Hit the LOG

$$n^{\frac{1}{2^L}} = 2$$

FIND L.

$$\log_2\left(n^{\frac{1}{2^L}}\right) = \log_2(2) = 1$$

$$\frac{1}{2^L}\log_2(n) = 1$$

$$\log_2(n) = 2^L$$

$$\log(\log(n)) = \log(2^L) = L$$

L4

shelat
16f-4800
sep 20 2016

$$T(n) = \underbrace{7}_{a}T(n/\underbrace{2}_{b}) + \underbrace{O(n^2)}_{f}$$

① $n^{\log_2 7}$ is $\underset{n^2}{\underset{\shortparallel}{f}}$ in $\underline{O(n^{\overset{2.8}{\underline{\log_2 7}} - \overset{0.01}{\epsilon}})}$? ?

Yes

case 1 applies

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

$$T(n) = 7T\left(\frac{n}{2}\right) + 1$$

f is $\underrightarrow{O\left(n^{\log_2 \overset{7}{a} - \epsilon}\right.}\left.\right)$ so case 1 applies.

$f = 1$

example:

$$T(n) = \underset{\overset{\uparrow}{a}}{T}\left(\underbrace{\frac{14}{17}}_{b}n\right) + \underline{24}$$

$T\left(\frac{n}{17/14}\right)$ $\Theta(1)$

$\underline{f(n) = 24}$ is that $O\left(\underline{n^{\log_{17/14} 1}}\right) = O(n^0) = O(1)$

case 2 applies, b/c

$\Theta\left(n^{\log_{17/14} 1} \cdot \log(n)\right) = \Theta(\log(n))$

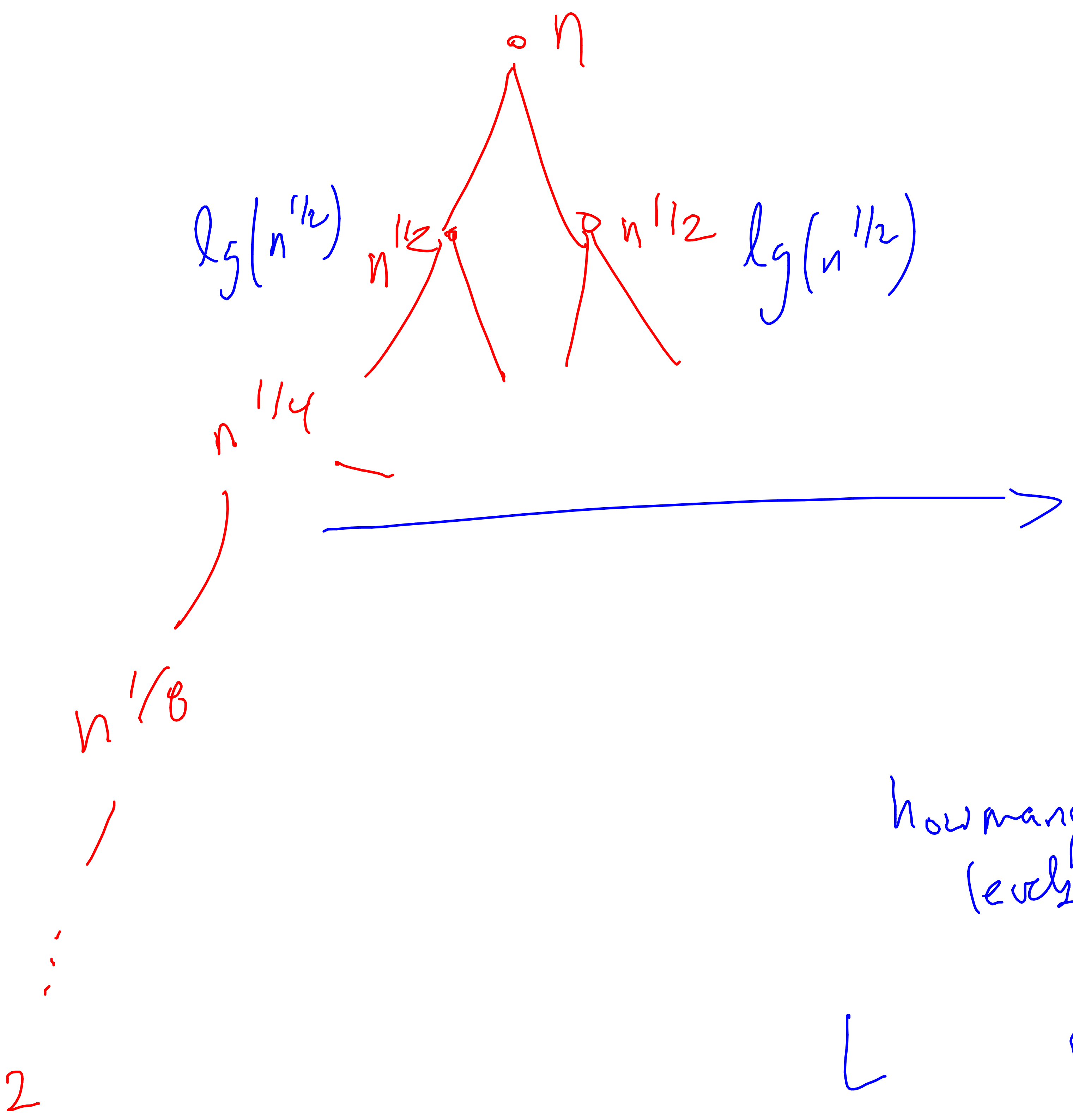
$n^0 = 1$

f is in $\Theta\left(n^{\log_b a}\right) = \Theta(1)$

$o(1)$

8
13
FIFA

$$T(n) = 2T(\sqrt{n}) + \lg n$$



$\lg n$

$\frac{1}{2}\lg(n) + \frac{1}{2}\lg(n) = \underline{\lg(n)}$

$= \underline{\lg(n)}$

how many levels.

$L \quad n^{\frac{1}{2^L}} = 2 \Rightarrow \underline{L = \log\log(n)}$ leves.

$$T(n) = 2T\left(\sqrt{n}\right) + \lg n$$

$$T(n) = O\left(\log n \cdot \log\log(n)\right)$$

$$T(n) = 2T(\sqrt{n}) + \lg n$$

① $T(2^m) = T(n)$

$\lg(2^m)$

$$T(2^m) = 2T(2^{m/2}) + c\cdot m$$

$m = \log(n)$

$$S(m) = 2S(m/2) + c\cdot m$$

② $S(m) = T(2^m) = T(n)$

$\sqrt{2^m} = 2^{m/2}$

$S(m) = \Theta(m\cdot\log m)$ by case 2 of Masters.

