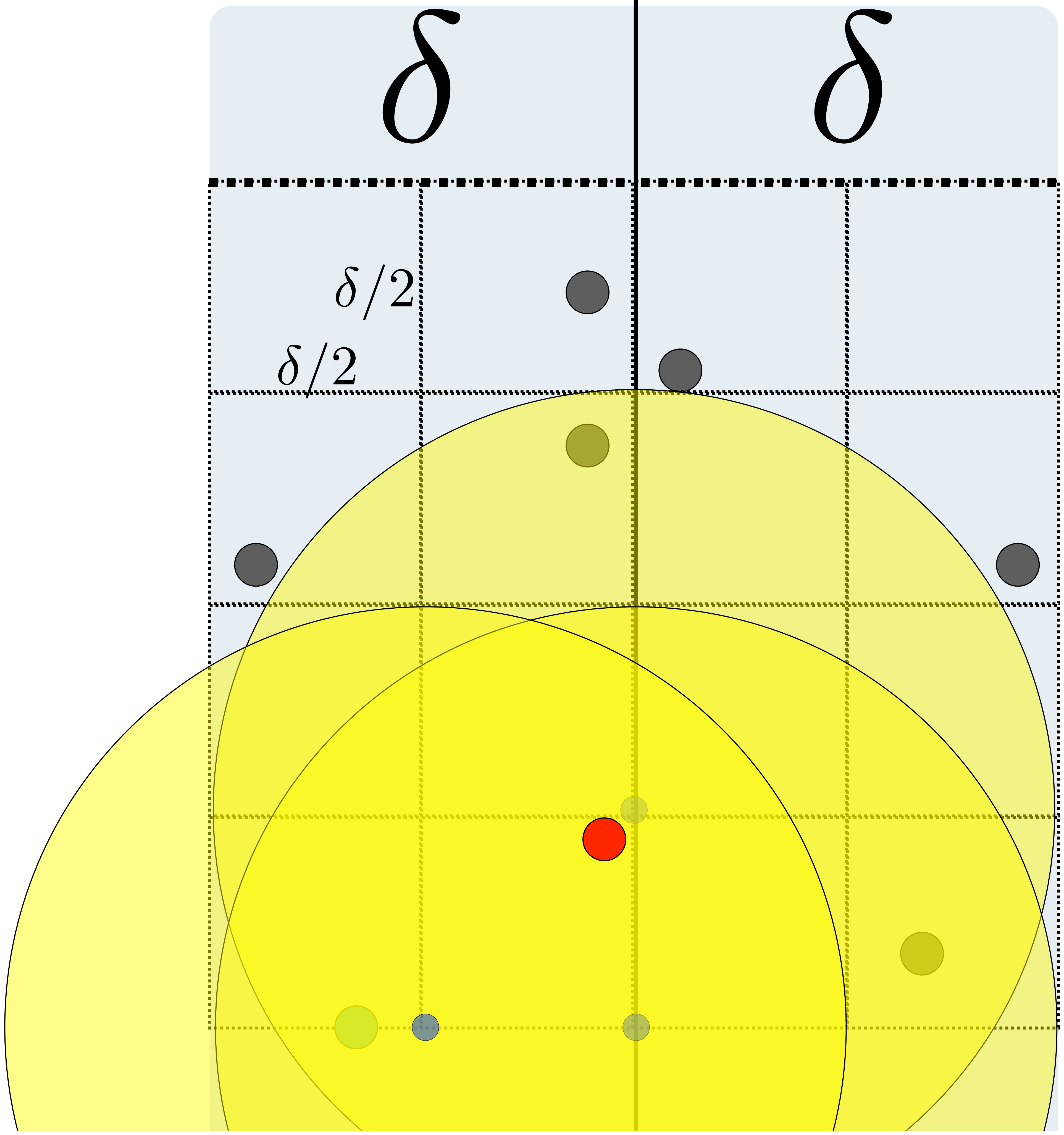

δ
δ
$\delta/2$
$\delta/2$



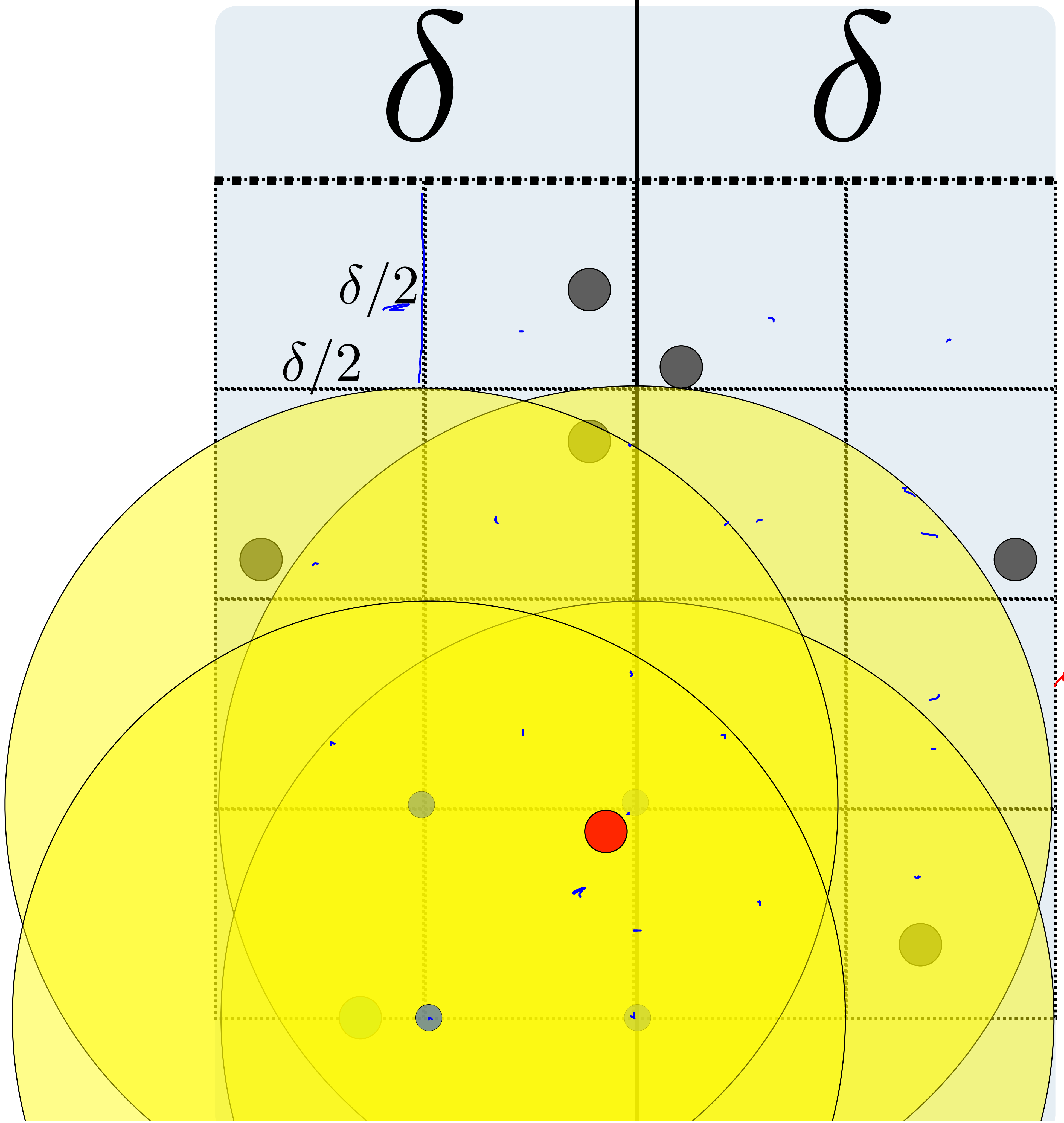
δ
δ
$\delta/2$
$\delta/2$



δ
δ
$\delta/2$
$\delta/2$



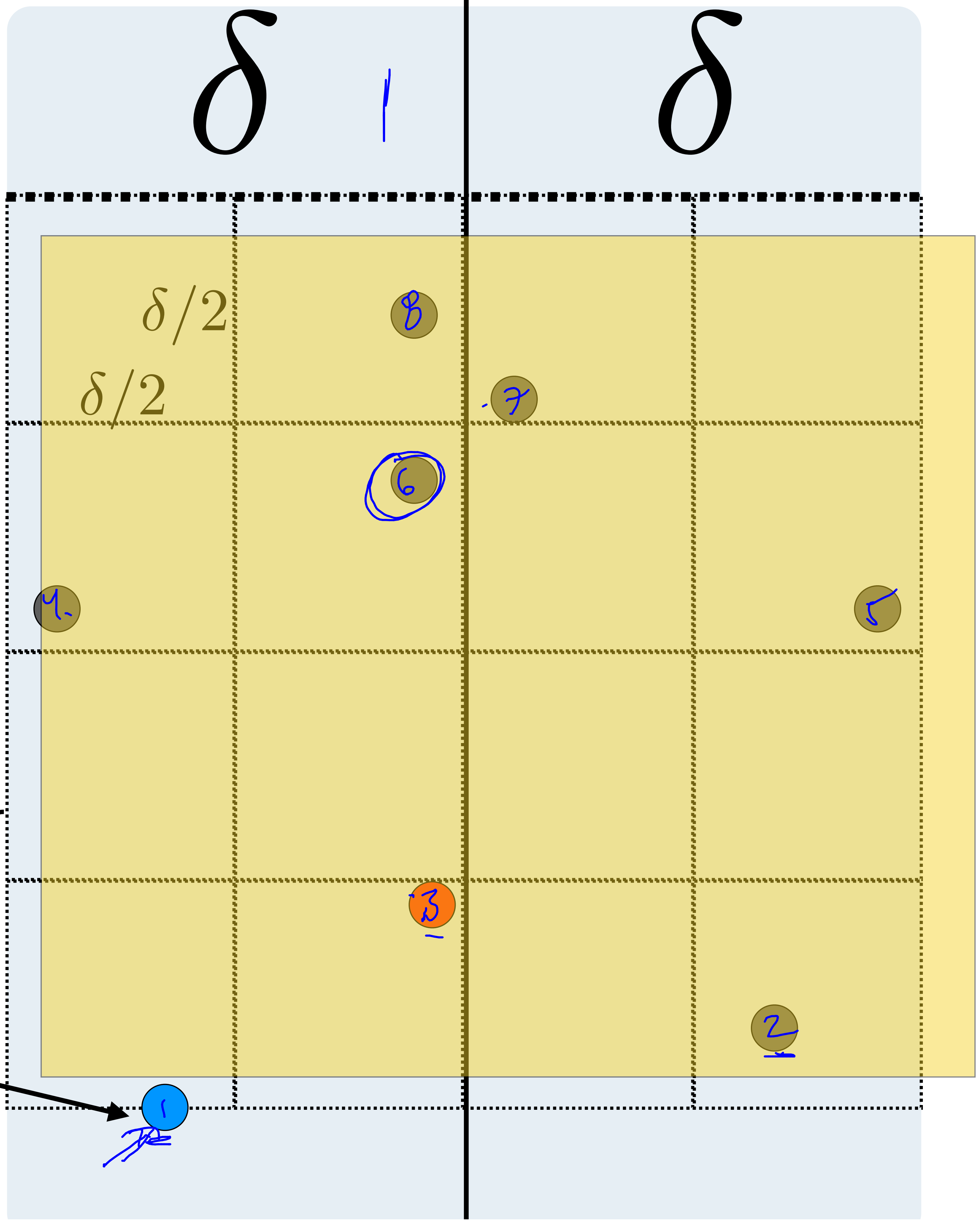
δ
δ
δ/2
δ/2



count the number of cubbies that juliet can reside in.

$\leq 15.$

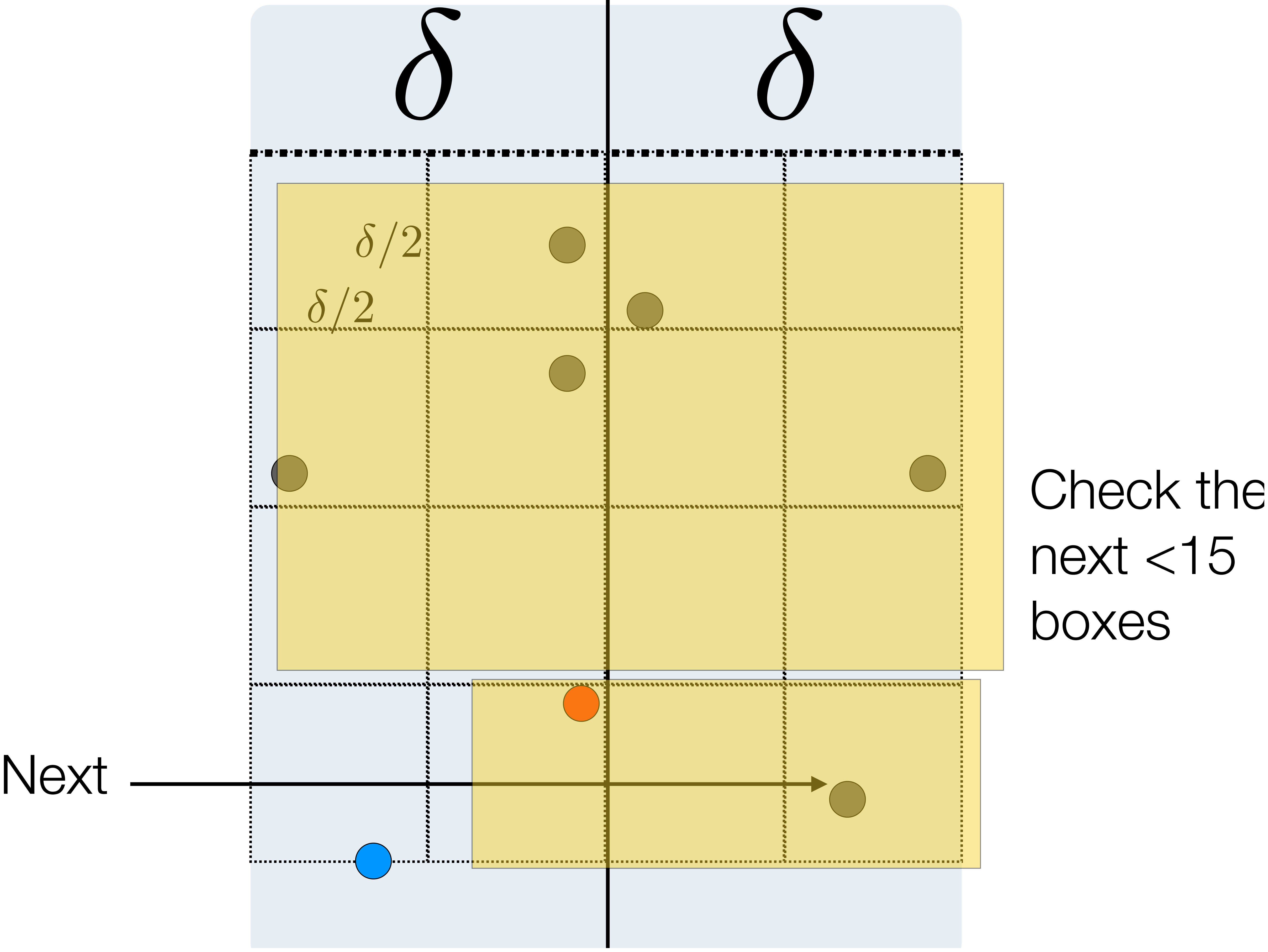
≤ 7 buckets

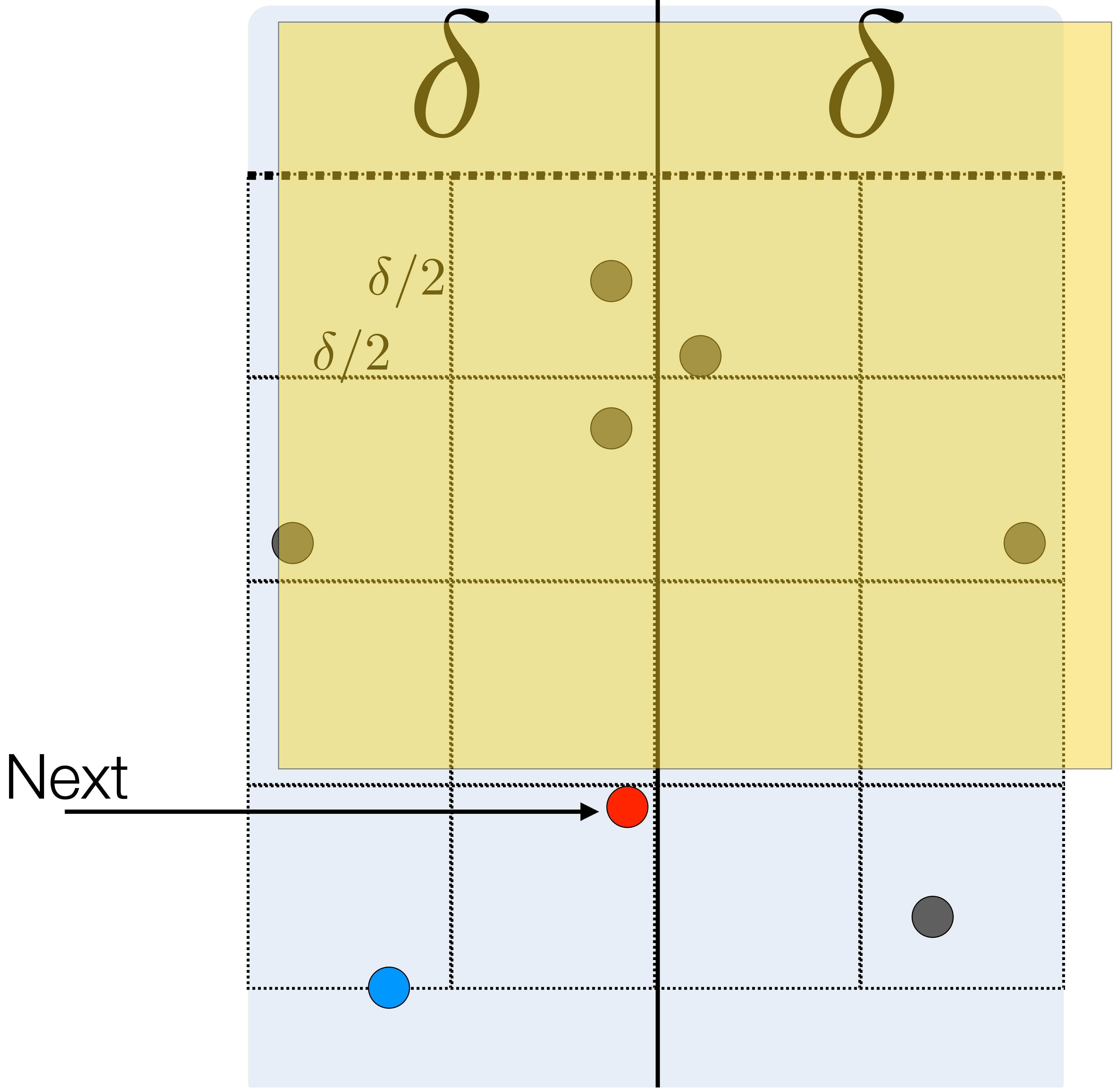


Check the next 15 boxes

Visit its by y-order

Start


δ
δ
$\delta/2$
$\delta/2$
Check the next <15 boxes
Next



Check the next <15 boxes

Closest(P) → points in 2D.