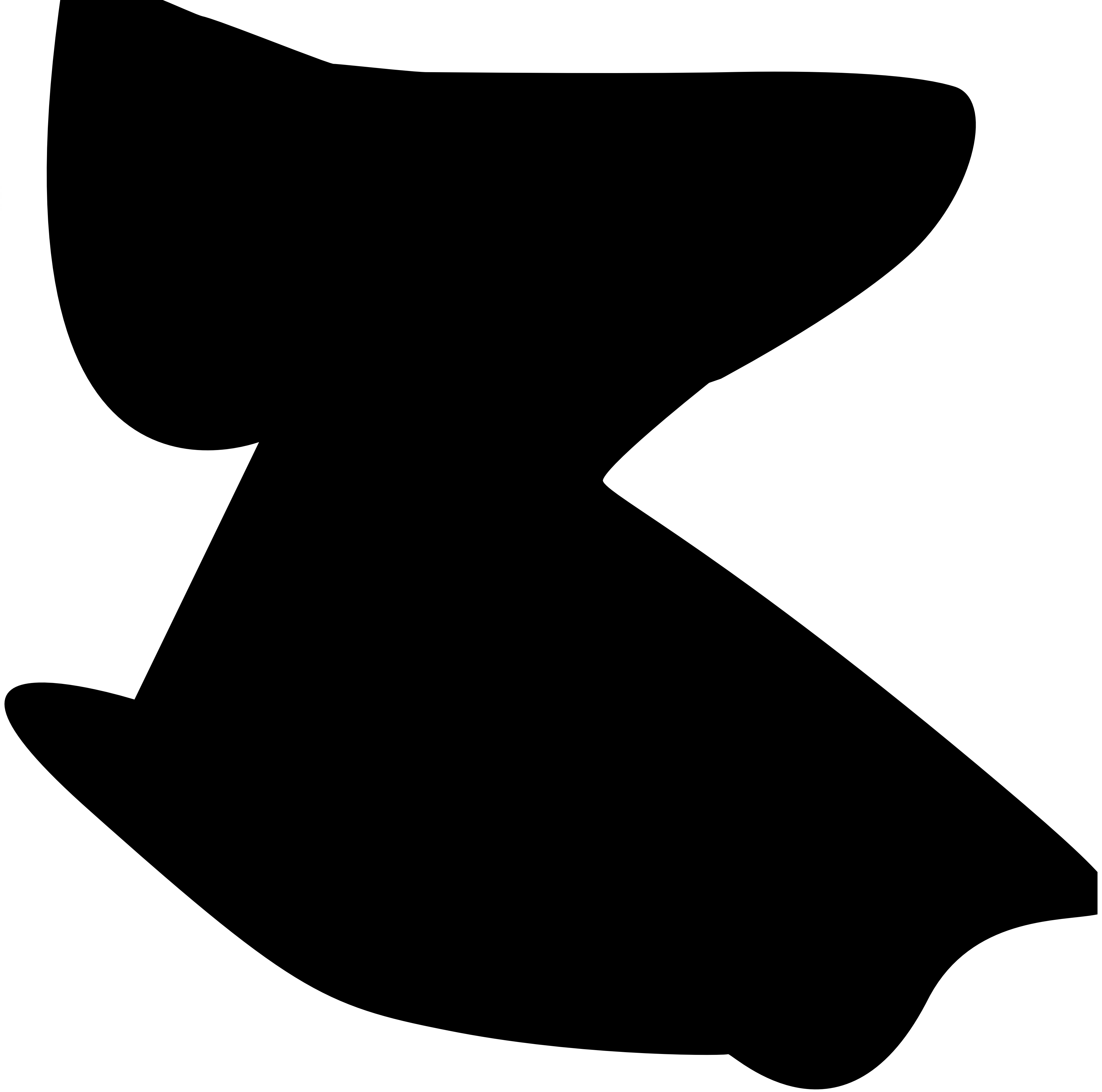
$||$

$$T(n) = \Theta(\log(n)\cdot\log(\log(n)))$$

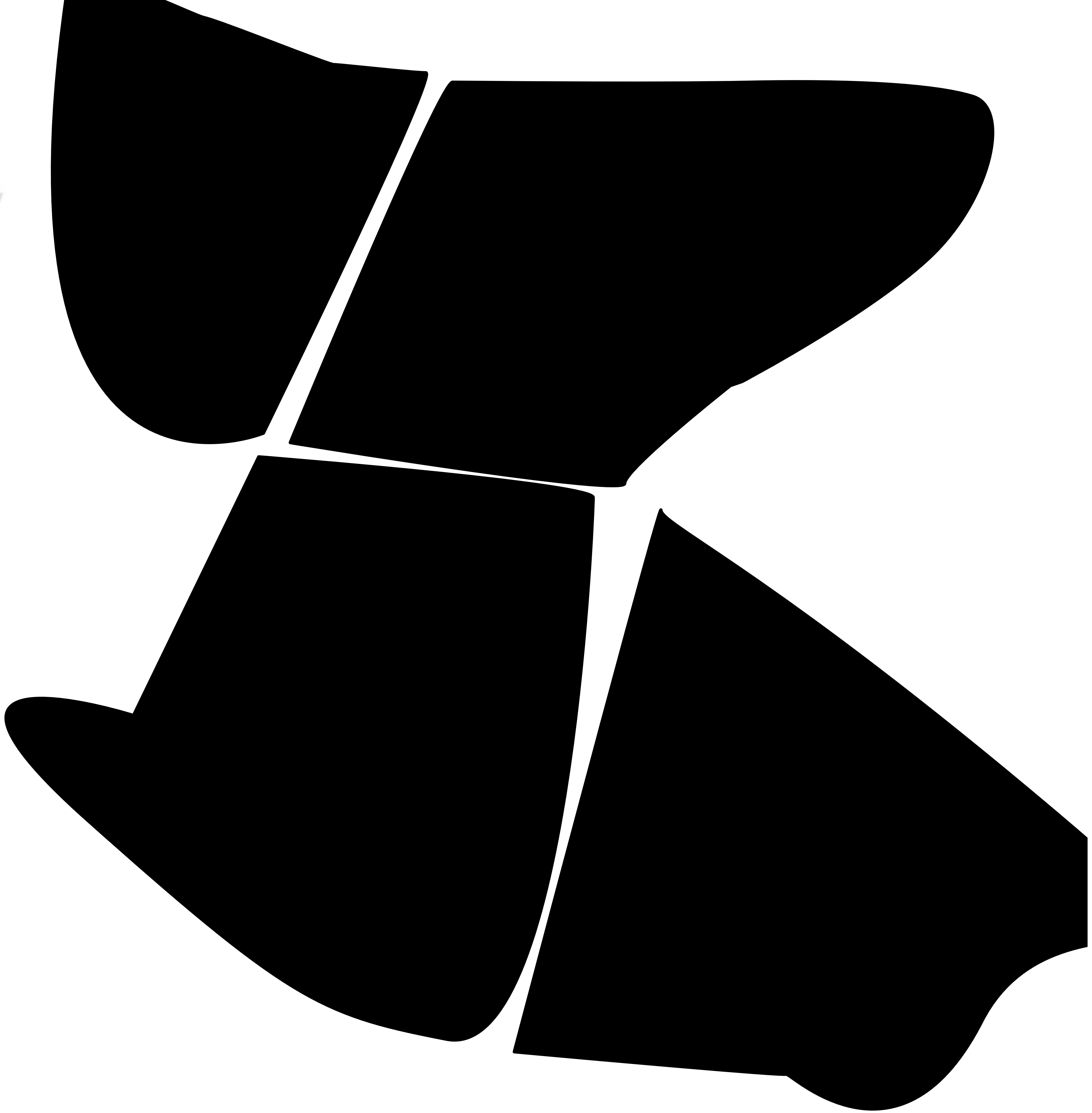
$S(m/2) = T(2^{m/2})$

$$T(m) = 2T\left(\frac{m}{2} + 13\right) +$$

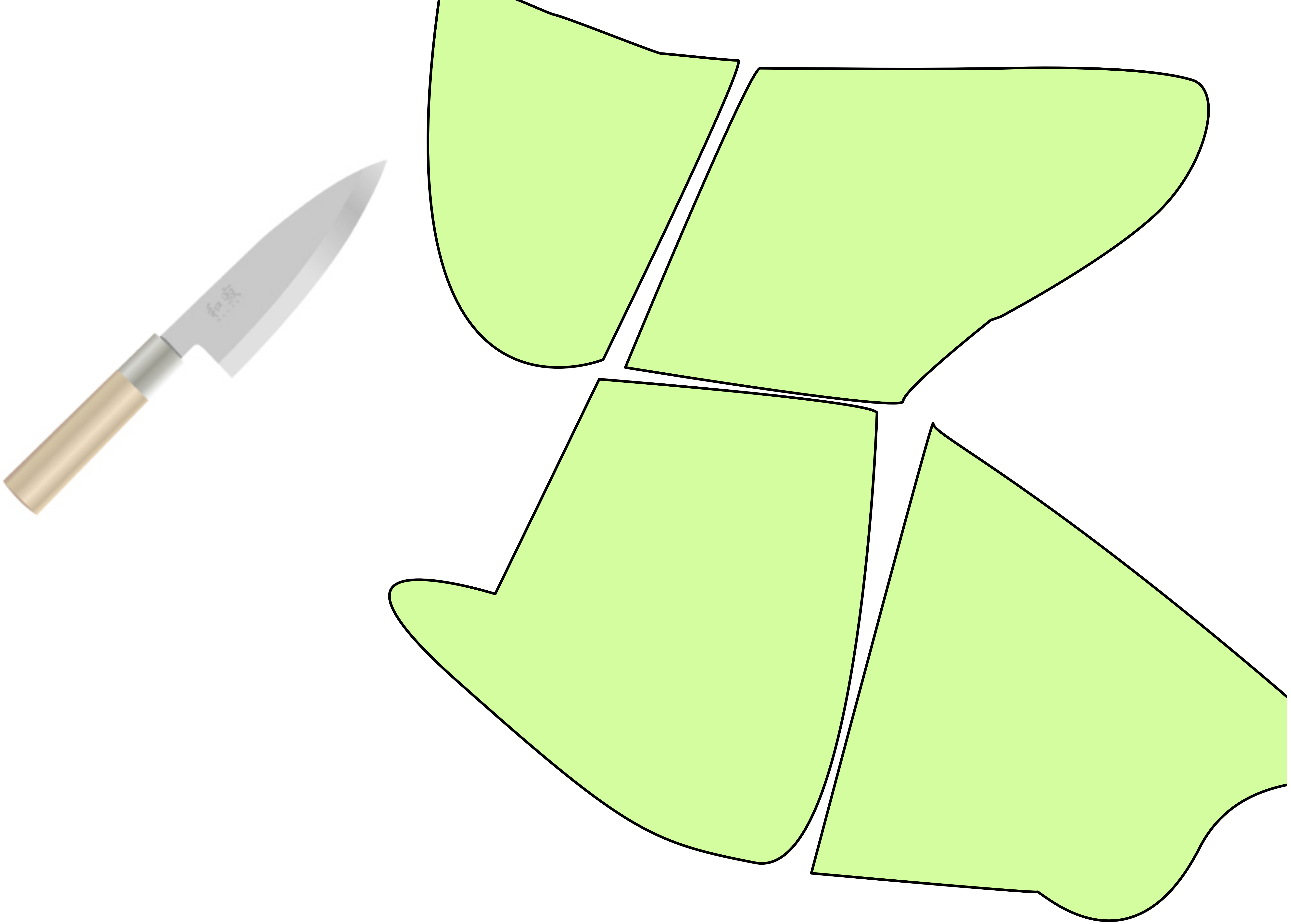
divide
& conquer



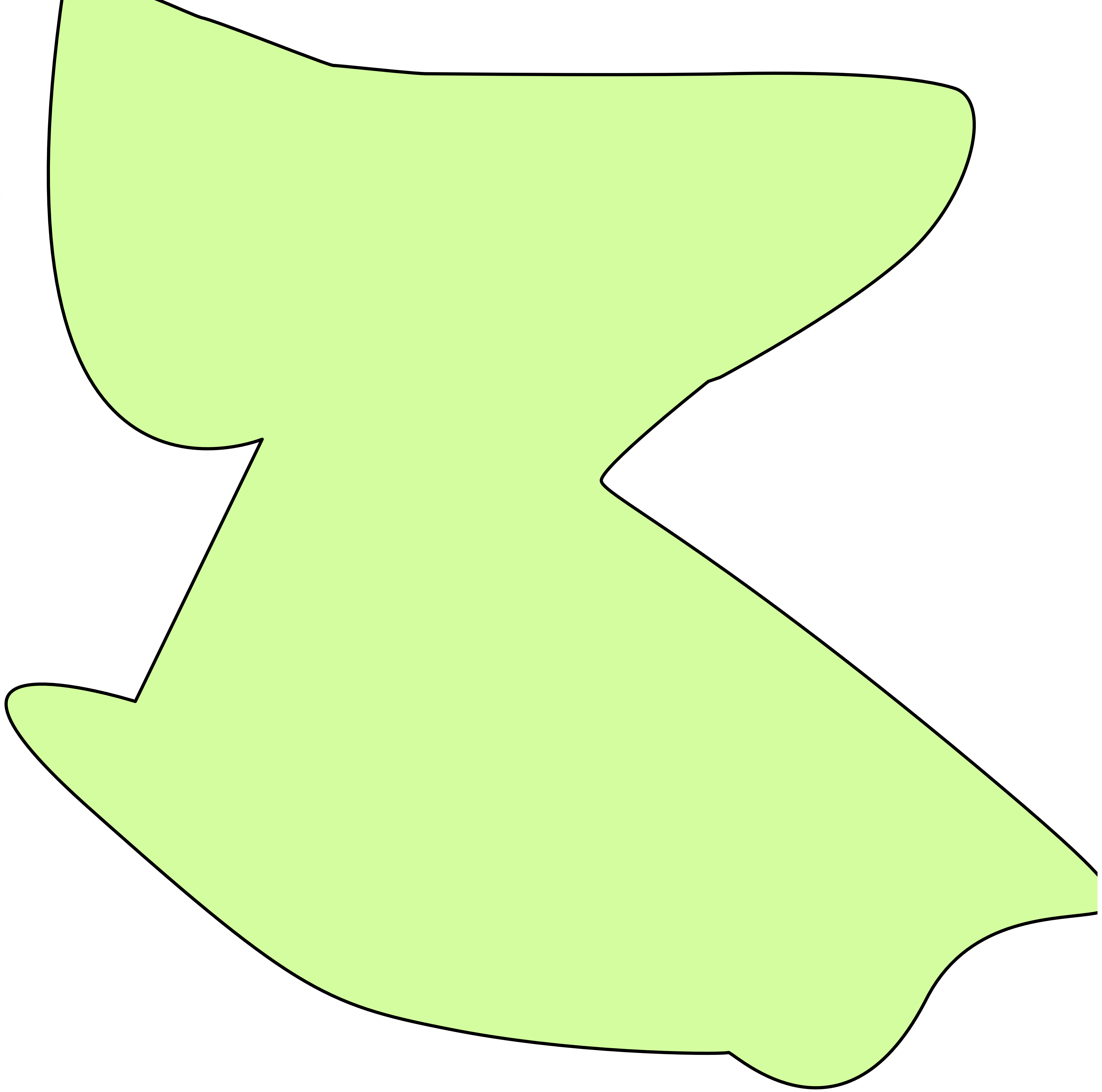
http://www.kitchenknifedrawer.com/files/1696205/uploaded/K6615D.jpg



http://www.kitchenknifedrawer.com/files/1696205/uploaded/K6615D.jpg



http://www.kitchenknifedrawer.com/files/1696205/uploaded/K6615D.jpg

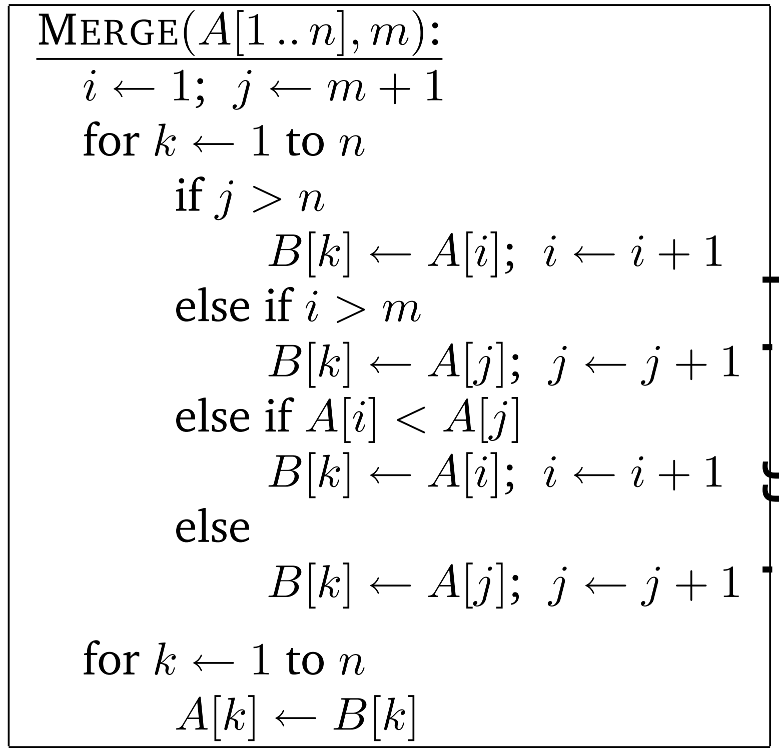


http://www.kitchenknifedrawer.com/files/1696205/uploaded/K6615D.jpg

examples

Merge

```
merge-sort $(A, p, r)$
    if $p < r$
        $q \leftarrow \lfloor (p + r)/2 \rfloor$
        merge-sort $(A, p, q)$
        merge-sort $(A, q + 1, r)$
        merge$(A, p, q, r)$
```

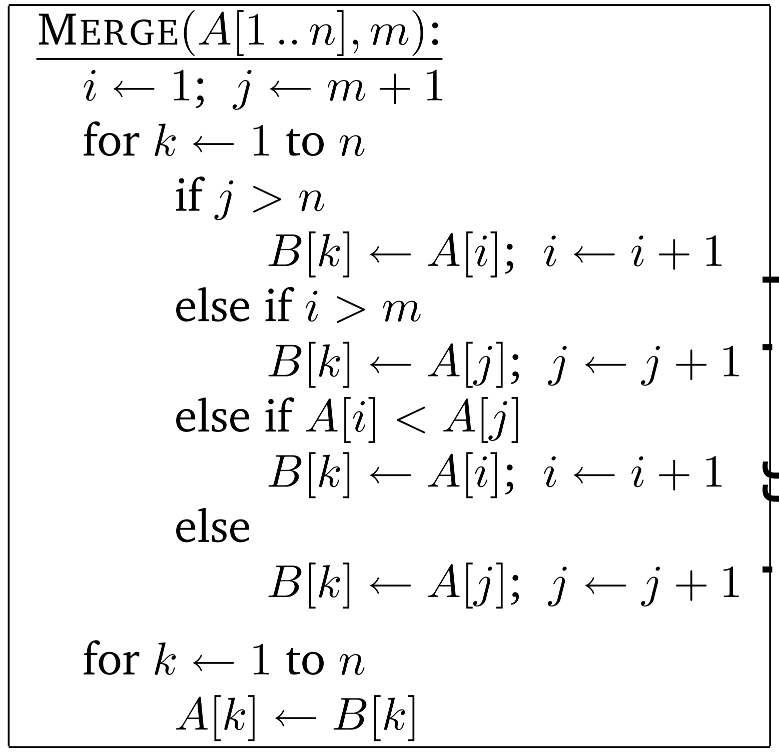


```
MERGE$(A[1..n], m)$:
    $i \leftarrow 1$;  $j \leftarrow m + 1$
    for $k \leftarrow 1$ to $n$
        if $j > n$
            $B[k] \leftarrow A[i]$;  $i \leftarrow i + 1$
        else if $i > m$
            $B[k] \leftarrow A[j]$;  $j \leftarrow j + 1$
        else if $A[i] < A[j]$
            $B[k] \leftarrow A[i]$;  $i \leftarrow i + 1$
        else
            $B[k] \leftarrow A[j]$;  $j \leftarrow j + 1$
    for $k \leftarrow 1$ to $n$
        $A[k] \leftarrow B[k]$
```