Base case: if $|P| \le 2$, brute force.

Let q be the mid-point along x coordinates.

$L, R =$ split points into left & right halves according to q

$\delta_L = \text{closest}(L)$
$\delta_R = \text{closest}(R)$
$\Longrightarrow$ let $\delta = \min(\delta_L, \delta_R)$

$M = \text{Mohawk}(q, \delta)$ // set of points that are w/in δ of q_x

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

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

$\delta = \min(\delta, d(r, j))$

Return δ.

Closest(P)

Running Analysis

need

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(Closest(Left) , Closest(Right)) → $2T(\frac{n}{2})$

$\frac{n}{2}$ $\frac{n}{2}$

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

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

$\Theta(n)$

6. Return (delta,r,j)

$\Rightarrow T(n) = 2T(\frac{n}{2}) + \Theta(n)$ 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 = MIN(Closest(Left) , Closest(Right))

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

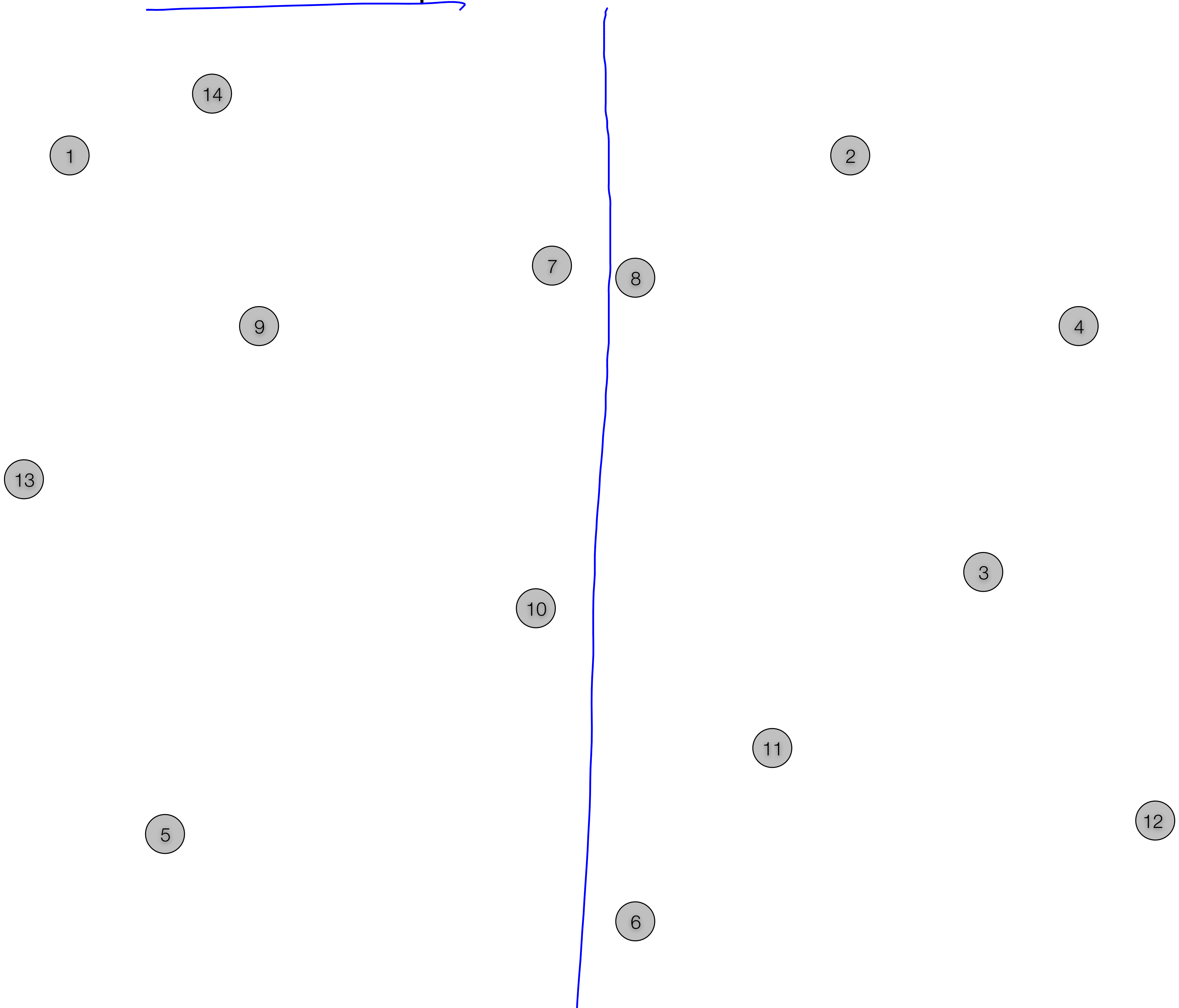
5. For each point x in Mohawk (in y-order):

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

6. Return (delta,r,j)

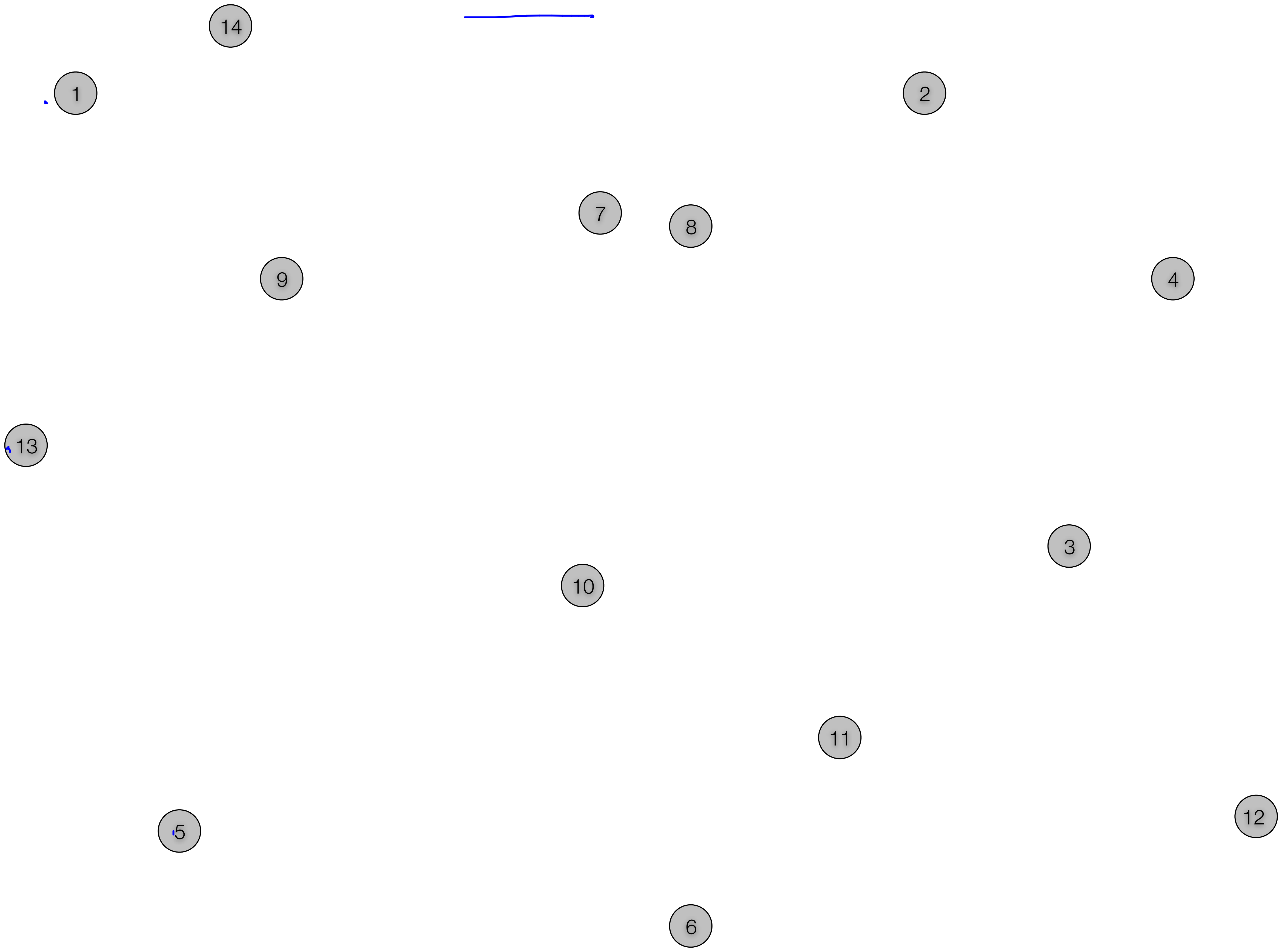
Can be reduced to 7!

Details: How to do step 1?



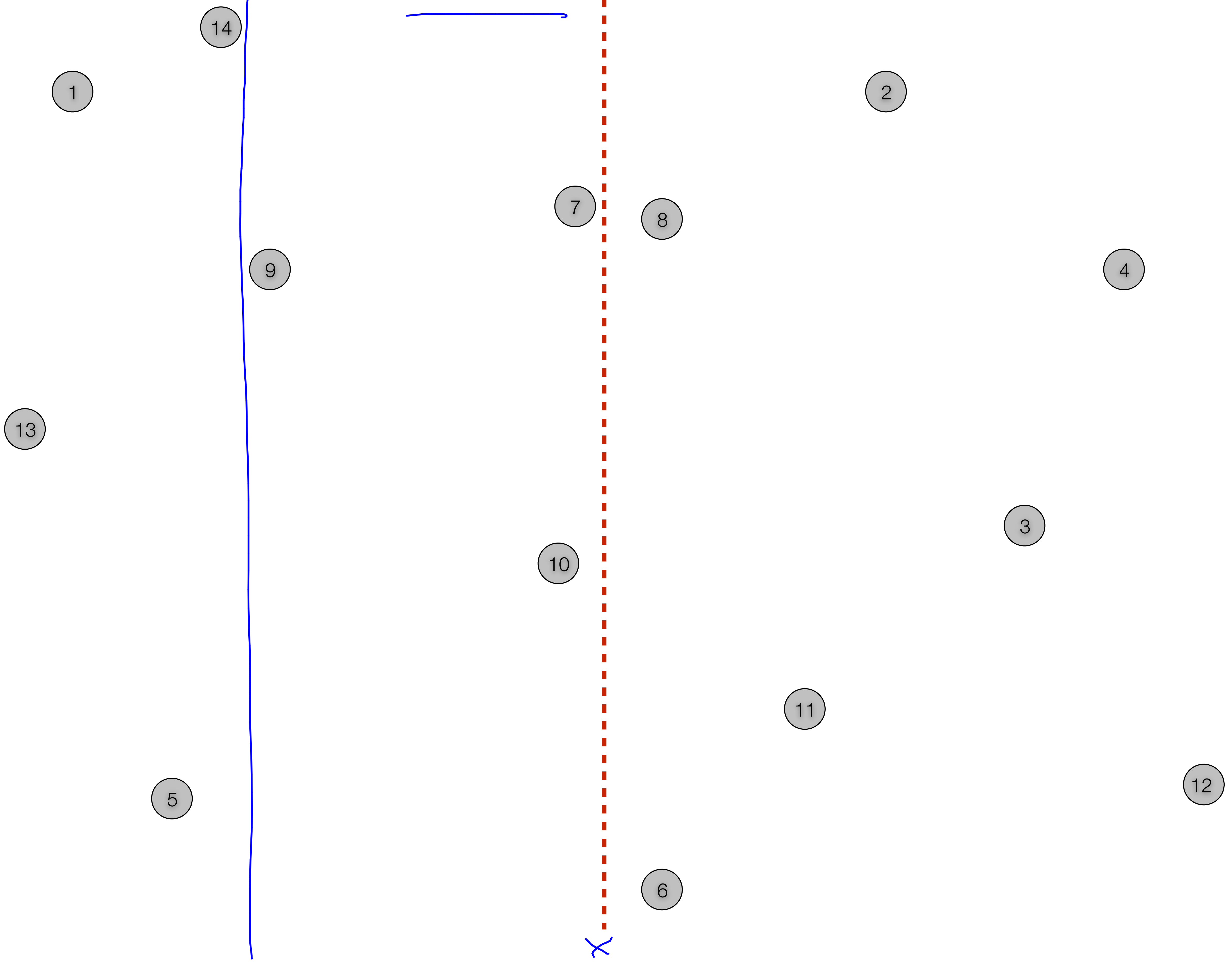
sorted in X: 13 1 5 14 9 10 7 9 8 11 2 3 4 12

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 $\Theta(1)$

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

LX – left points sorted
LY – left point sorted by y.

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

how to compute

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)

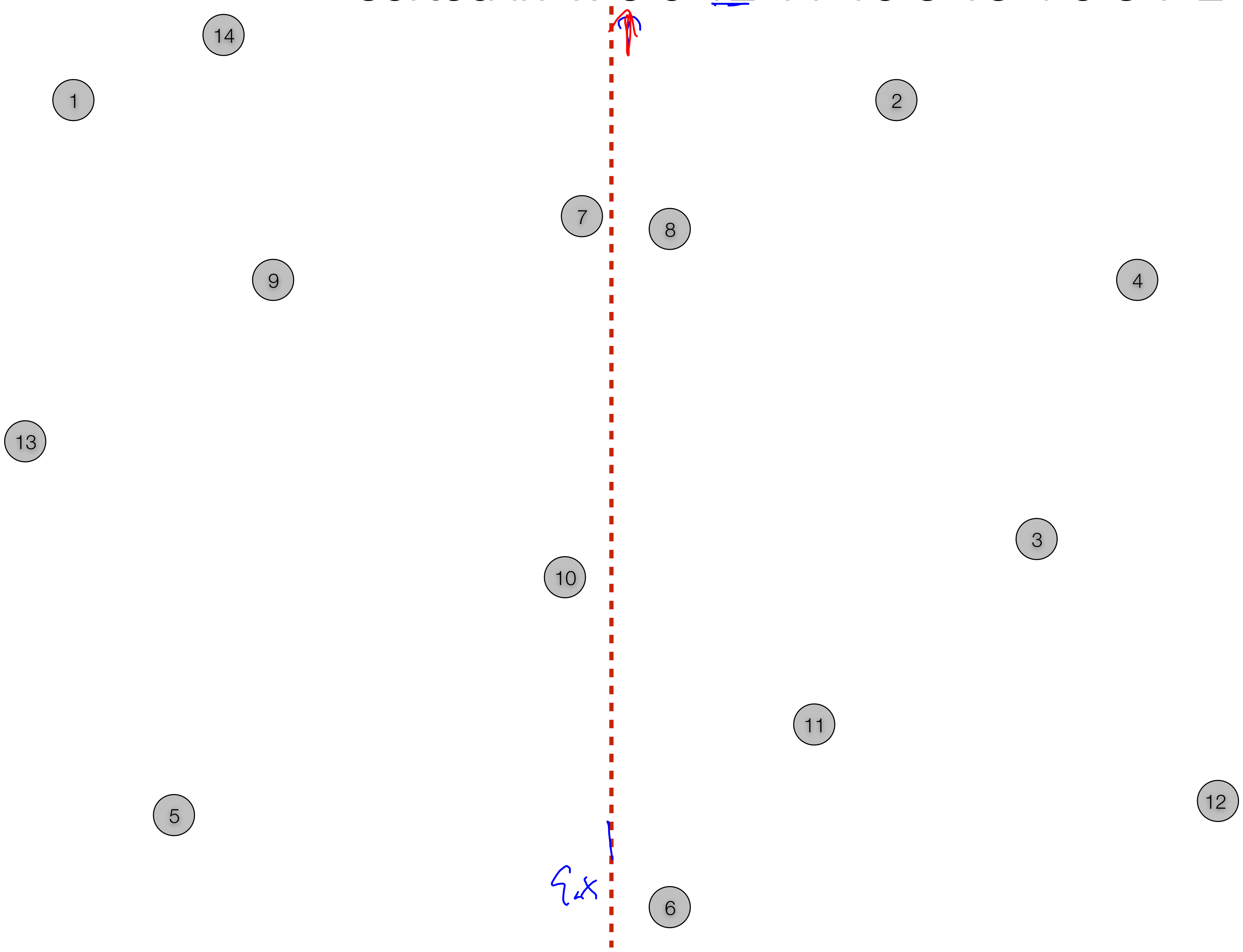
Can be reduced to 7!