jeff erickson

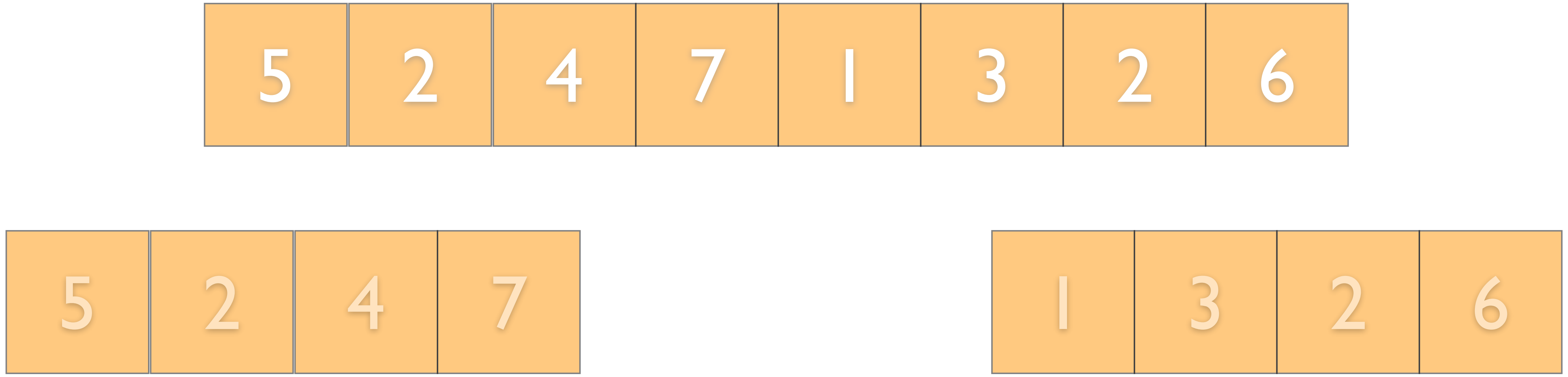
5	2	4	7	1	3	2	6

```
merge-sort $(A, p, r)$
  if $p < r$
      $q \leftarrow \lfloor (p+r)/2 \rfloor$
      merge-sort $(A, p, q)$
      merge-sort $(A, q+1, r)$
      merge$(A, p, q, r)$
```



```
MERGE$(A[1..n], m)$:
  $i \leftarrow 1$; $j \leftarrow m+1$
  for $k \leftarrow 1$ to $n$
      if $j > n$
          $B[k] \leftarrow A[i]$; $i \leftarrow i+1$
      else if $i > m$
          $B[k] \leftarrow A[j]$; $j \leftarrow j+1$
      else if $A[i] < A[j]$
          $B[k] \leftarrow A[i]$; $i \leftarrow i+1$
      else
          $B[k] \leftarrow A[j]$; $j \leftarrow j+1$
  for $k \leftarrow 1$ to $n$
      $A[k] \leftarrow B[k]$
```

jeff erickson



5	2	4	7	1	3	2	6

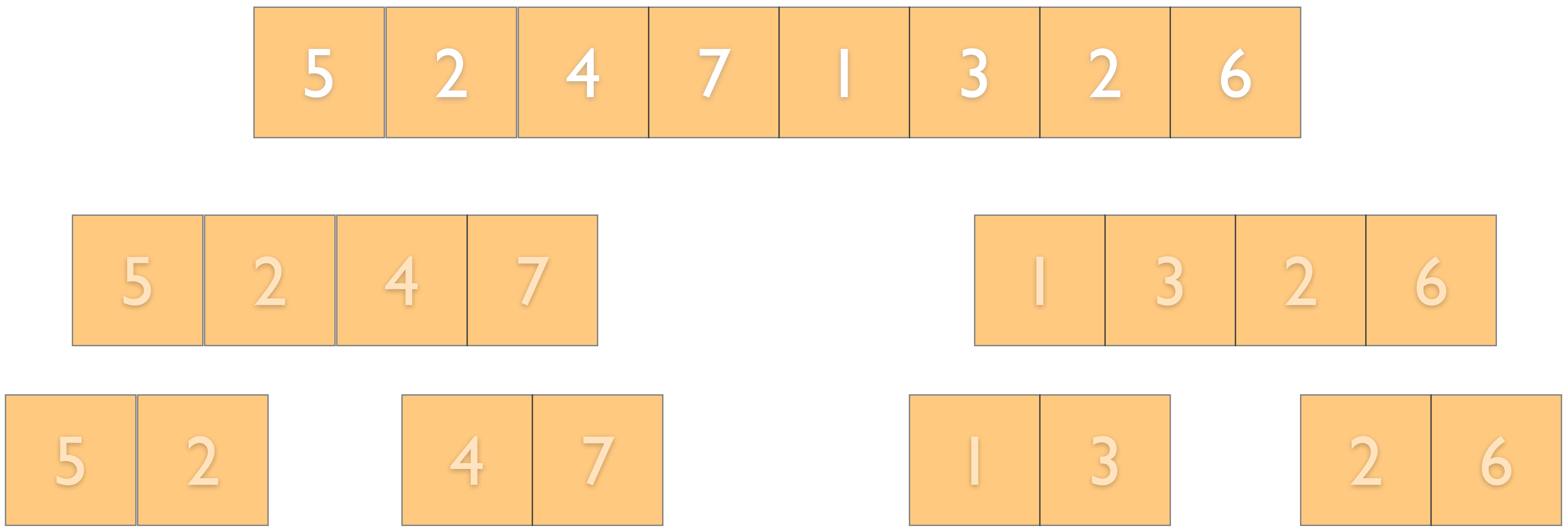
5	2	4	7

1	3	2	6

```
merge-sort $(A, p, r)$
    if $p < r$
        $q \leftarrow \lfloor (p + r)/2 \rfloor$
        merge-sort $(A, p, q)$
        merge-sort $(A, q + 1, r)$
        merge$(A, p, q, r)$
```

```
MERGE$(A[1\,..\,n], m)$:
    $i \leftarrow 1$;  $j \leftarrow m + 1$
    for $k \leftarrow 1$ to $n$
        if $j > n$
            $B[k] \leftarrow A[i]$;  $i \leftarrow i + 1$
        else if $i > m$
            $B[k] \leftarrow A[j]$;  $j \leftarrow j + 1$
        else if $A[i] < A[j]$
            $B[k] \leftarrow A[i]$;  $i \leftarrow i + 1$
        else
            $B[k] \leftarrow A[j]$;  $j \leftarrow j + 1$
    for $k \leftarrow 1$ to $n$
        $A[k] \leftarrow B[k]$
```