Lx Ly

sorted in X: 13 1 5 14 9 10 7 6 8 11 2 3 4 12

Rx Ry

sorted in Y: 6 5 12 11 10 3 13 4 9 8 7 2 1 14



LY	RY
5	6
10	12
13	11
9	3
7	4
1	8
14	2

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.

(SY, 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)

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

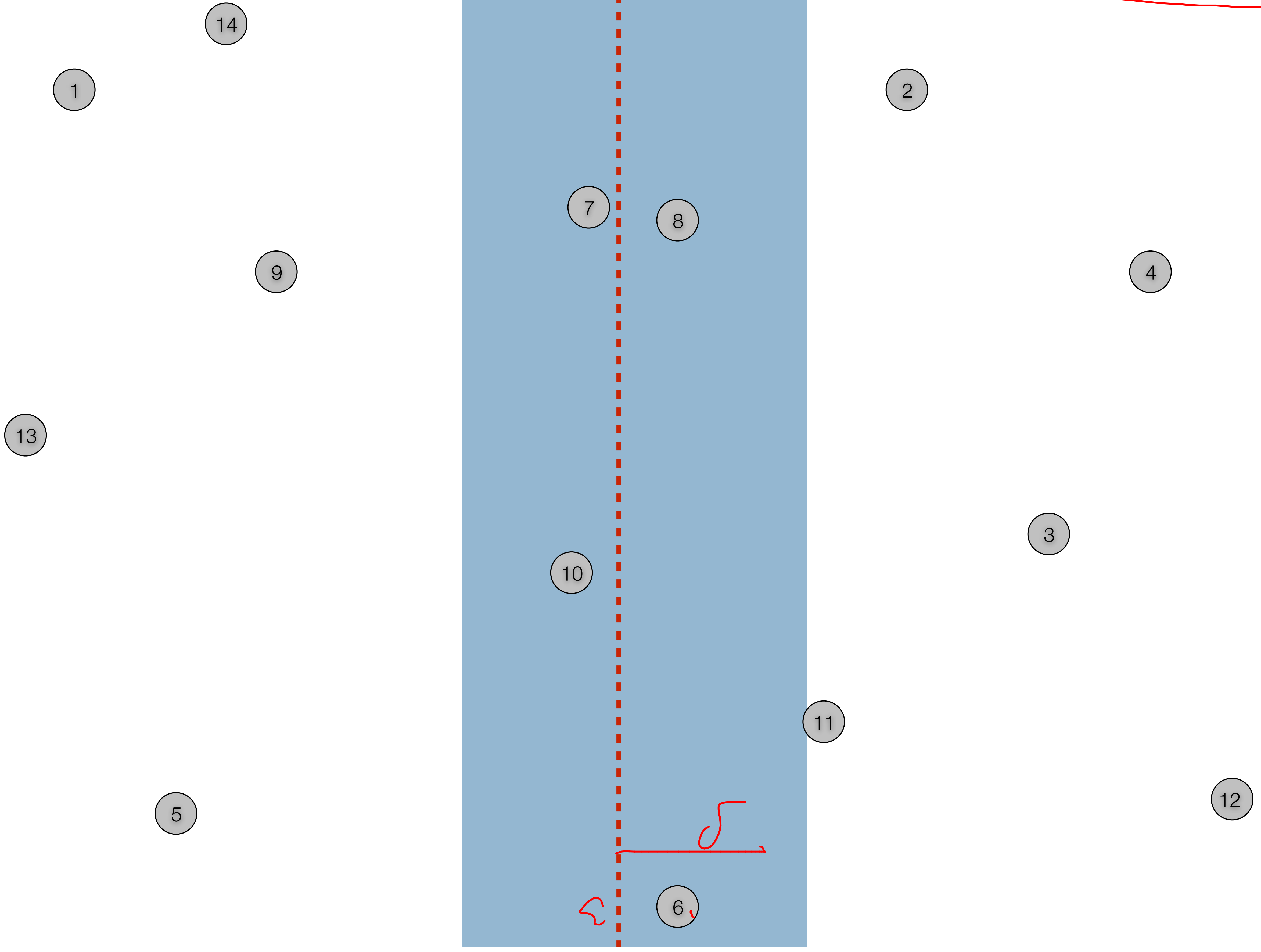
Update delta,r,j if any pair (x,y) is < delta

Return (delta,r,j)

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.

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.

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

$$T(n) =$$

$$T(n) = 2T(n/2) + \Theta(n) = \Theta(n \log n)$$

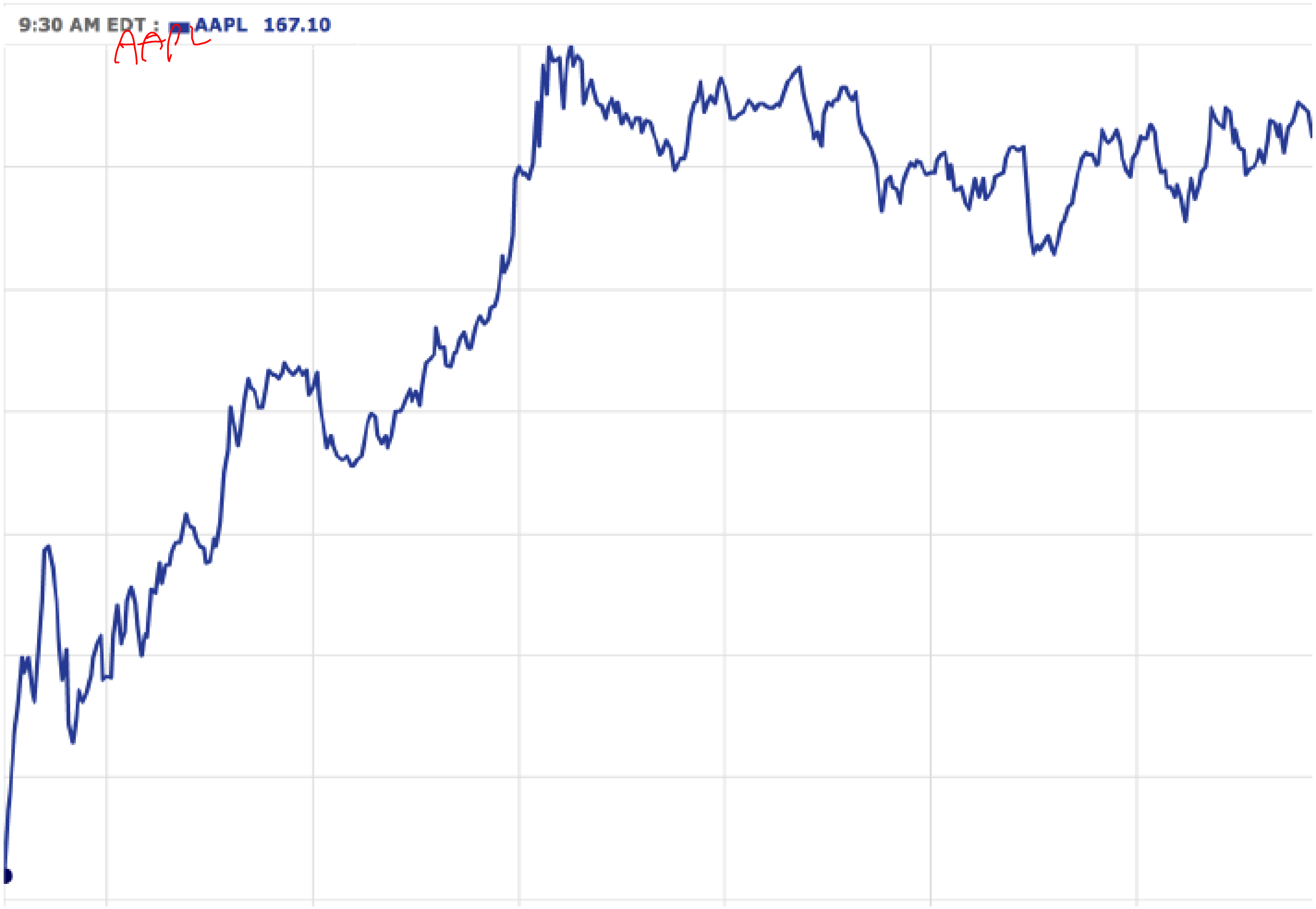
@author Robert Sedgewick
@author Kevin Wayne

http://algs4.cs.princeton.edu/99hull/ClosestPair.java.html

```java
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);
}
```

```java
// find closest pair of points in pointsByX[lo..hi]
// precondition:  pointsByX[lo..hi] and pointsByY[lo..hi] are the same sequence of points, sorted by x,y-coord
private double closest(Point2D[] pointsByX, Point2D[] pointsByY, Point2D[] aux, int lo, int hi) {
    if (hi <= lo) return Double.POSITIVE_INFINITY;

    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;
}
```

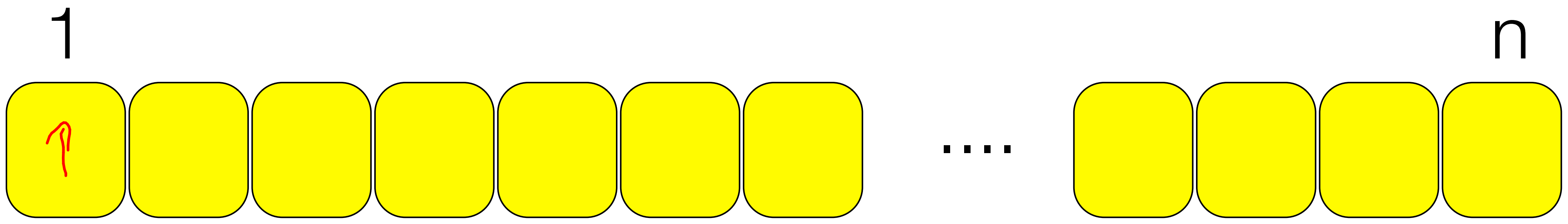
arbitrage


9:30 AM EDT : AAPL 167.10
AAPL

Open 27.46
Close 27.07
Low 26.65
High 27.69
Vol 33.79K
% Chg -75.68%
BPT 27.07
YAHOO! FINANCE


12:38 PM EDT : AIG 40.58
AIG
© 2008 Yahoo! Inc.
10:00 AM
11:00 AM
12:00 PM
1:00 PM
2:00 PM
3:00 PM
4:00

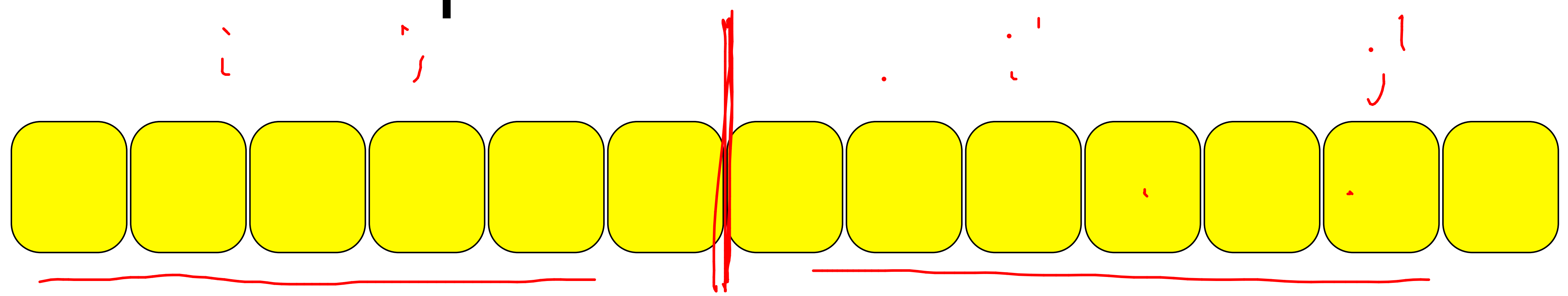
input: array of n numbers



goal: to find index i, j s.t. $i < j$ which maximize $A[j] - A[i]$

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

first attempt



`arbit(A[1...n])` {

-handle base case. if $|A| \le 2$.

→ $(i,j) \leftarrow \text{Arbit}(A[1,\dots \frac{n}{2}])$

→ $(i',j') \leftarrow \text{Arbit}(A[\frac{n}{2}+1,\dots n])$

$r^* \leftarrow \min(A[1\dots \frac{n}{2}])$

$j^* \leftarrow \max(A[\frac{n}{2}+1, n])$

Return $\max\{(i,j)\ (i',j')\ (r^*,j^*)\}$

}

ℓ mean

$A[j]-A[i]$

$A[j']-A[i']$

$A[j]-A[r]$

first attempt

```
arbit(A[1...n])
    base case if |A|<=2
    lg = arbit(left(A))
    rg = arbit(right(A))
    minl = min(left(A))
    maxr = max(right(A))

    return max{maxr-minl,lg,rg}
```

$\to T(\frac{n}{2})$

$\to T(\frac{n}{2})$

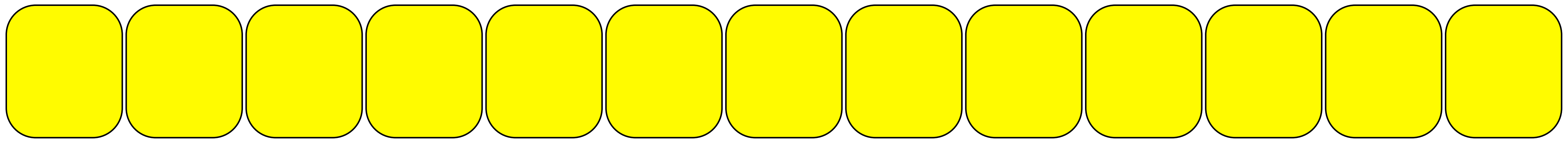
$\to \Theta(n)$

$\to \Theta(n)$

$\to$ too much

$T(n) = 2T(\frac{n}{2}) + \Theta(n) \Rightarrow \Theta(n \log n)$ solution.

first attempt: time $\Theta(n \log n)$



```
arbit(A[1...n])
    base case if |A|<=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

better approach

Can we find a solution that has $T(n) = 2T(n/2) + O(1)$?

better approach

Can we find a solution that has $T(n) = 2T(n/2) + O(1)$?