jeff erickson

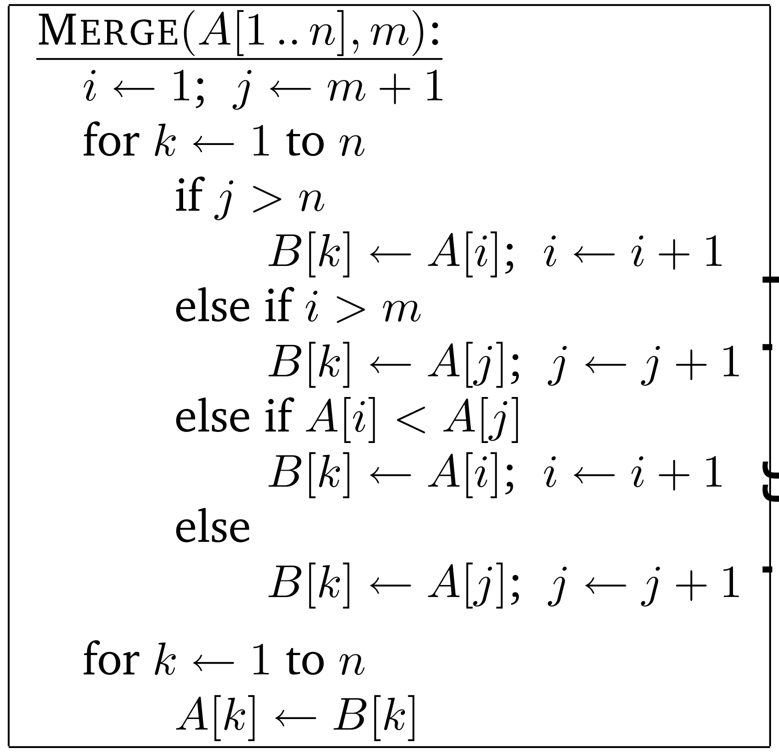


5 2 4 7 1 3 2 6

5 2 4 7   1 3 2 6

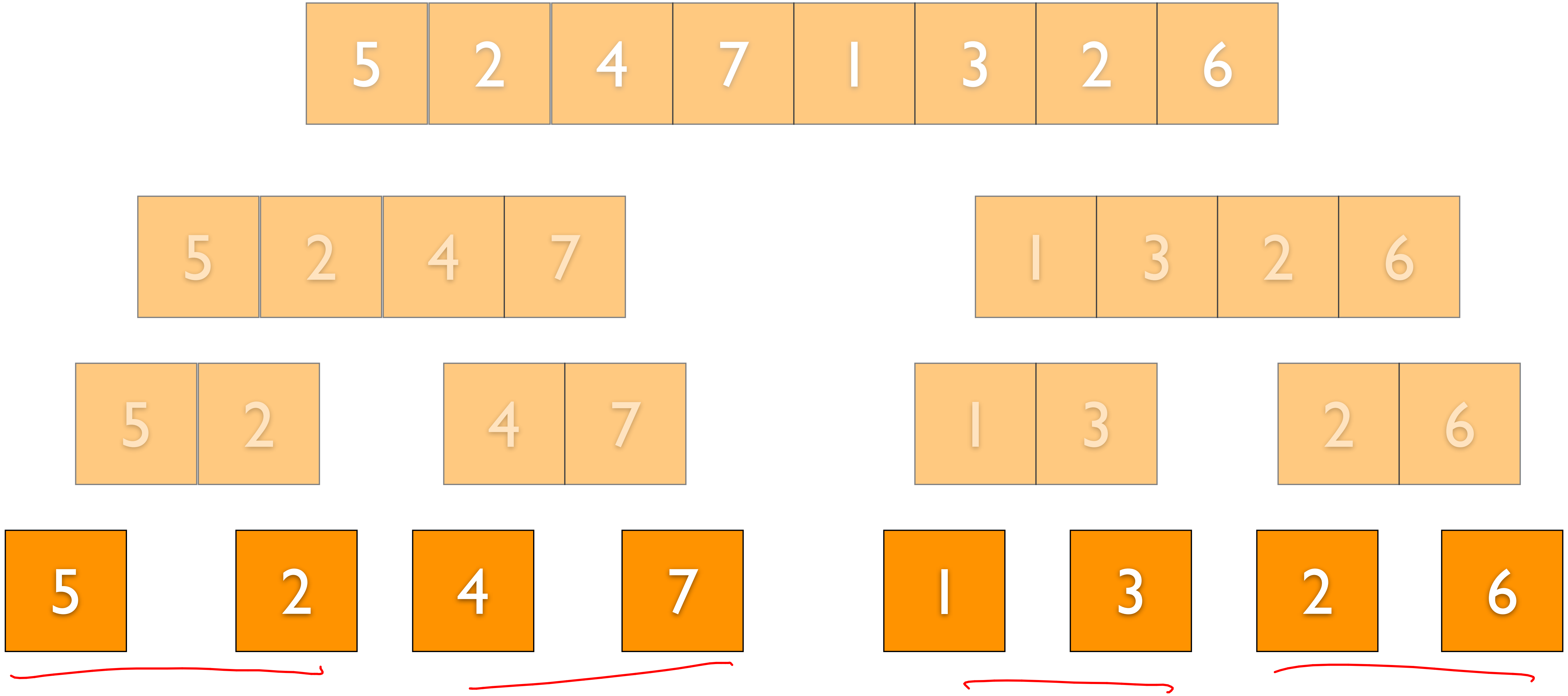
5 2   4 7   1 3   2 6

```
merge-sort $(A, p, r)$
    if $p < r$
        $q \leftarrow \lfloor (p+r)/2 \rfloor$
        merge-sort $(A, p, q)$
        merge-sort $(A, q+1, r)$
        merge$(A, p, q, r)$
```

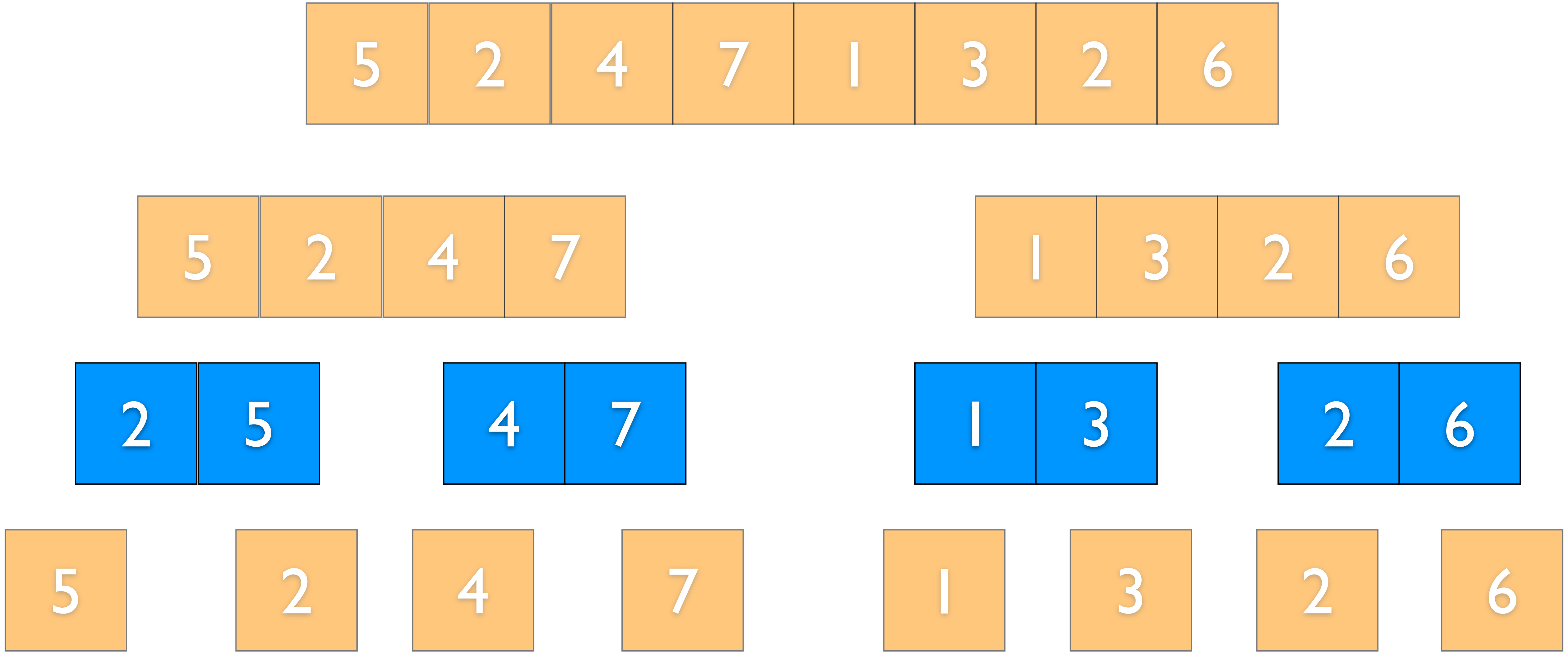


```
MERGE$(A[1..n], m)$:
    $i \leftarrow 1$; $j \leftarrow m+1$
    for $k \leftarrow 1$ to $n$
        if $j > n$
            $B[k] \leftarrow A[i]$; $i \leftarrow i+1$
        else if $i > m$
            $B[k] \leftarrow A[j]$; $j \leftarrow j+1$
        else if $A[i] < A[j]$
            $B[k] \leftarrow A[i]$; $i \leftarrow i+1$
        else
            $B[k] \leftarrow A[j]$; $j \leftarrow j+1$

    for $k \leftarrow 1$ to $n$
        $A[k] \leftarrow B[k]$
```

jeff erickson



```
merge-sort $(A, p, r)$
    if $p < r$
        $q \leftarrow \lfloor (p + r)/2 \rfloor$
        merge-sort $(A, p, q)$
        merge-sort $(A, q + 1, r)$
        merge$(A, p, q, r)$
```