```
minl = min(left(A))
maxr = max(right(A))

return max{maxr-minl,lg,rg}
```

$\Theta(n)$

second attempt

`arbit+(A[1...n])` returns (best trade, min, max)

min of the array A

max of A

`base case if |A|<=2` (_ , _ _ ,)

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

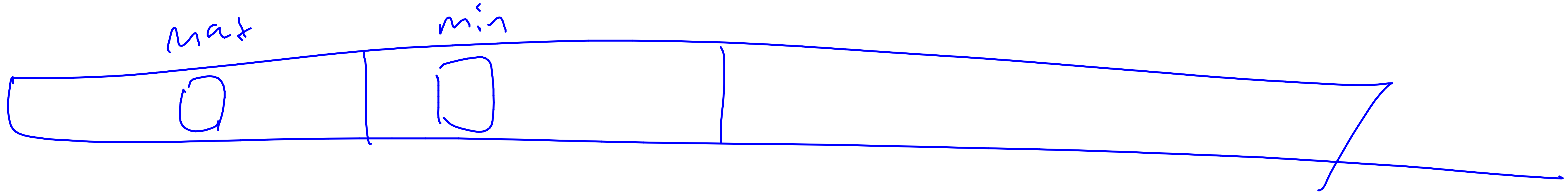
(lg, minl, maxl) ← Arbit (A[1.. n/2])

(rg, minr, maxr) ← Arbit (A[n/2+1.. n])

Return (max { lg, rg, maxr − minl }, $\Theta(1)$

min { minl, minr }

max { maxl, maxr })


max
0
min
0

second attempt

```
arbit+(A[1...n])
 base case if |A|<=2, …
 (lg,minl,maxl) = arbit(left(A))
 (rg,minr,maxr) = arbit(right(A))
   return max{maxr-minl,lg,rg},
          min{minl, minr},
          max{maxl, maxr}
```

Matrix

multiplication

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \bigstar \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \bigstar \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 5+14 & 6+16 \\ 15+28 & 18+32 \end{bmatrix}$$

$$= \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & & & \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & & & \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & & & \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & & & \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}$$

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & & & \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & & & \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

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

$$\Theta(n^3)$$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

[Strassen]

$$P_1 = A(F - H)$$

$$P_2 = (A + B)H$$

$$P_3 = (C + D)E$$

$$P_4 = D(G - E)$$

$$P_5 = (A + D)(E + H)$$

$$P_6 = (B - D)(G + H)$$

$$P_7 = (A - C)(E + F)$$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$R = P_5 + P_4 - P_2 + P_6$

$S = P_1 + P_2$

$T = P_3 + P_4$

$U = P_5 + P_1 - P_3 - P_7$

[strassen]

$P_1 = A(F - H)$

$P_2 = (A + B)H$

$P_3 = (C + D)E$

$P_4 = D(G - E)$

$P_5 = (A + D)(E + H)$

$P_6 = (B - D)(G + H)$

$P_7 = (A - C)(E + F)$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$R = P_5 + P_4 - P_2 + P_6$

$S = P_1 + P_2$

$T = P_3 + P_4$

$U = P_5 + P_1 - P_3 - P_7$

[strassen]

$P_1 = A(F - H)$

$P_2 = (A + B)H$

$P_3 = (C + D)E$

$P_4 = D(G - E)$

$P_5 = (A + D)(E + H)$

$P_6 = (B - D)(G + H)$

$P_7 = (A - C)(E + F)$

$$= \begin{bmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{bmatrix}$$

$$R = P_5 + P_4 - P_2 + P_6 \qquad S = P_1 + P_2$$

$$T = P_3 + P_4 \qquad U = P_5 + P_1 - P_3 - P_7$$

[strassen]

$$P_1 = A(F - H)$$

$$P_2 = (A + B)H$$

$$P_3 = (C + D)E$$

$$P_4 = D(G - E)$$

$$P_5 = (A + D)(E + H)$$