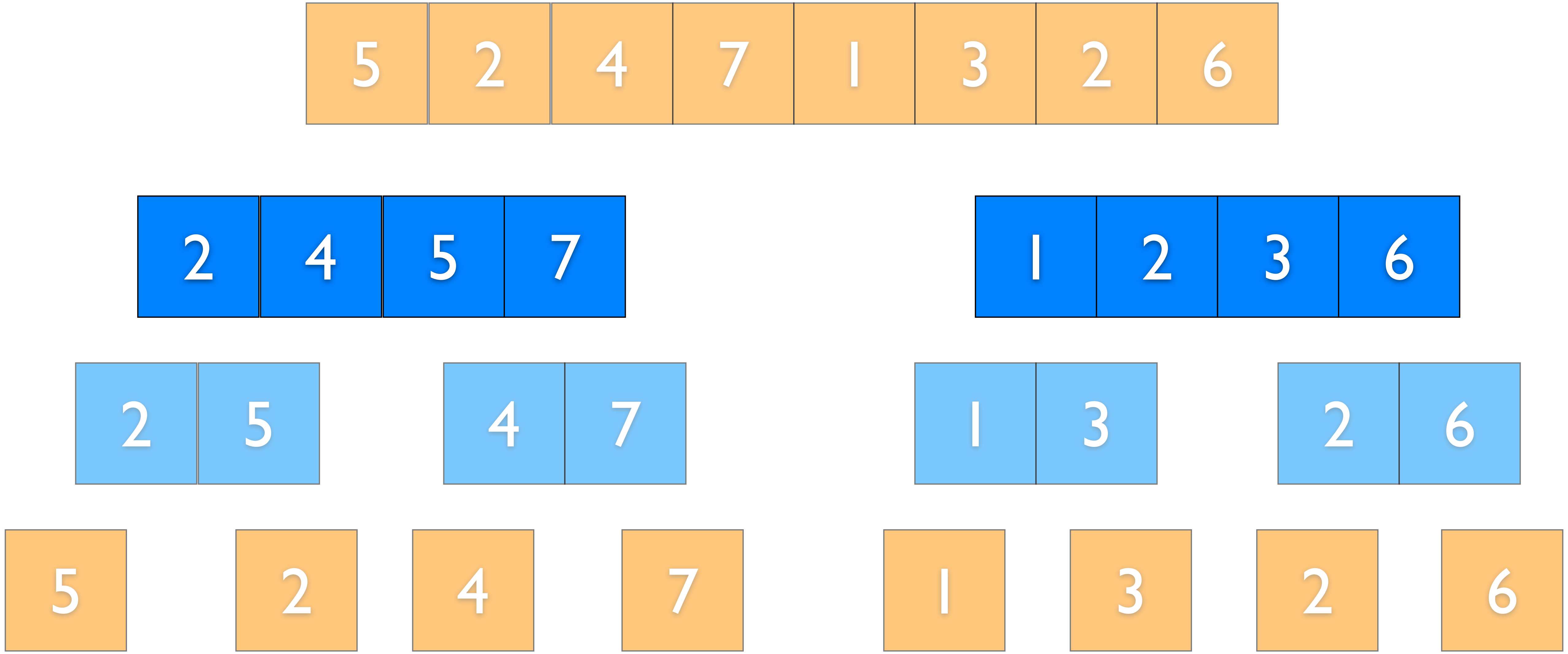
5 2 4 7 1 3 2 6

5 2 4 7 1 3 2 6

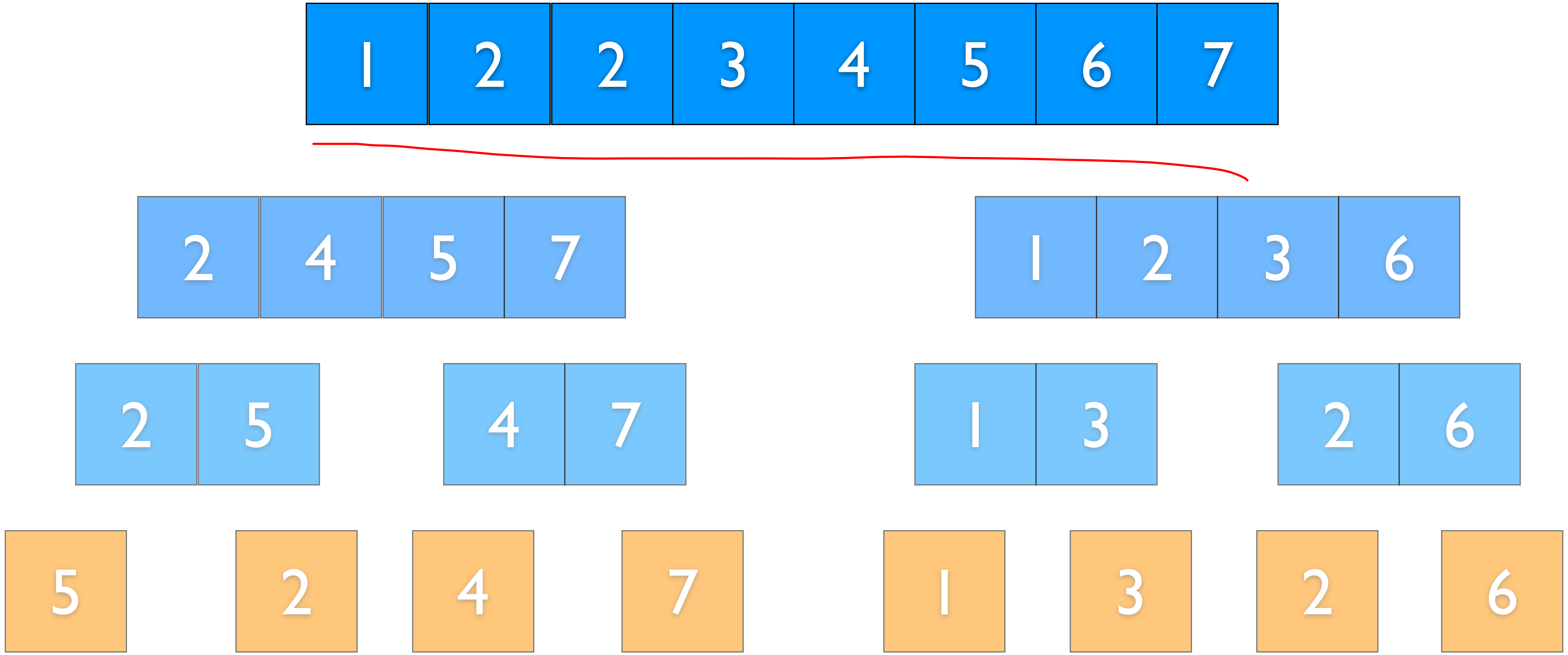
2 5 4 7 1 3 2 6

5 2 4 7 1 3 2 6

```
merge-sort $(A, p, r)$
    if $p < r$
        $q \leftarrow \lfloor (p+r)/2 \rfloor$
        merge-sort $(A, p, q)$
        merge-sort $(A, q+1, r)$
        merge$(A, p, q, r)$
```



```
merge-sort $(A, p, r)$
    if $p < r$
        $q \leftarrow \lfloor (p+r)/2 \rfloor$
        merge-sort $(A, p, q)$
        merge-sort $(A, q+1, r)$
        merge$(A, p, q, r)$
```