$$P_6 = (B - D)(G + H)$$

$$P_7 = (A - C)(E + F)$$

$$M(n) = 7M(n/2) + 18n^2$$

$$= \Theta(n^{\log_2 7})$$

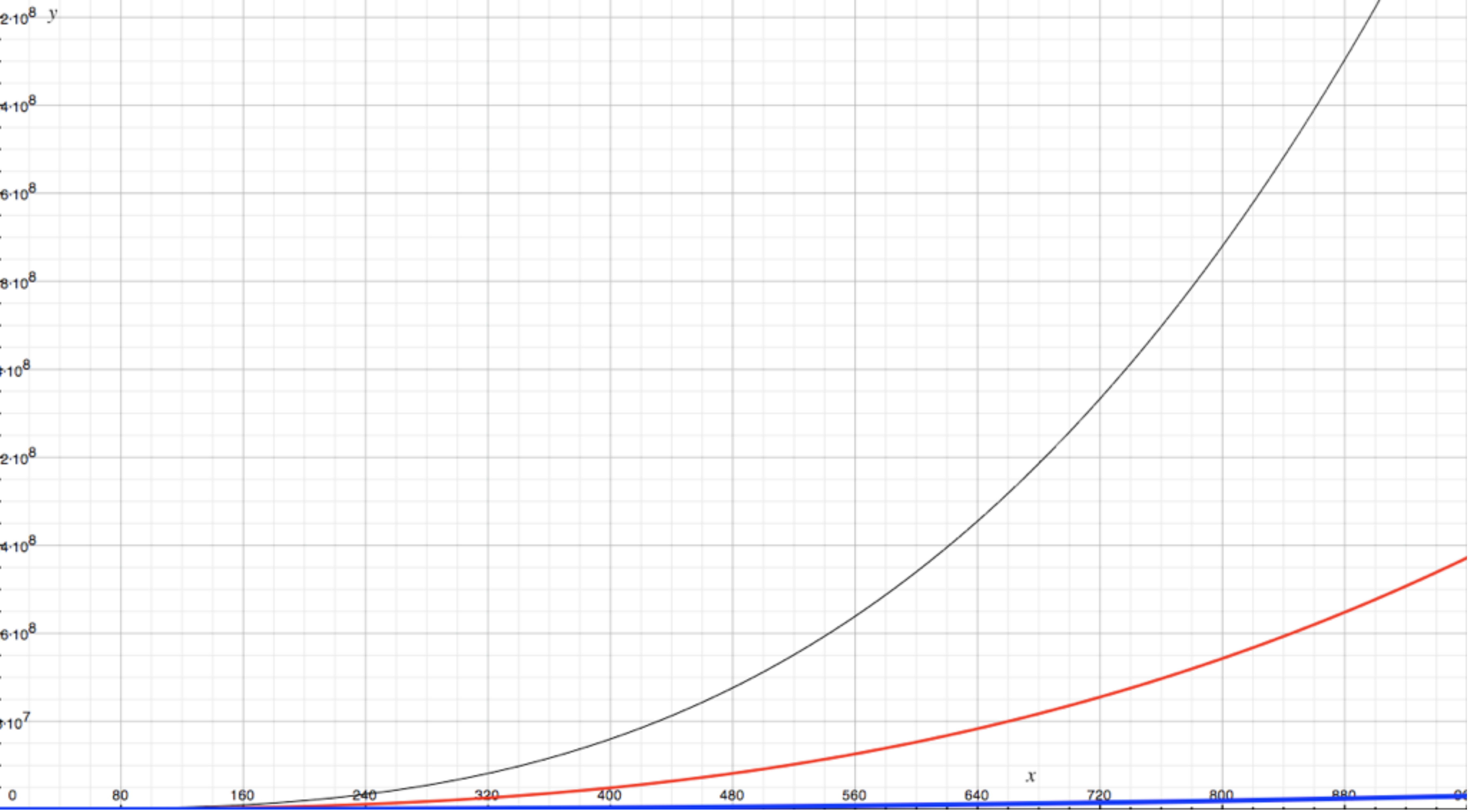
taking this idea further

3x3 matricies

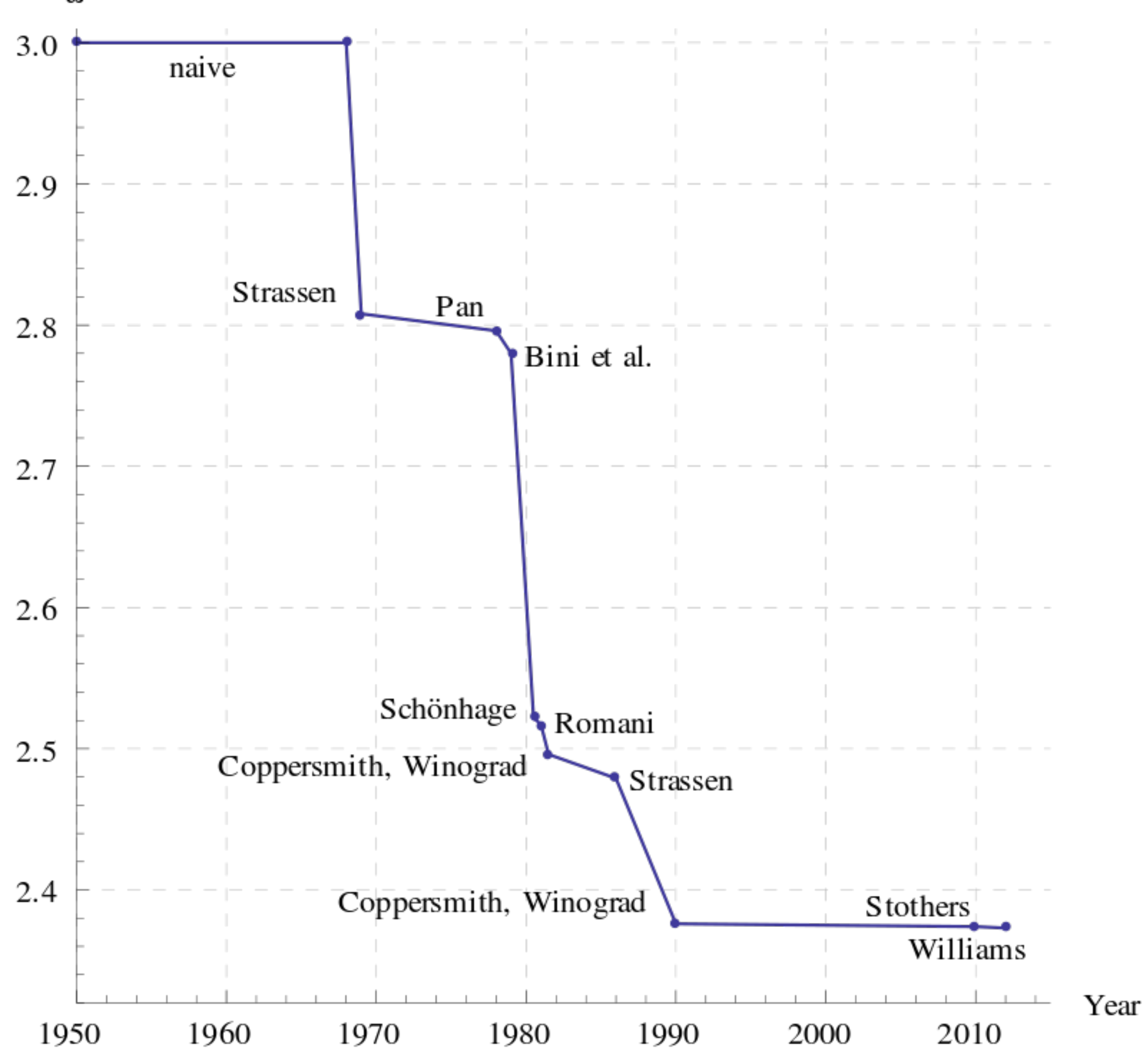
1978 victor pan method

70x70 matrix using 143640
mults

what is the recurrence:

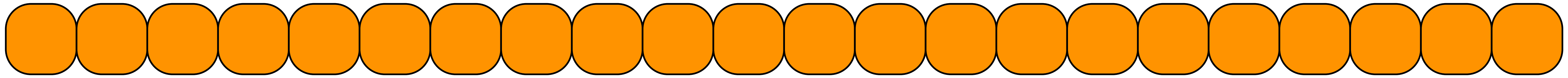


y
x
0
80
160
240
320
400
480
560
640
720
800
880

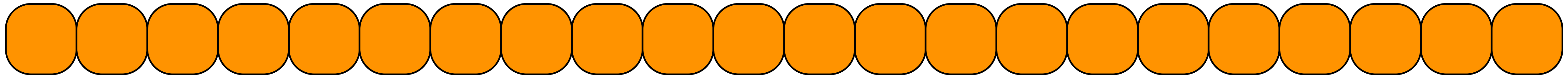


https://en.wikipedia.org/wiki/File:Bound_on_matrix_multiplication_omega_over_time.svg

MEDIAN



problem: given a list of **n** elements, find the element of rank **n**/2. (half are larger, half are smaller)

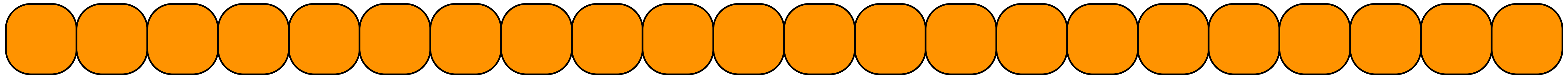


problem: given a list of **n** elements, find the element of rank n/2. (half are larger, half are smaller)

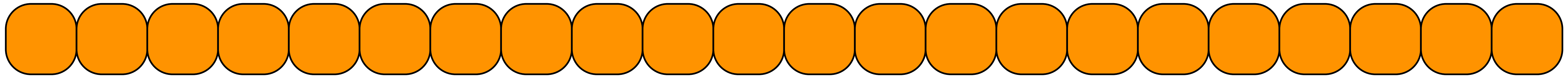
can generalize to **i**

first solution: sort and pluck.

$$O(n \log n)$$



problem: given a list of **n** elements, find the element of rank **i**.

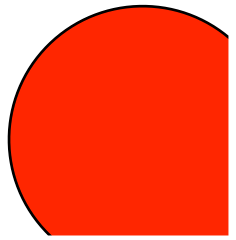


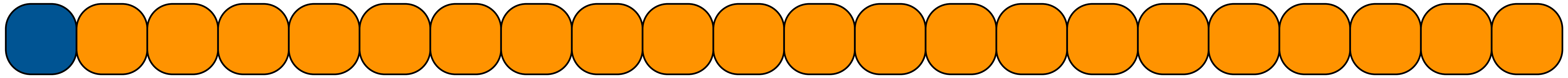
problem: given a list of **n** elements, find the element of rank **i**.

key insight:

we do not have to “fully” sort.

semi sort can suffice.





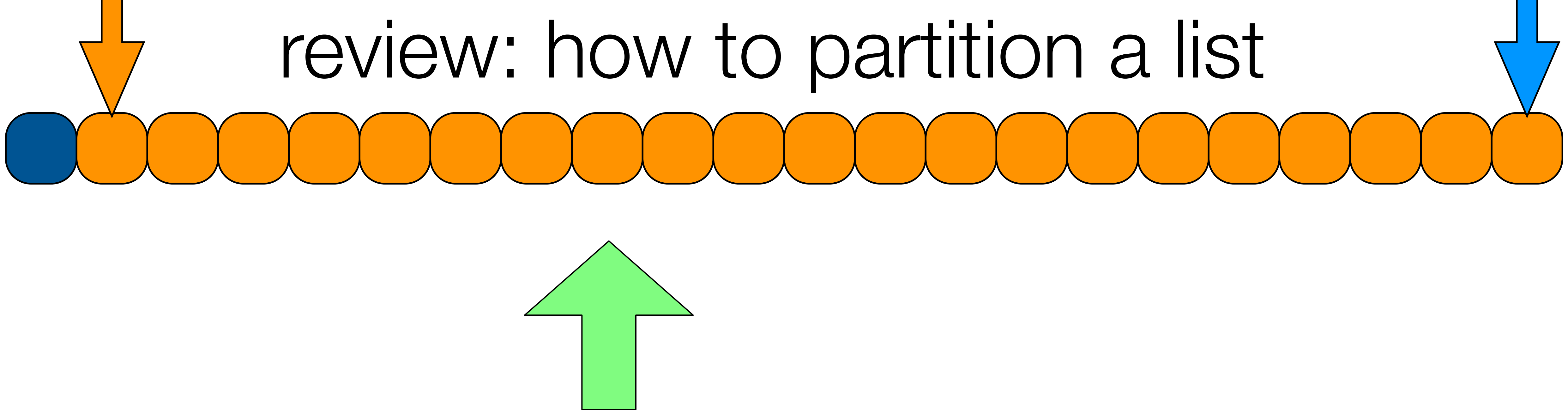
pick first element

partition list about this one

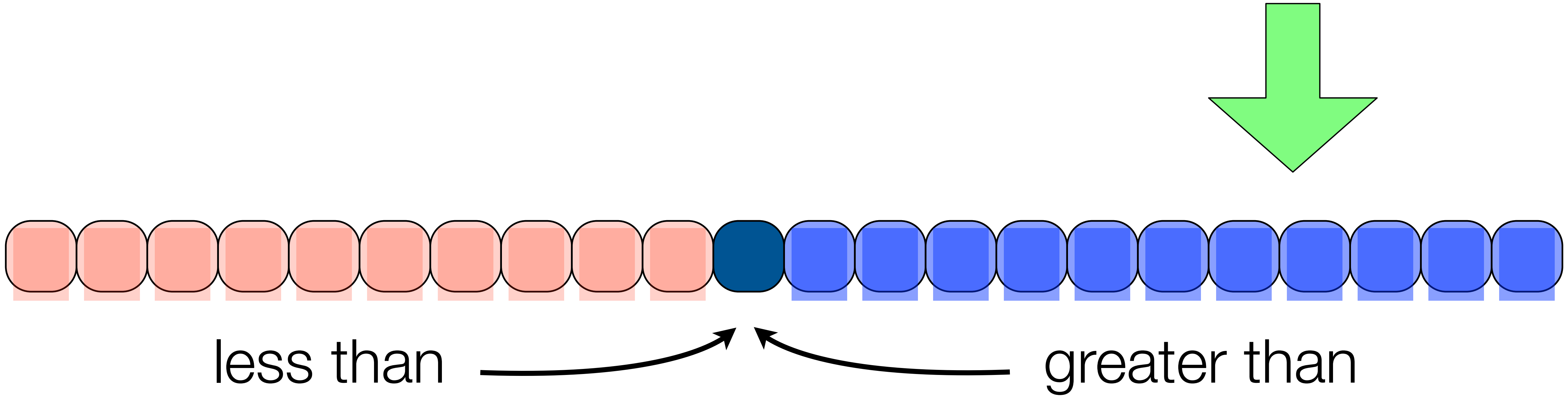
see where we stand

review: how to partition a list

review: how to partition a list

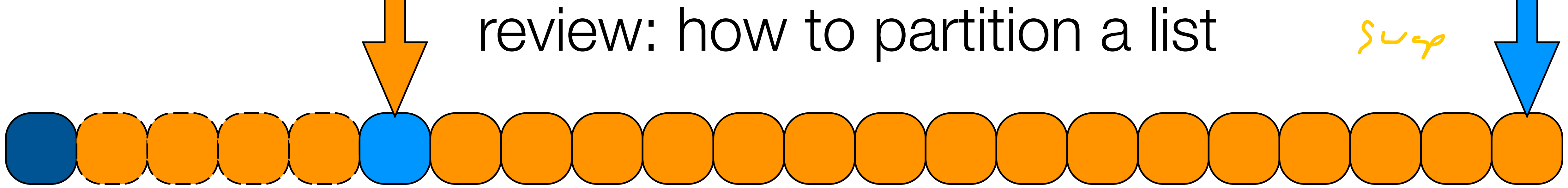


GOAL: start with THIS LIST and END with THAT LIST

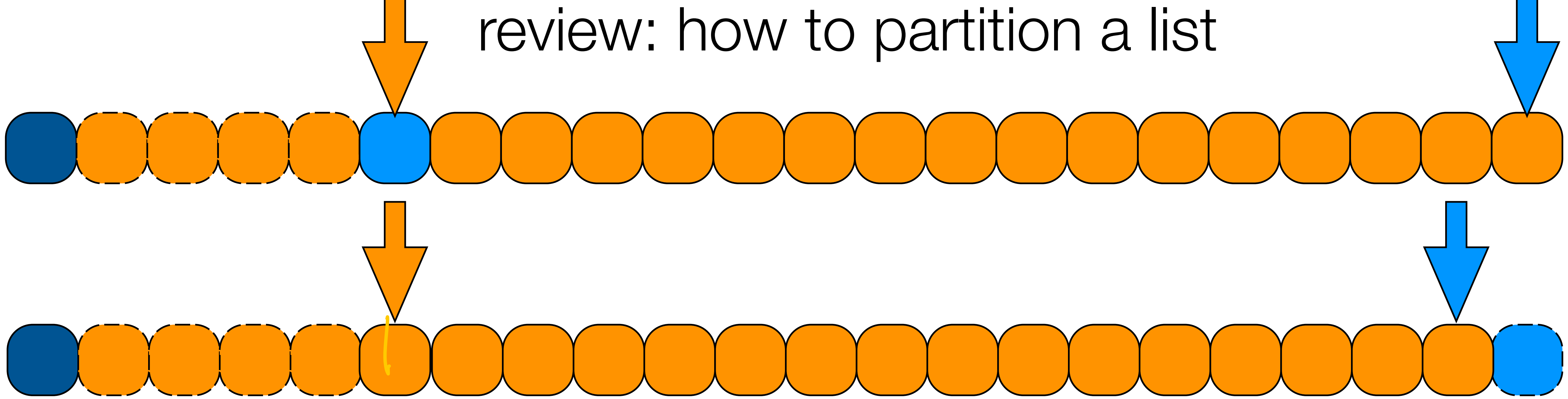


review: how to partition a list

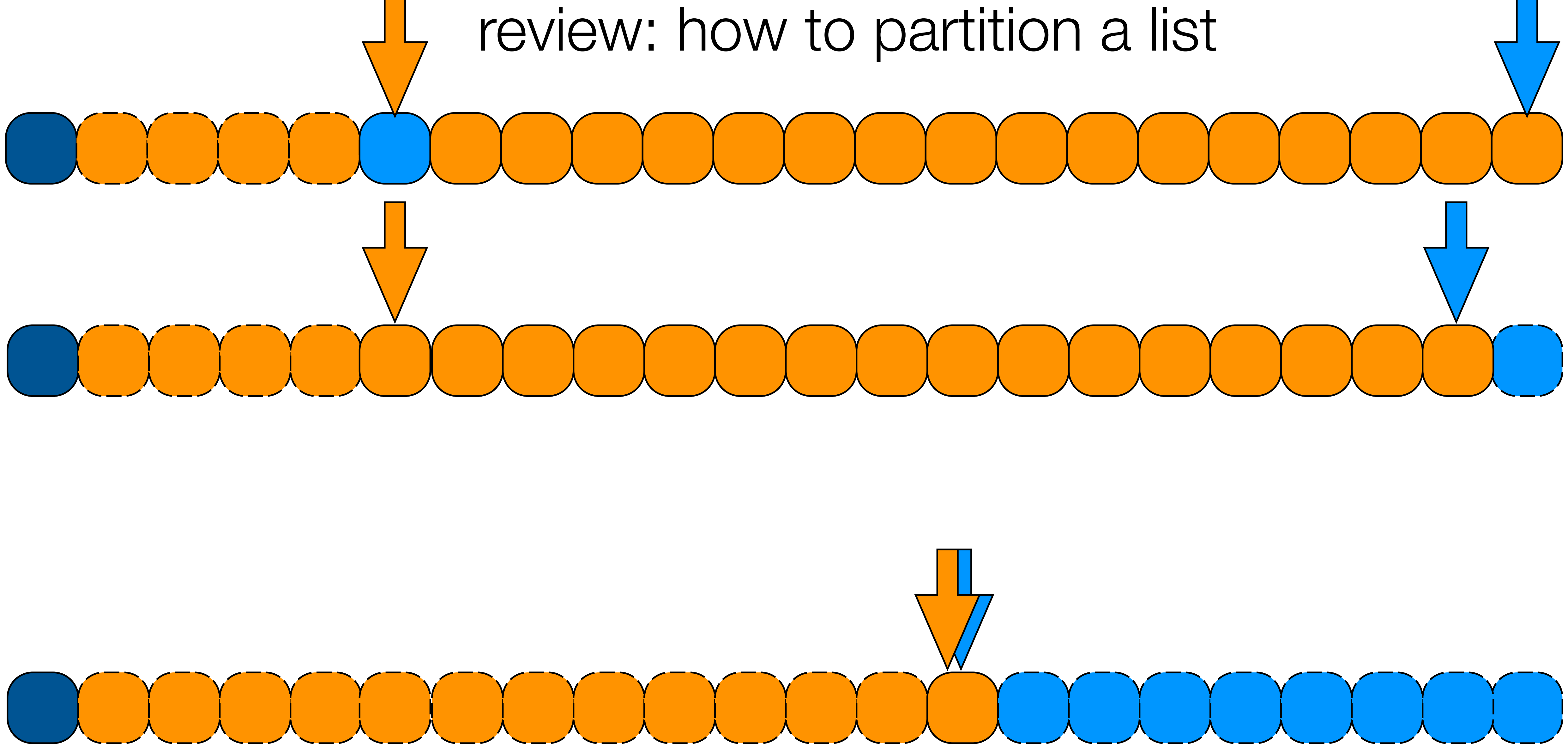
review: how to partition a list



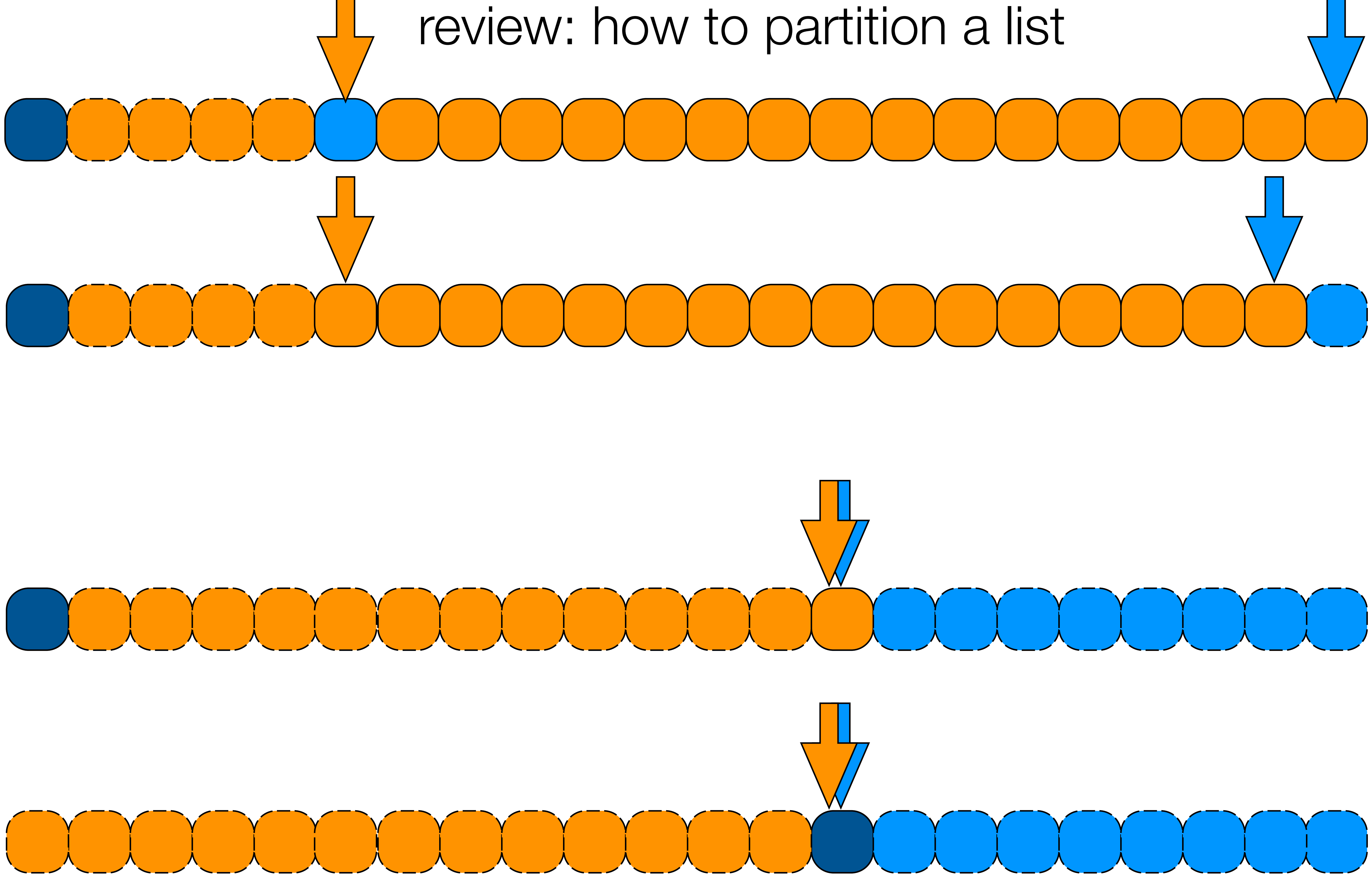
review: how to partition a list



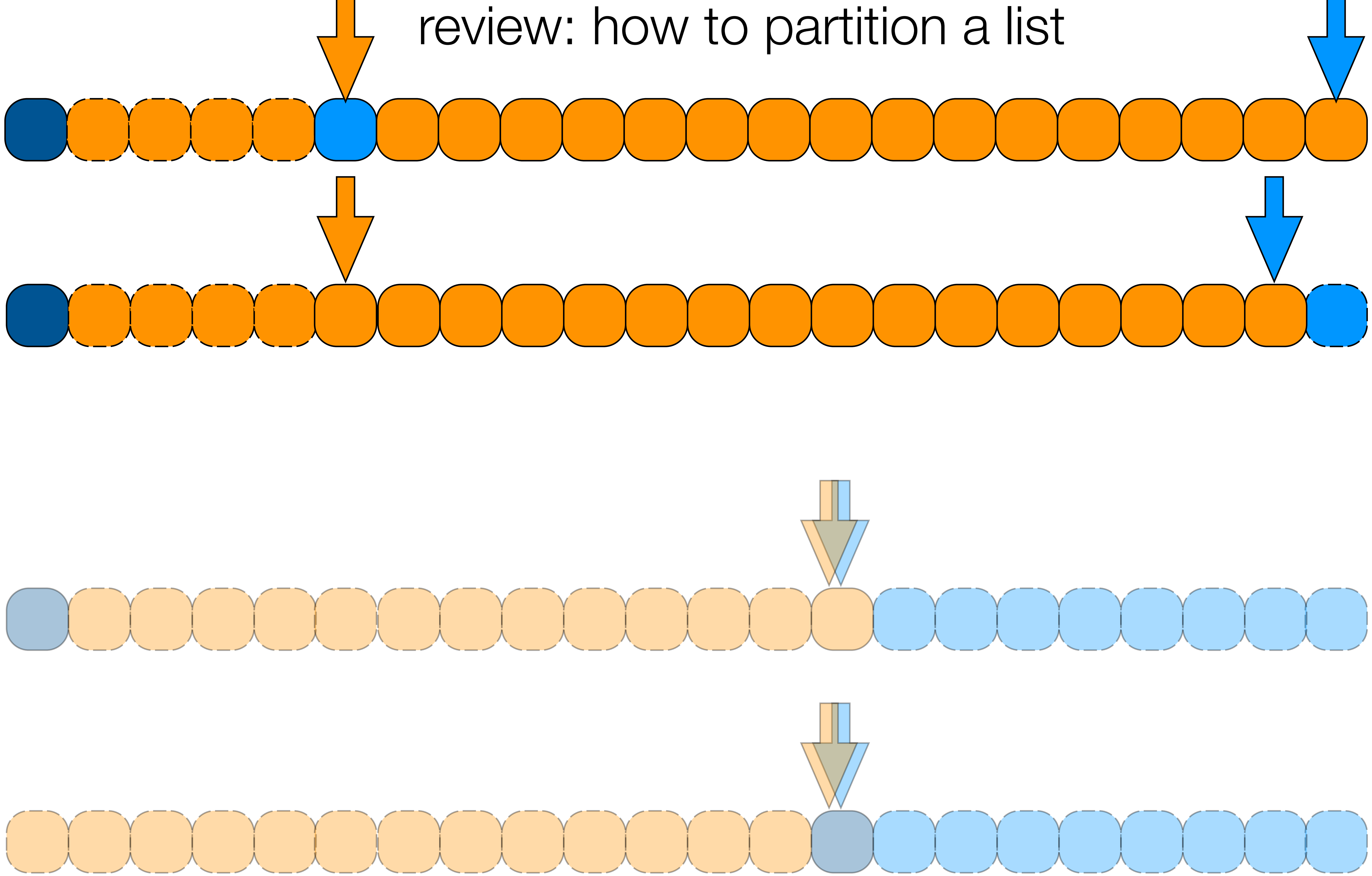
review: how to partition a list



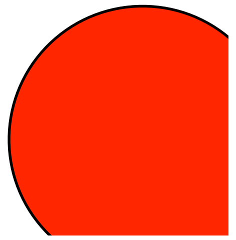
review: how to partition a list

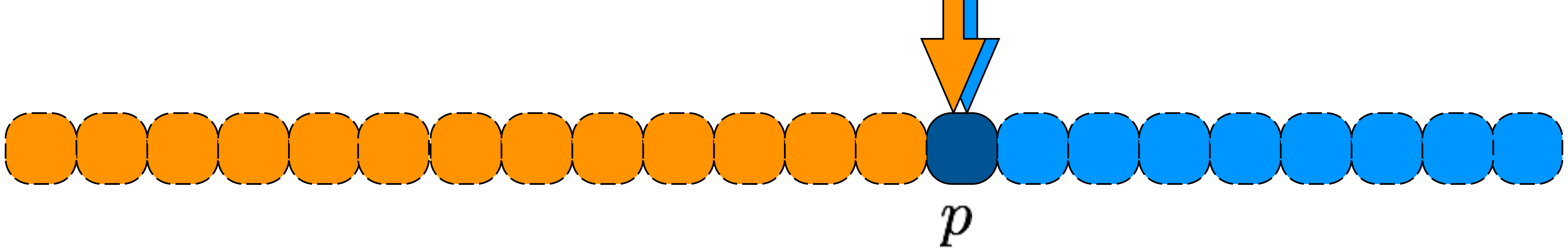


review: how to partition a list

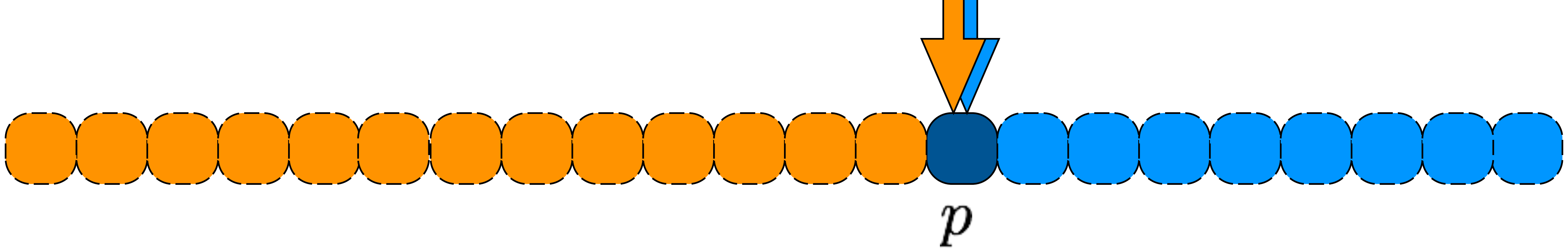


partitioning a list about an element takes linear time.





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