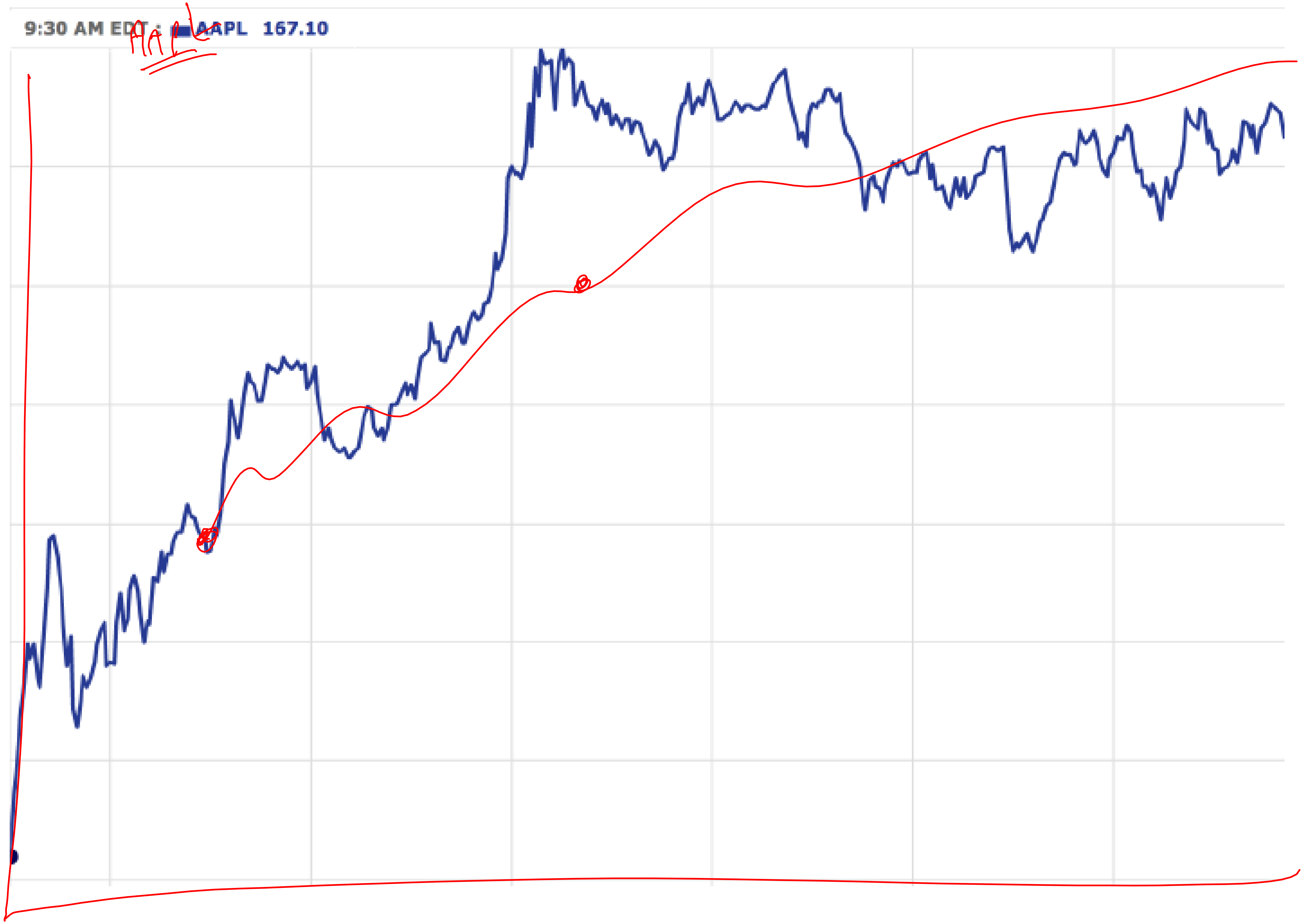


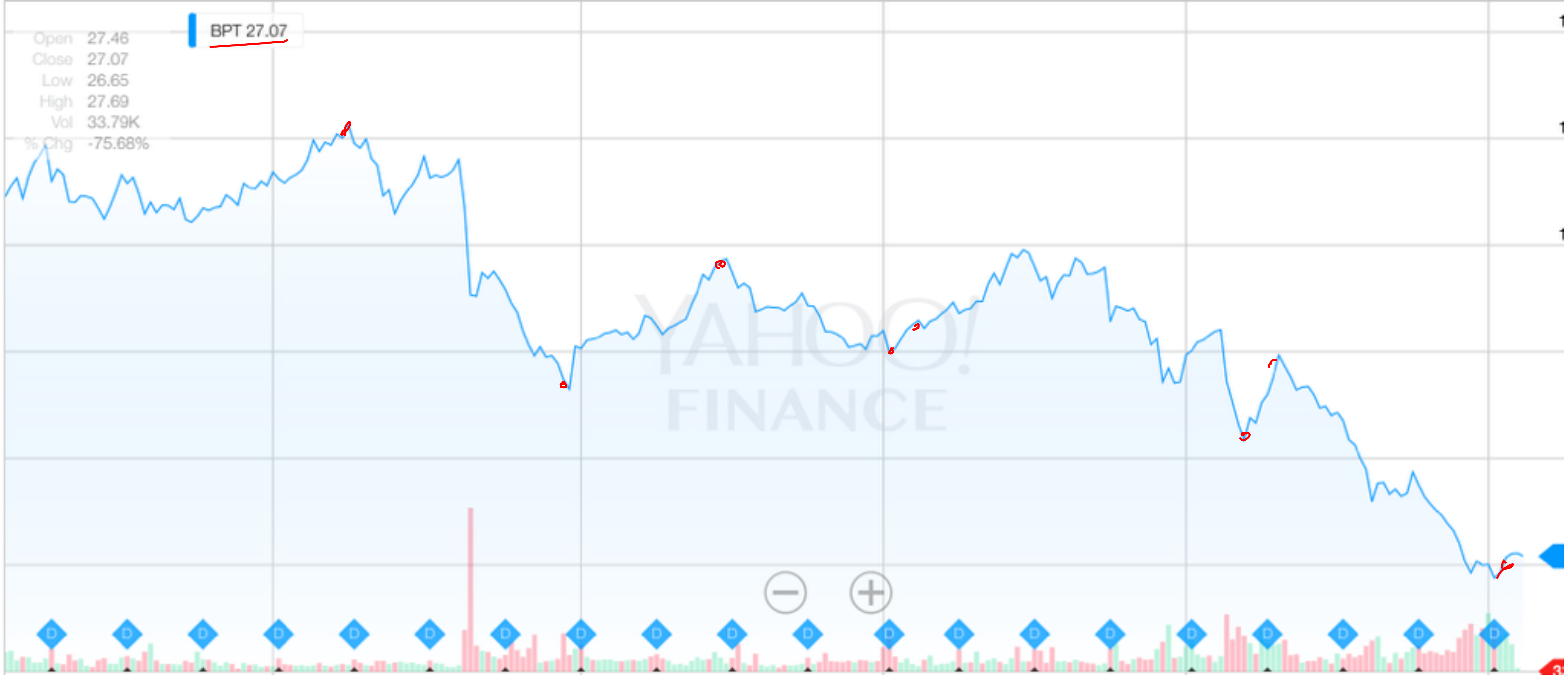
```
merge-sort $(A, p, r)$
    if $p < r$
        $q \leftarrow \lfloor (p + r)/2 \rfloor$
        merge-sort $(A, p, q)$ → $T(n/2)$
        merge-sort $(A, q + 1, r)$ → $T(n/2)$
        merge$(A, p, q, r)$ → $\Theta(n)$
```

$$\underline{T(n) = 2T(n/2) + \Theta(n)}$$

$$= \underline{\Theta(n \log n)}$$

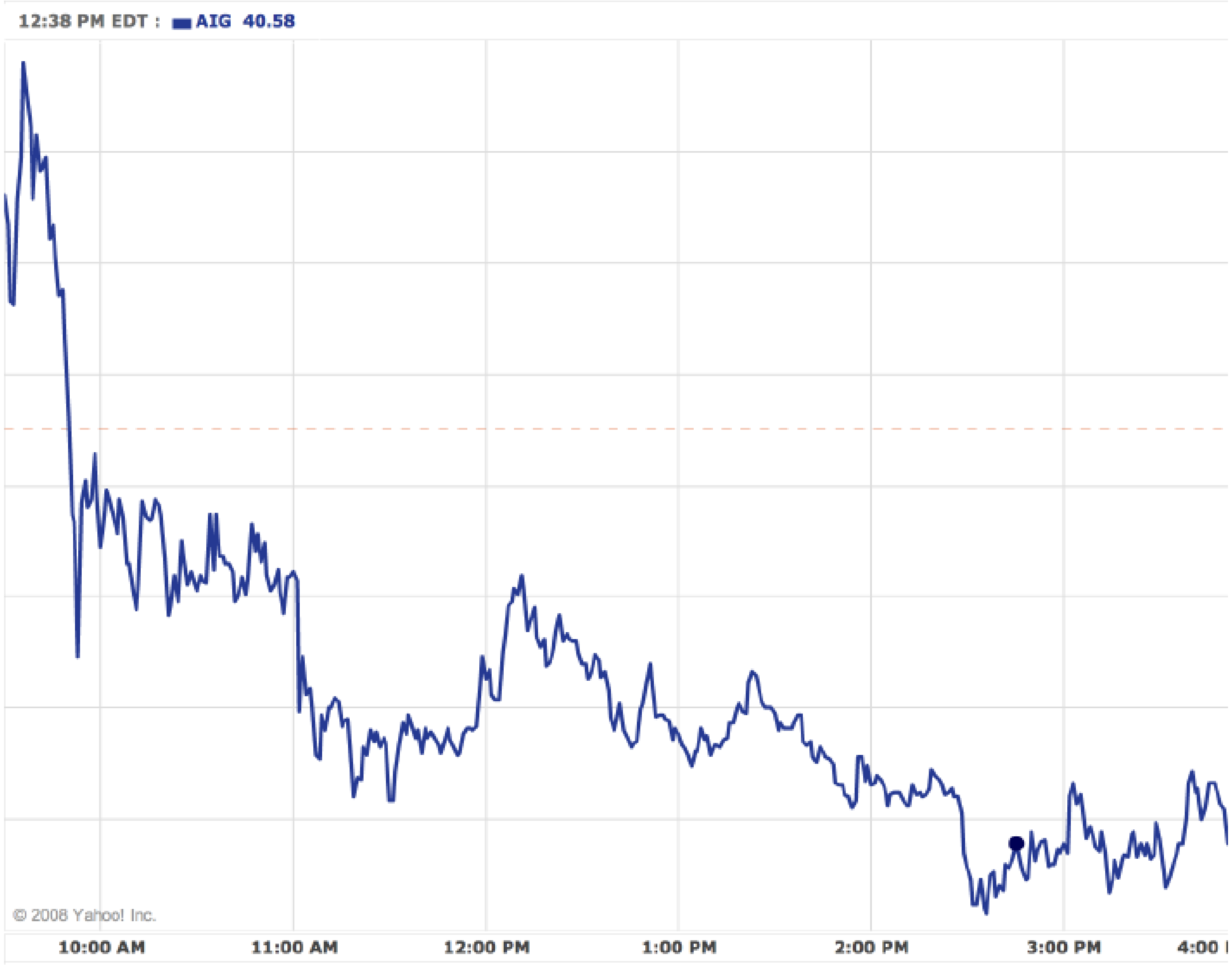
arbitrage


9:30 AM EDT : ■ AAPL 167.10
AAPL

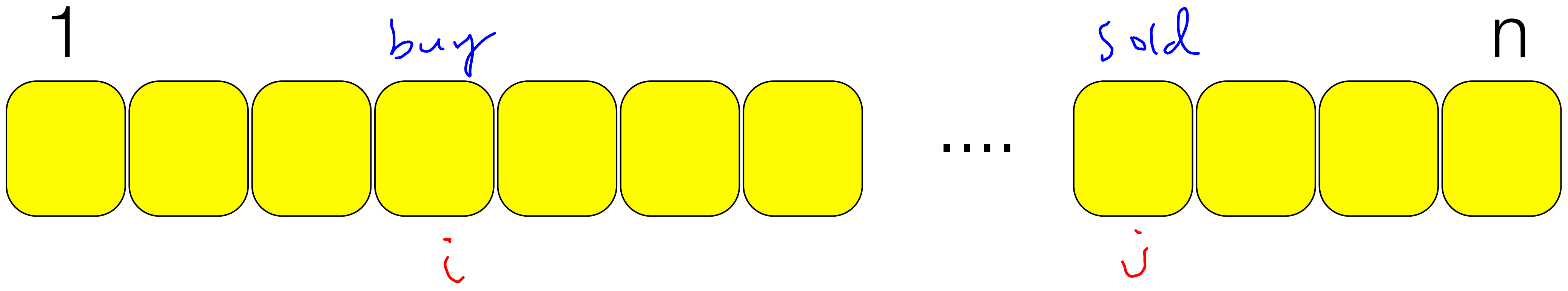


BPT 27.07

Open 27.46
Close 27.07
Low 26.65
High 27.69
Vol 33.79K
% Chg -75.68%


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

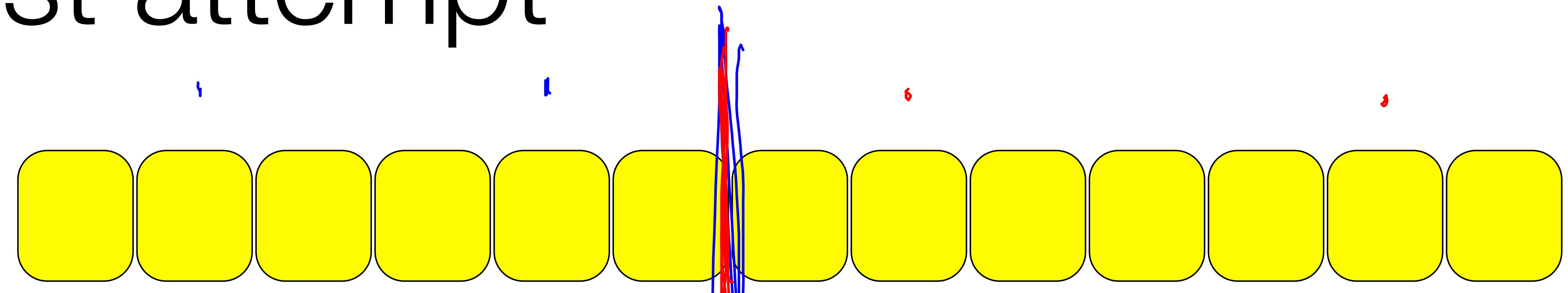
input: array of n numbers



goal: index i, j such that $\underline{i < j}$

which maximizes $\underbrace{A[j] - A[i]}_{\text{profit}}$

first attempt



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

handle base case

$(i^l, j^l\ \underline{bt^l})$ on the left = $\text{arbit}(A[1..n/2])$ // max profit from a trade that begins & ends on the left