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

select $(i, A[1, \ldots, n])$

Assume our partition always splits list into two eql 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 $((i-p-1), A[p+1, \ldots, n])$

select $(i, A[1, \ldots, n])$

Assume our partition always splits list into two eql parts

handle base case.

partition list about first element

if pivot is position i, return pivot

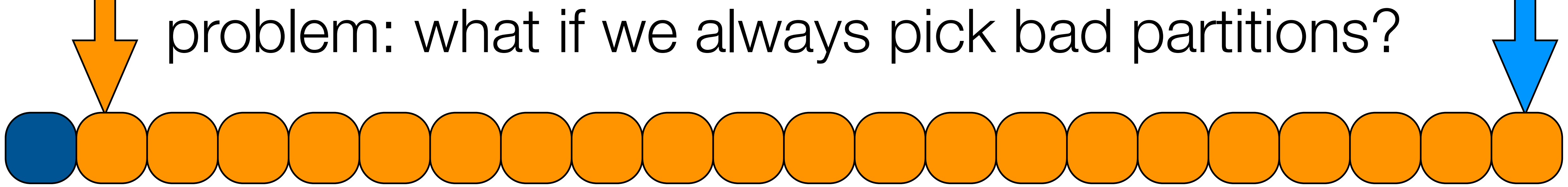
else if pivot is in position > i select $(i, A[1, \ldots, p-1])$

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

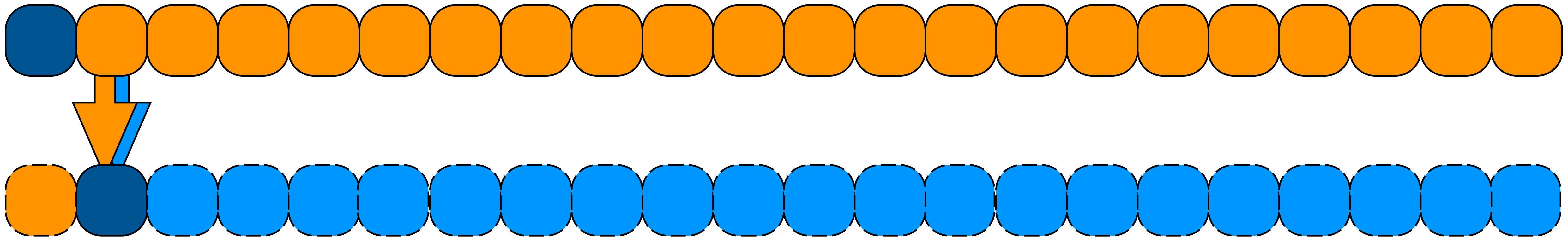
$$T(n) = T(n/2) + O(n)$$

$$\Theta(n)$$

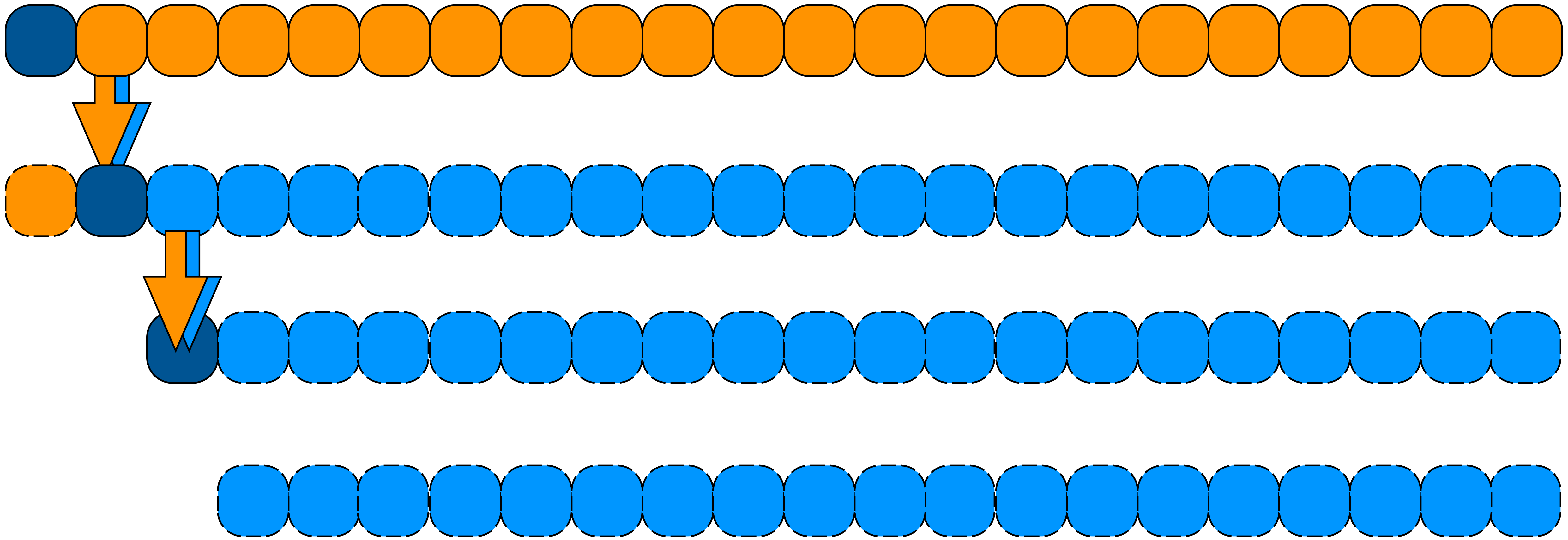
problem: what if we always pick bad partitions?

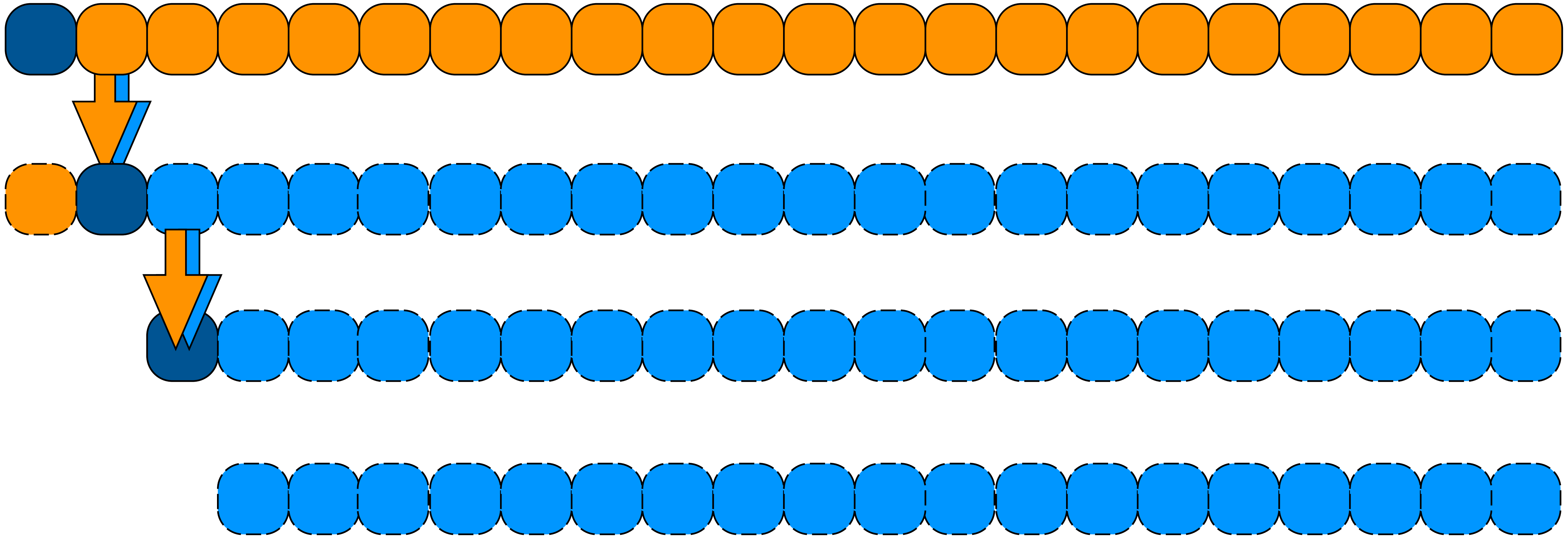


problem: what if we always pick bad partitions?



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 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   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

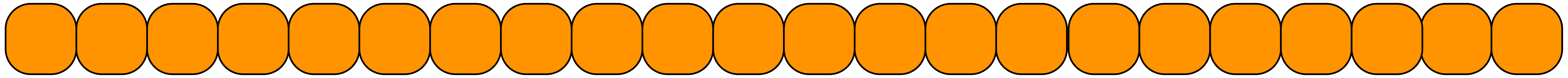
solution: bootstrap

image: gucci

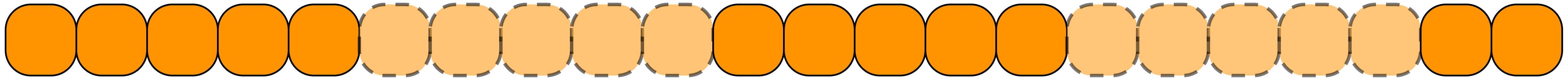
image: mark nason

image: d&g

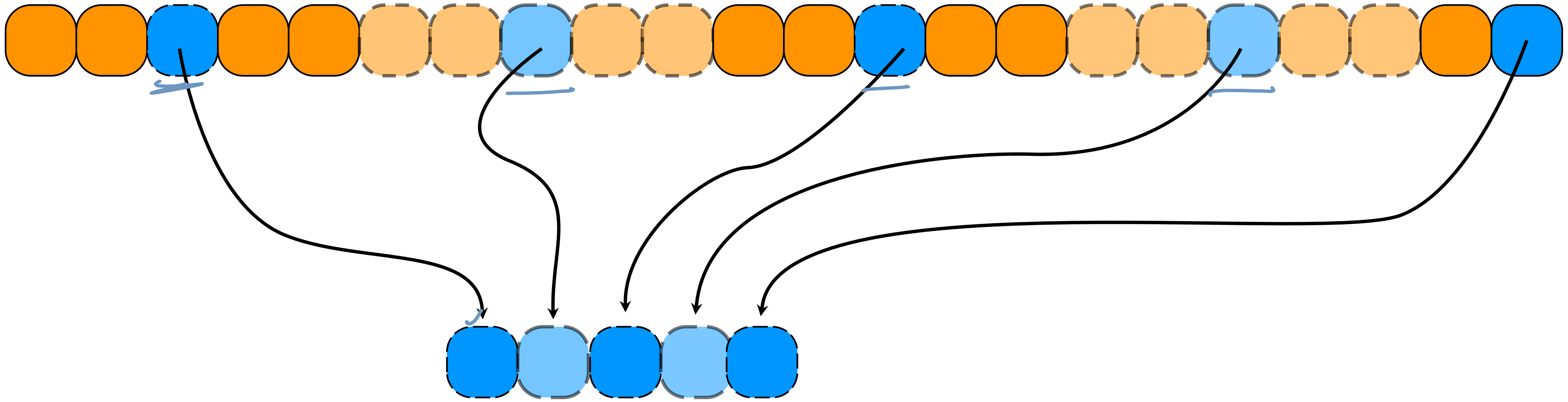
partition $(A[1, \ldots, n])$

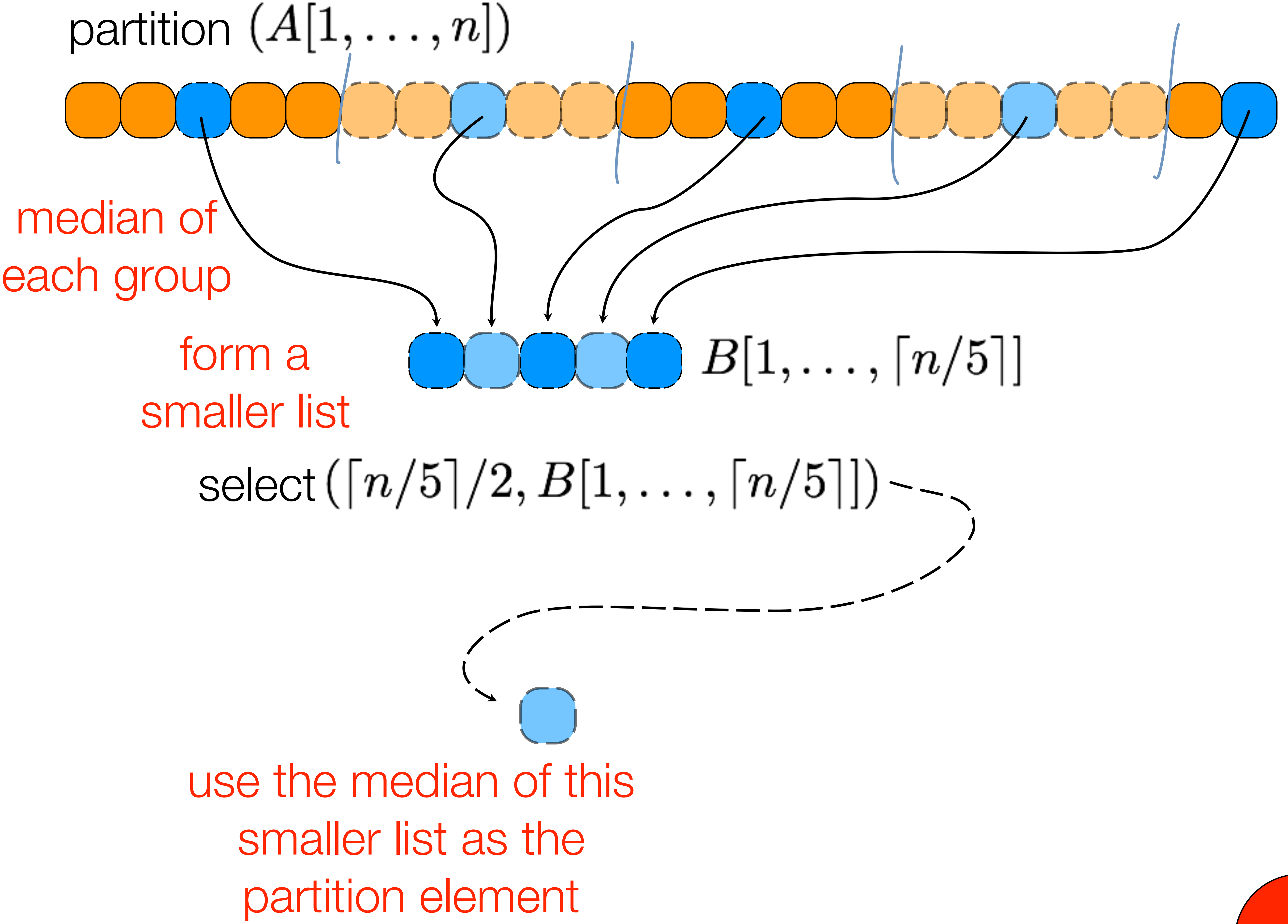


partition $(A[1, \ldots, n])$

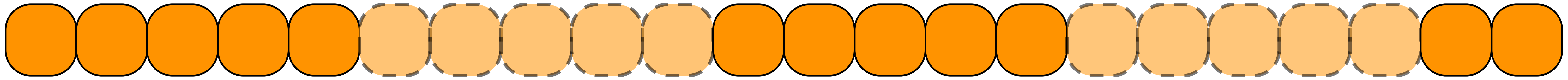


partition $(A[1,\ldots,n])$



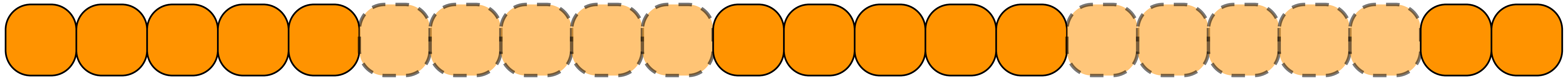

partition $(A[1, \ldots, n])$
median of each group
form a smaller list
$B[1, \ldots, \lceil n/5 \rceil]$
select $(\lceil n/5 \rceil / 2, B[1, \ldots, \lceil n/5 \rceil])$
use the median of this smaller list as the partition element

partition $(A[1, \ldots, n])$



1.
2.
3.
4.
5.

partition $(A[1,\ldots,n])$



divide list into groups of 5 elements

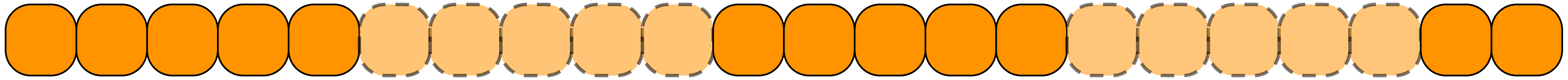
find median of each small list

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

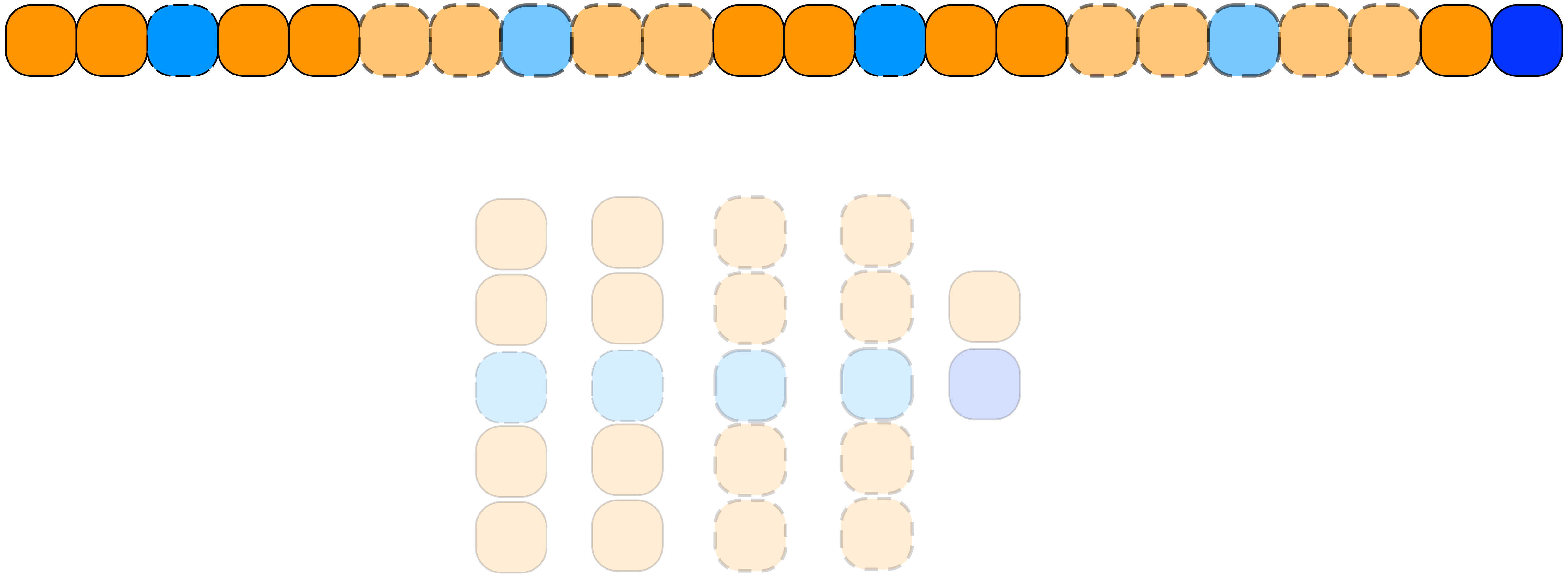
gather all medians

call select(...) on this sublist to find median

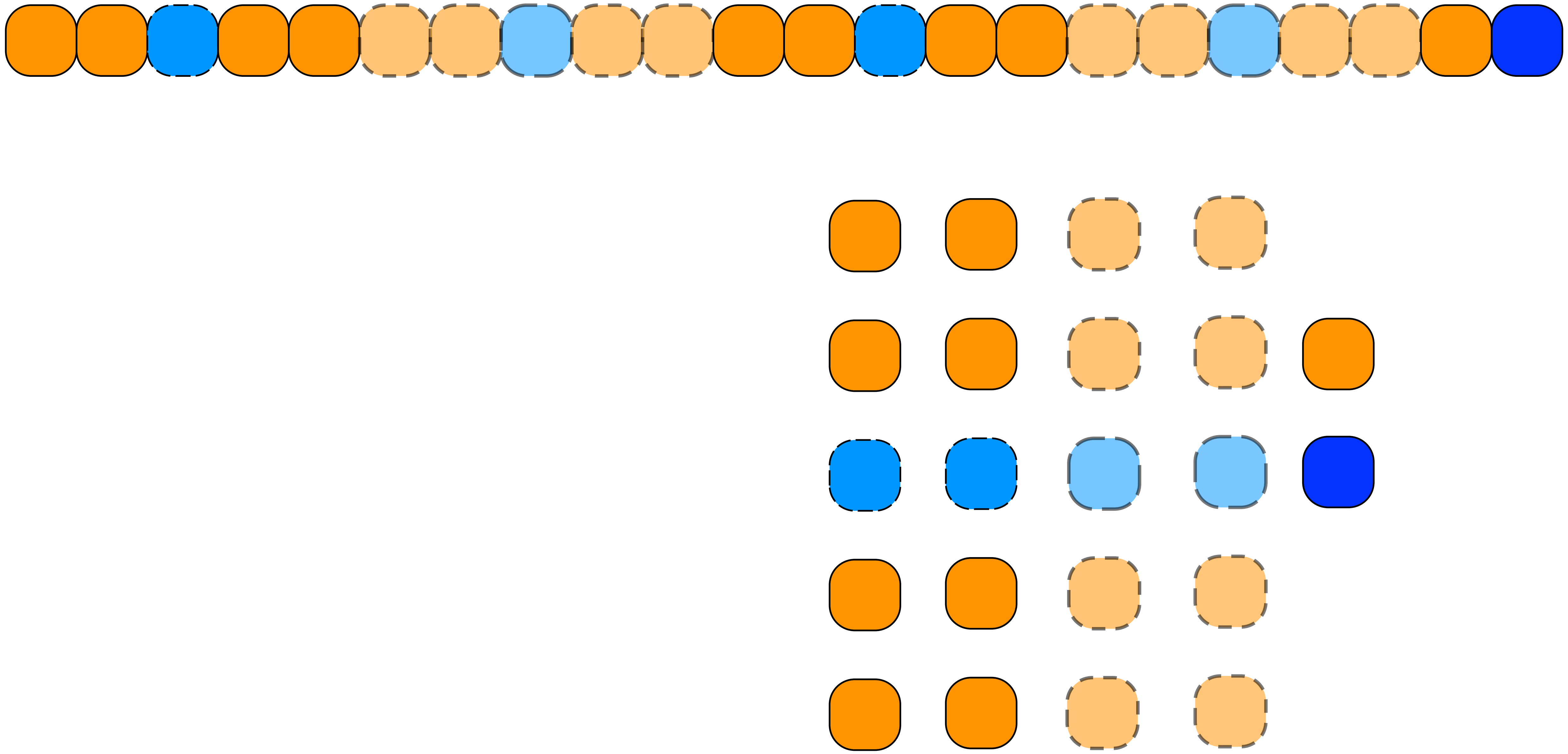
return the result

$$P(n) = S(\lceil n/5 \rceil) + O(n)$$

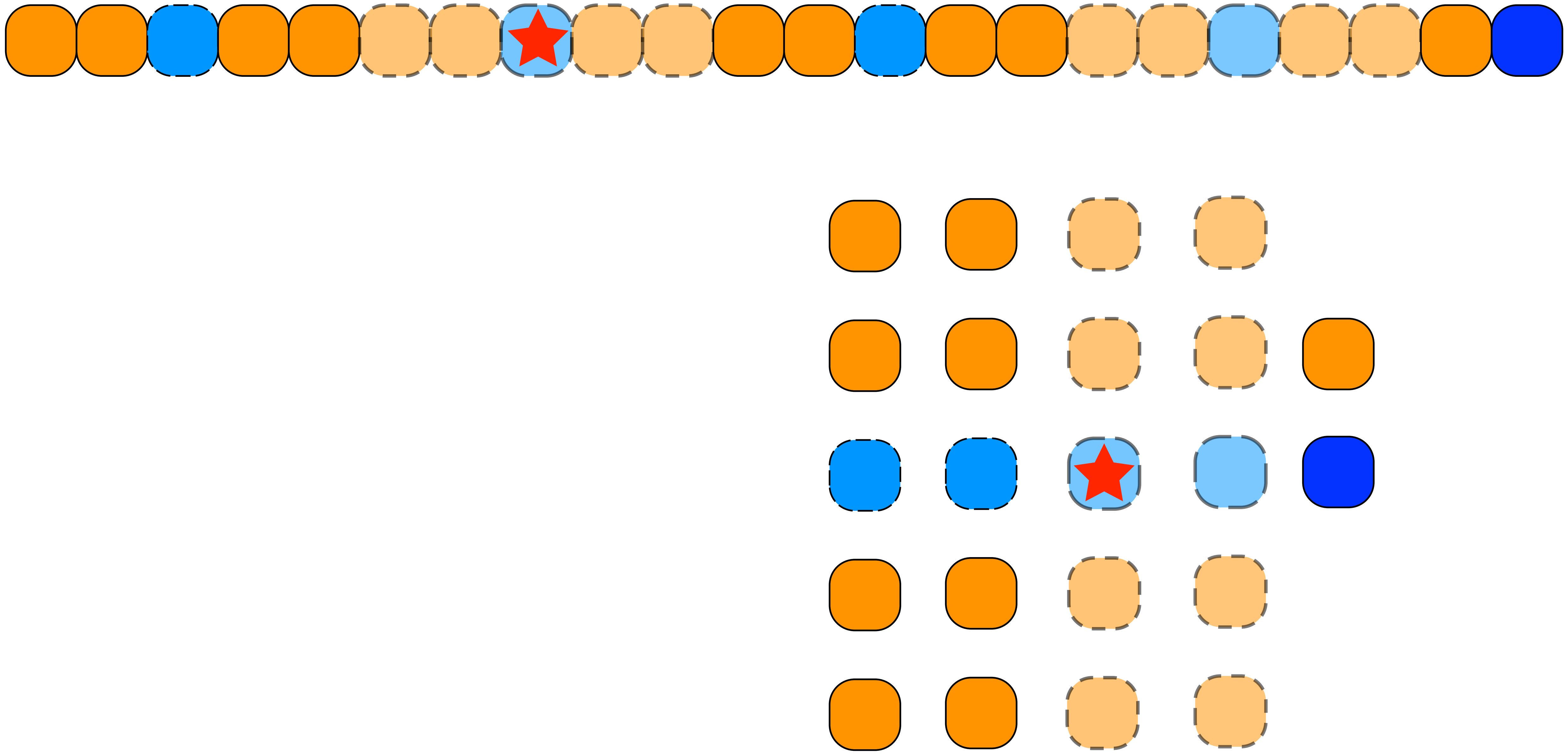
a nice property of our partition



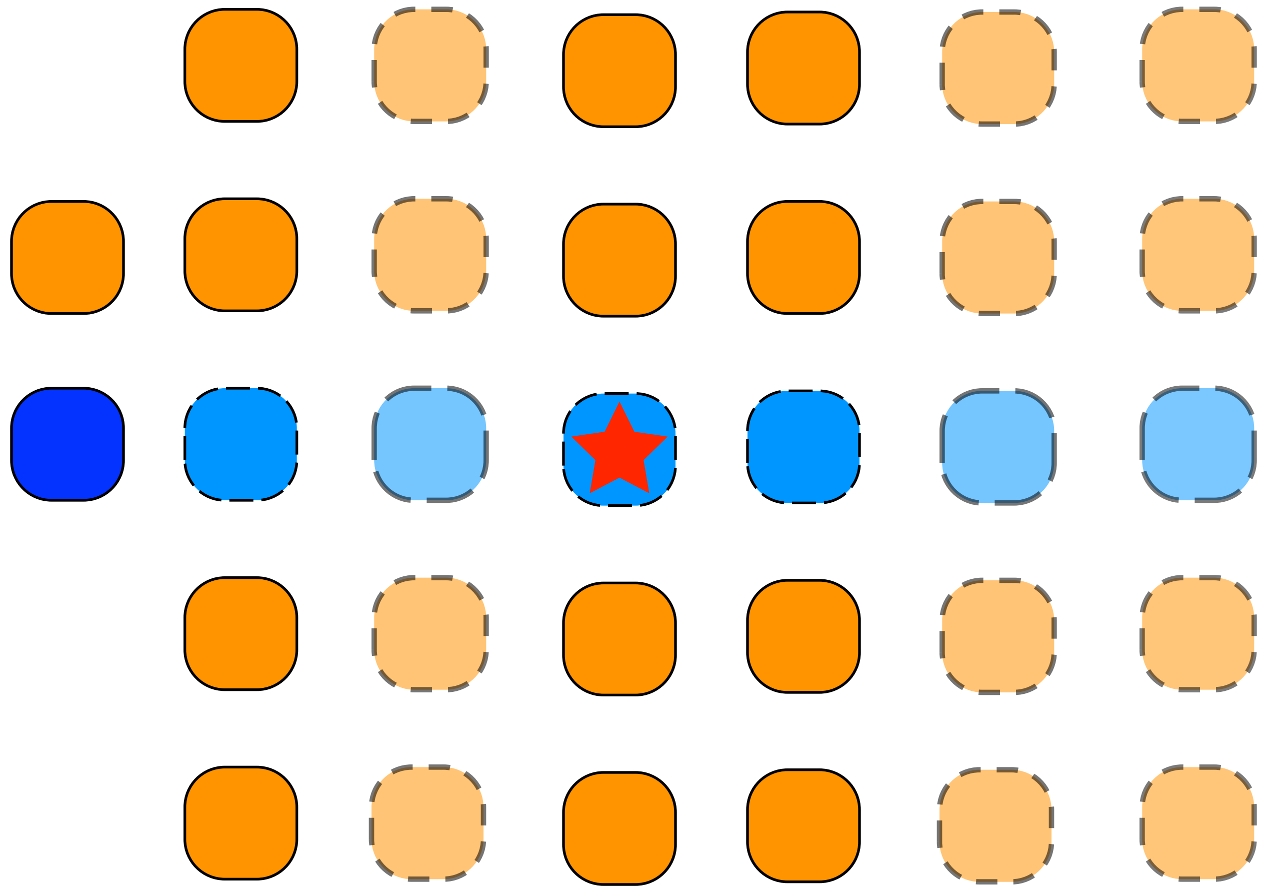
a nice property of our partition



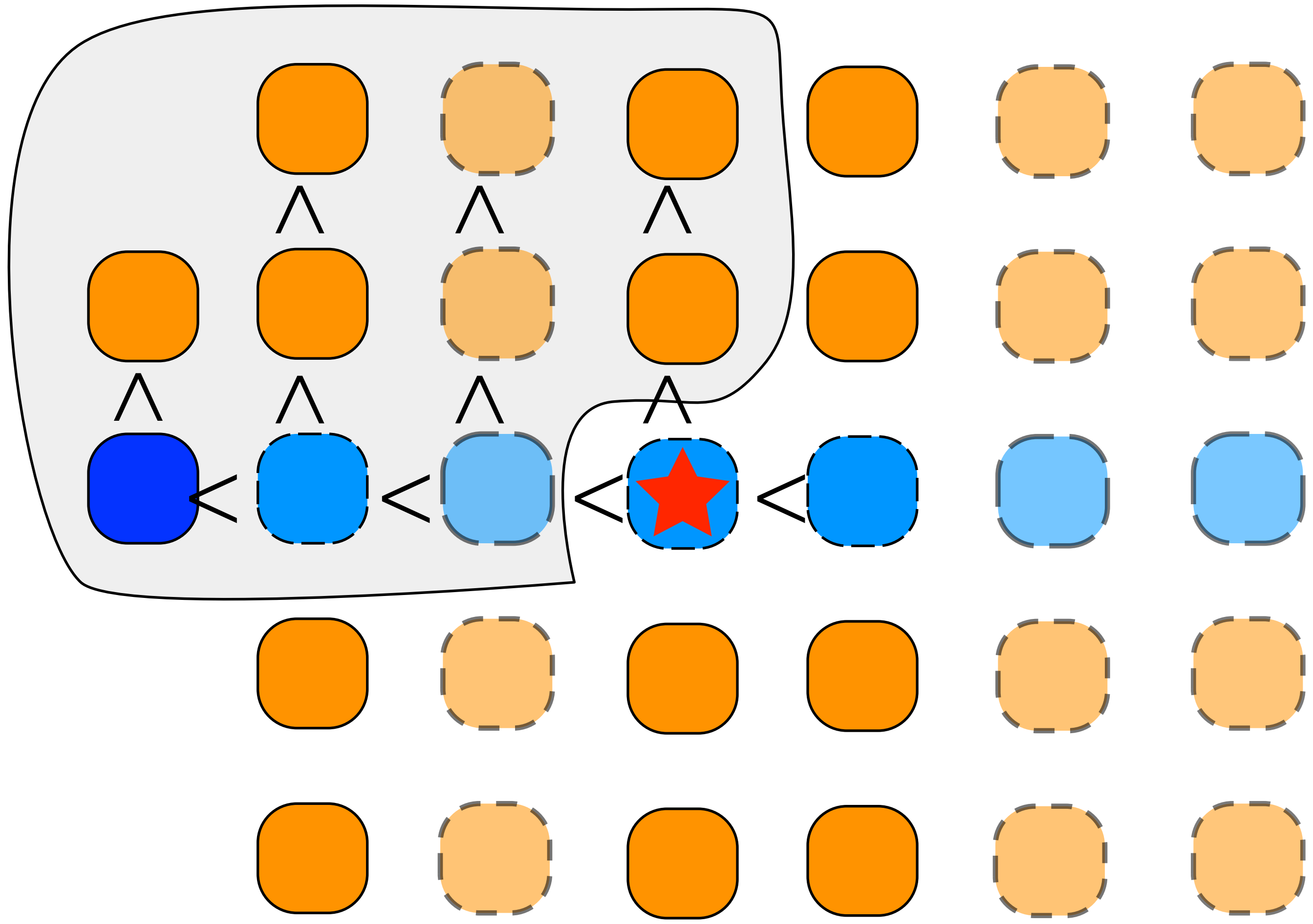
a nice property of our partition



SWITCH TO A BIGGER EXAMPLE



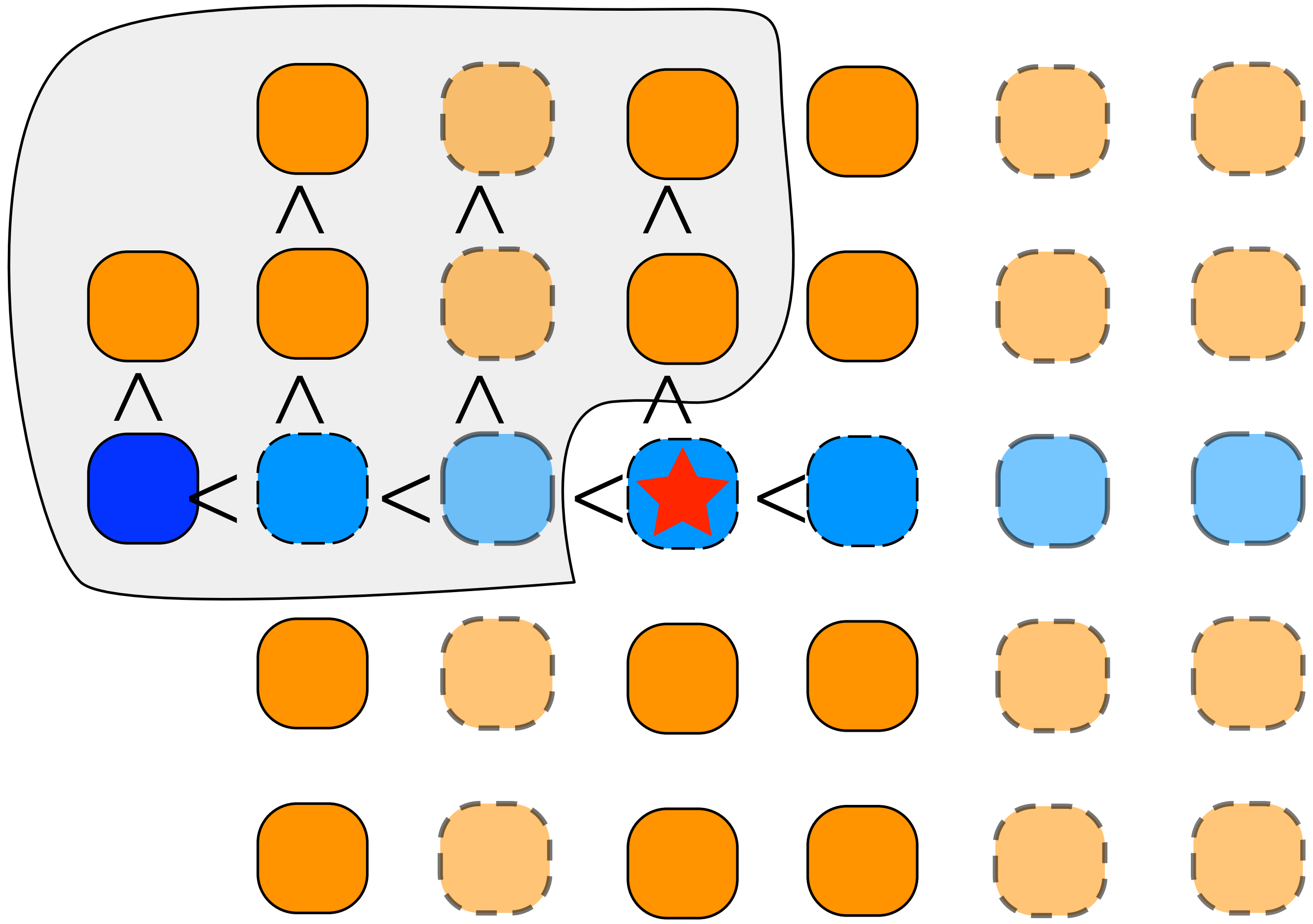
a nice property of our partition



a nice property of our partition

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

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

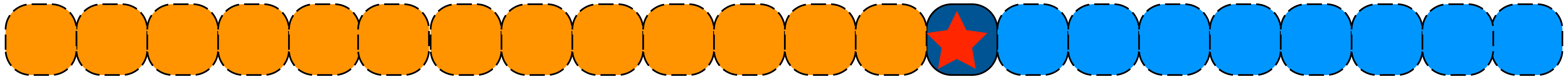


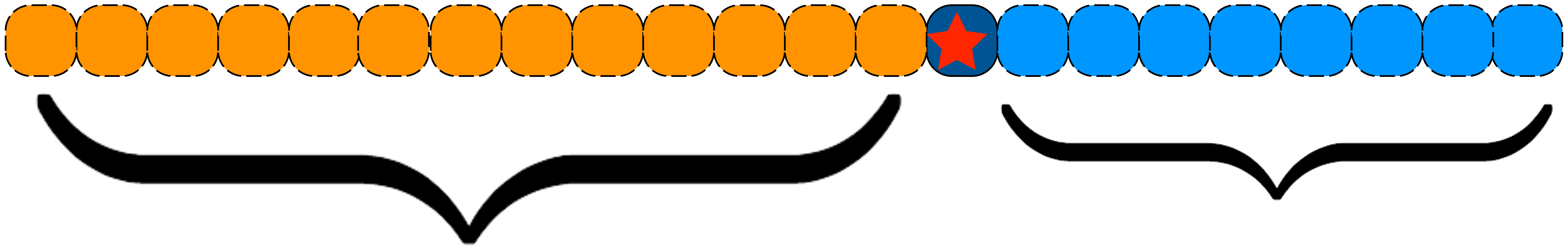
this implies there are

at most $\frac{7n}{10} + 6$ numbers

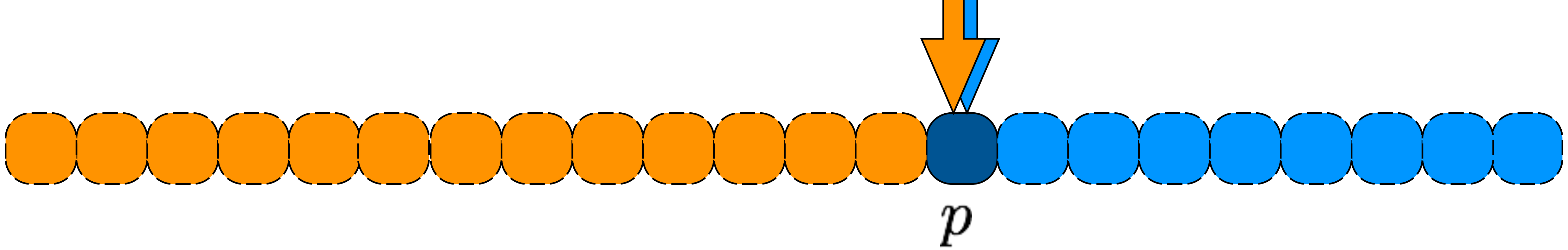
larger than ★
/smaller

a nice property of our partition

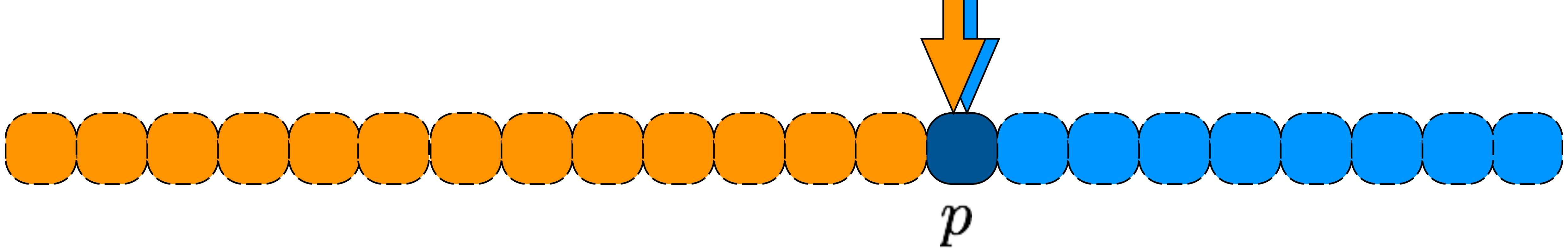




$$\leq \frac{7n}{10} + 6 \qquad\qquad \leq \frac{7n}{10} + 6$$



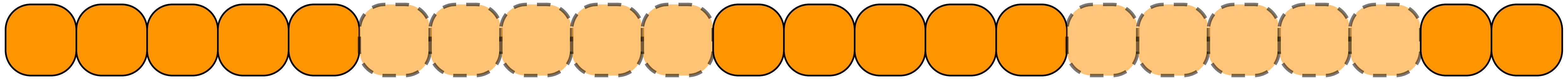
select $(i, A[1, \ldots, n])$



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

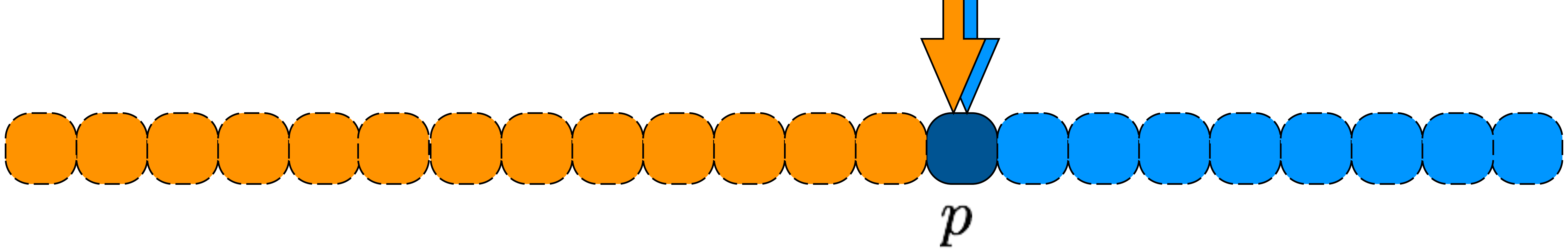
find median of each small list

gather all medians

call select(...) on this sublist to find median

return the result

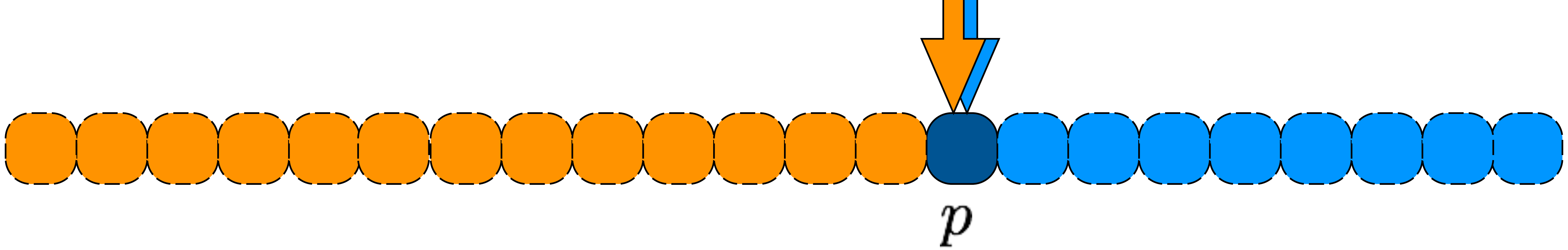
$$P(n) = S(\lceil n/5 \rceil) + O(n)$$



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

$$S(n) = S(\lceil n/5 \rceil) + O(n) + S(7n/10 + 6)$$



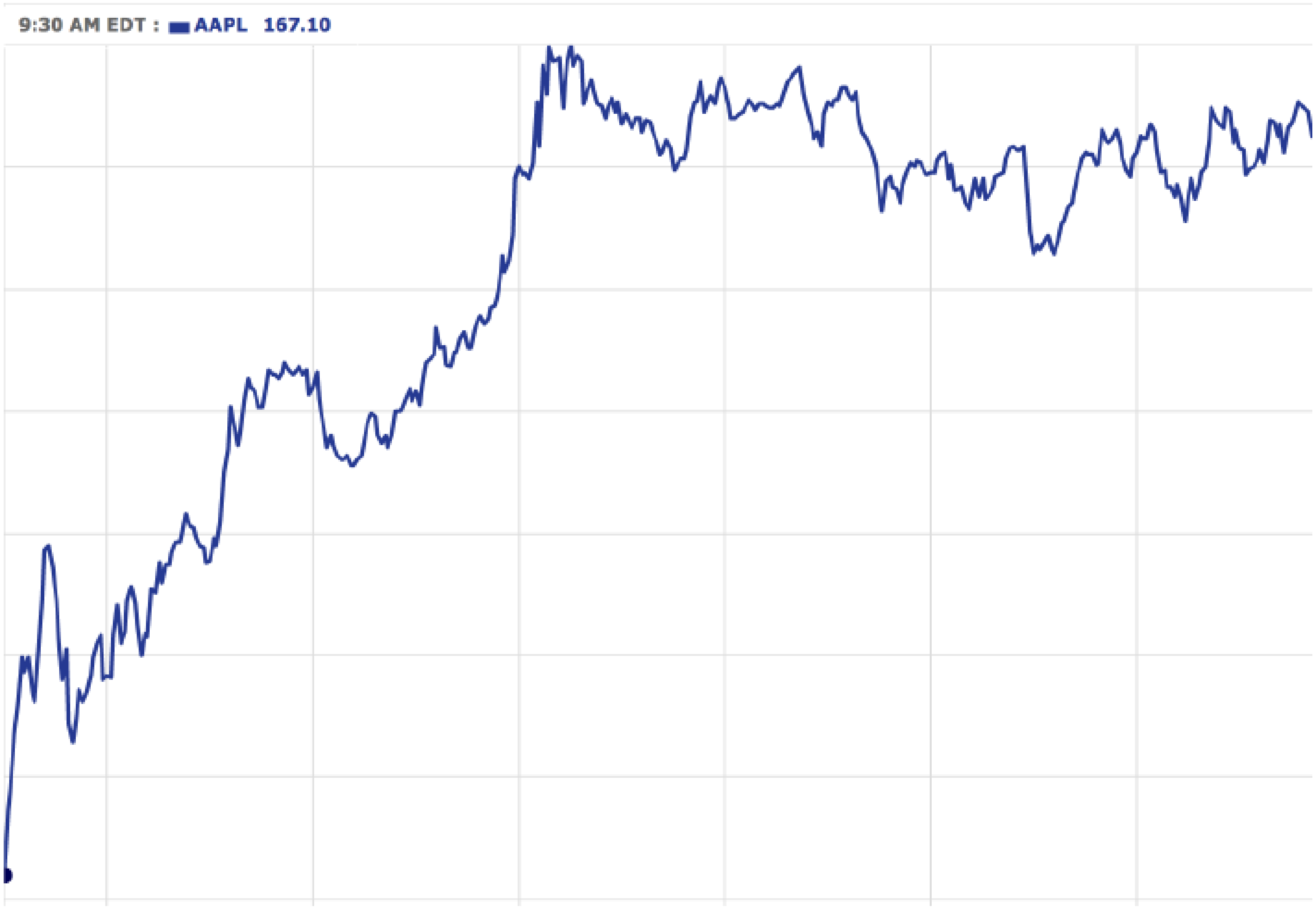
select $(i, A[1, \ldots, n])$

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

$$S(n) = S(\lceil n/5 \rceil) + O(n) + S(7n/10 + 6)$$

$$\Theta(n)$$

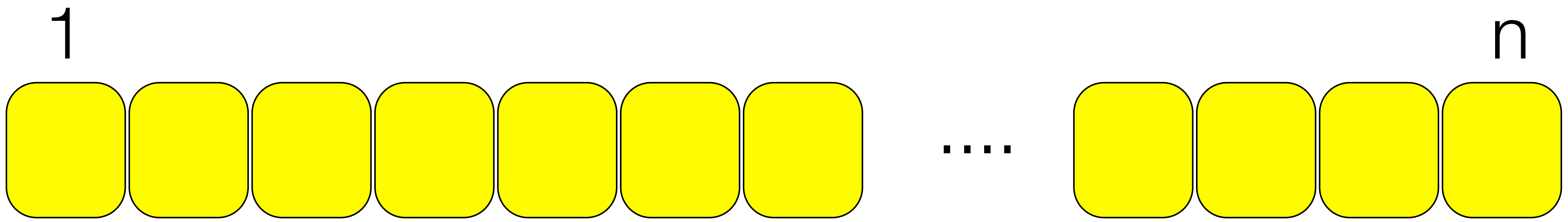
arbitrage


9:30 AM EDT : AAPL 167.10

Jan 14, 2009 :
BPT 72.59
100
90
80
70
60
© 2008 Yahoo! Inc.
2008
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
2009
Feb
Mar

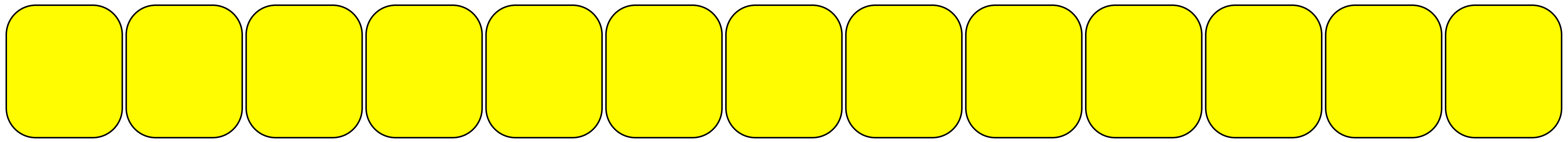

12:38 PM EDT : AIG 40.58
© 2008 Yahoo! Inc.
10:00 AM
11:00 AM
12:00 PM
1:00 PM
2:00 PM
3:00 PM
4:00

input: array of n numbers

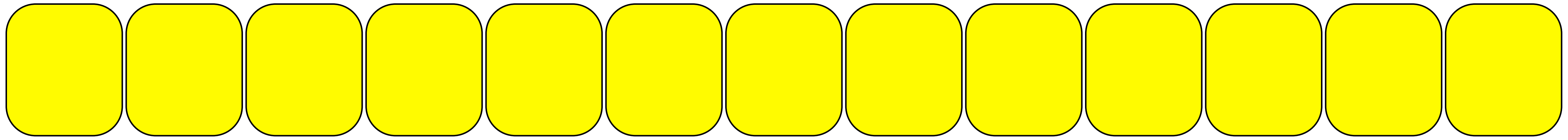


goal:

first attempt

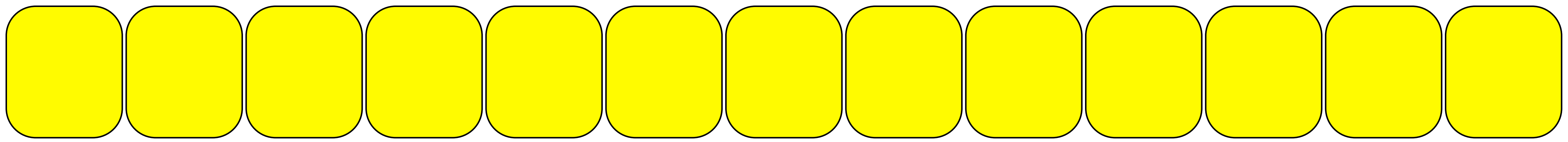


first attempt



```
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

```
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}
```