(i^r, j^r, bt^r) on the right = $\text{arbil}(A[n/2+1...n])$

$(i^*, j^*\ bt^b) = \max(A[n/2+1, n]) - \min(A[1...n/2])$

return $\max(\underset{\underline{i^l\ j^l}}{bt^l}, \underset{\underline{i^r\ j^r}}{bt^r}, \underset{\underline{i^*\ j^*}}{bt^b})$

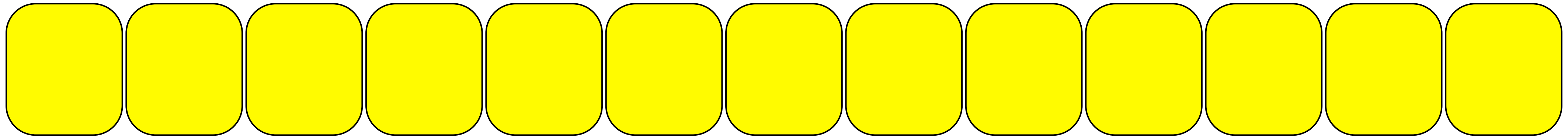
first attempt

```
arbit(A[1...n])
    base case if |A|<=2
    lg = arbit(left(A))  —— $T(n/2)$
    rg = arbit(right(A)) —— $T(n/2)$
    minl = min(left(A))  → $\Theta(n)$
    maxr = max(right(A)) → $\Theta(n)$
    return max{maxr-minl,lg,rg}  $\Theta(1)$
```

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

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

can we do better??



```
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}
```

take $\Theta(n)$ time.

Can we figure these values out in less time???

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}
```

second attempt

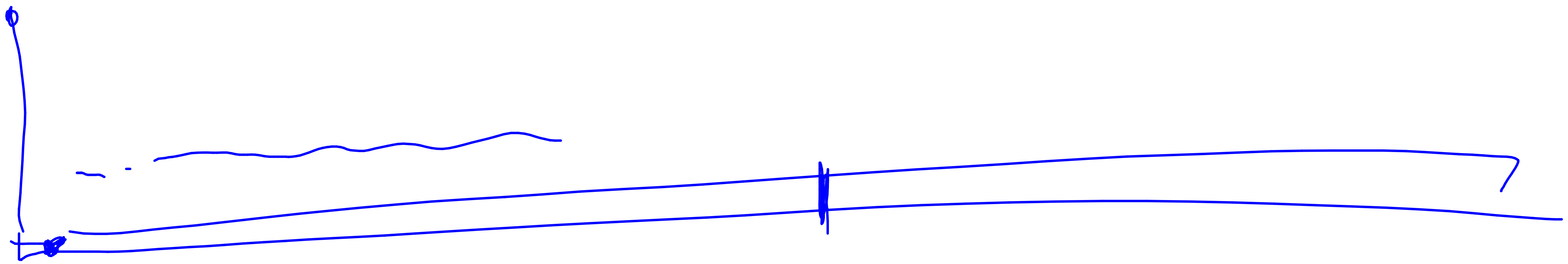
```
arbit+(A[1...n])
   base case if |A|<=2
```

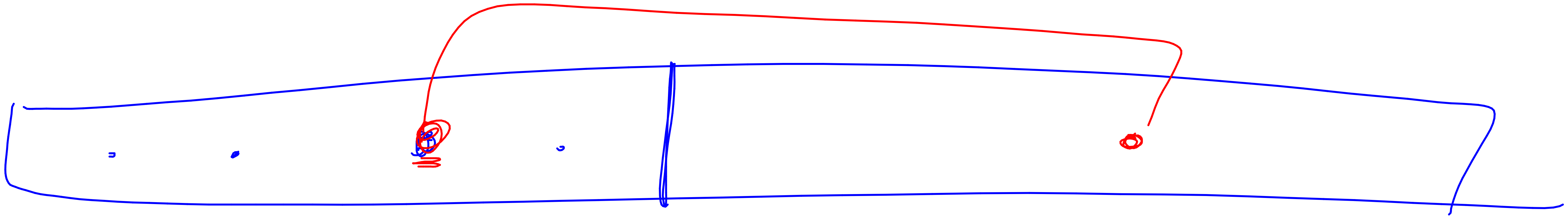
second attempt

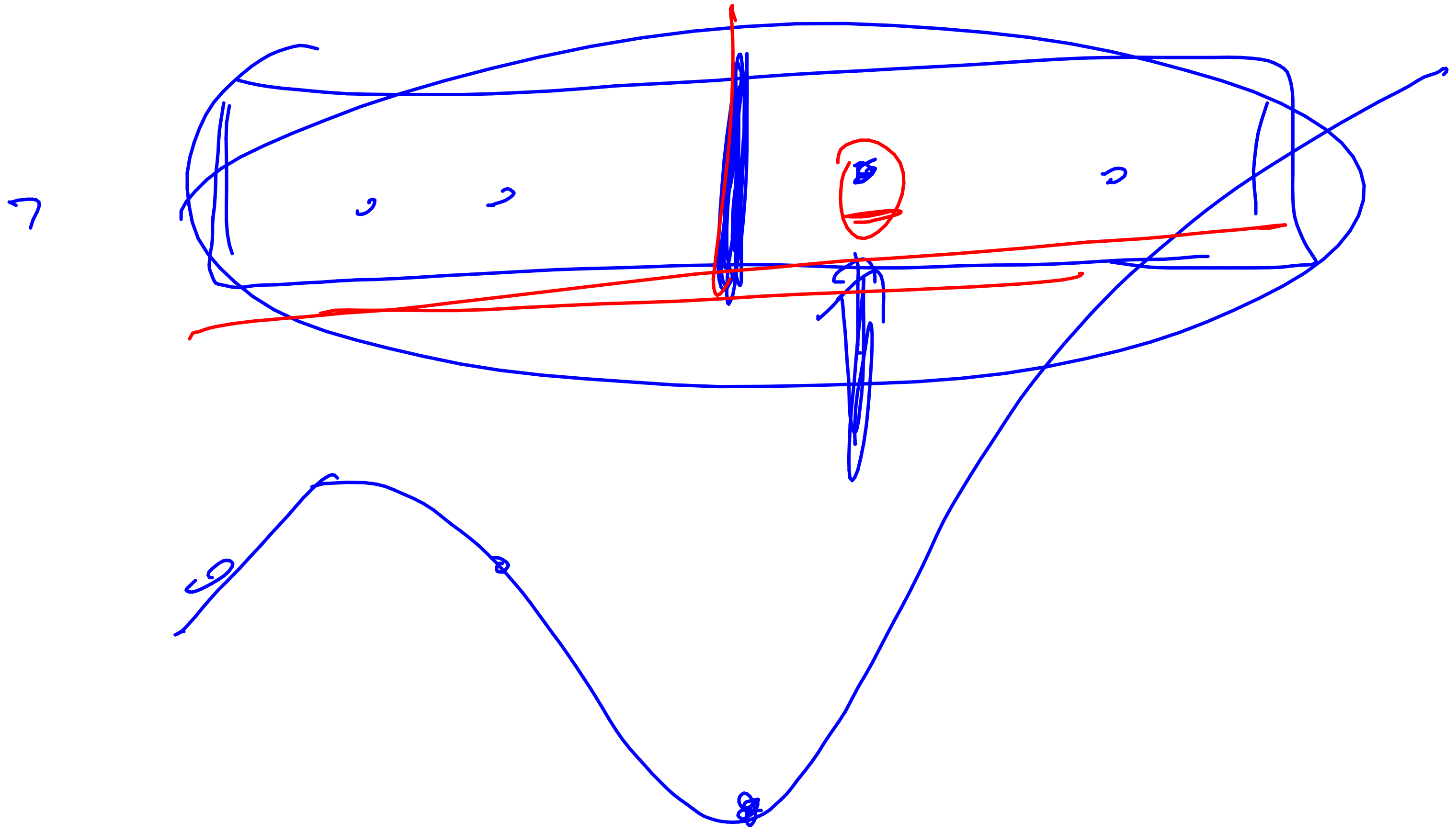
```
arbit+(A[1...n])
 base case if |A|<=2, …
 (lg,minl,maxl) = arbit(left(A))
 (rg,minr,maxr) = arbit(right(A))
   return max{maxr-minl,lg,rg},
          min{minl, minr},
          max{maxl, maxr}
```

$\Theta(n)$ algorithm b/c

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



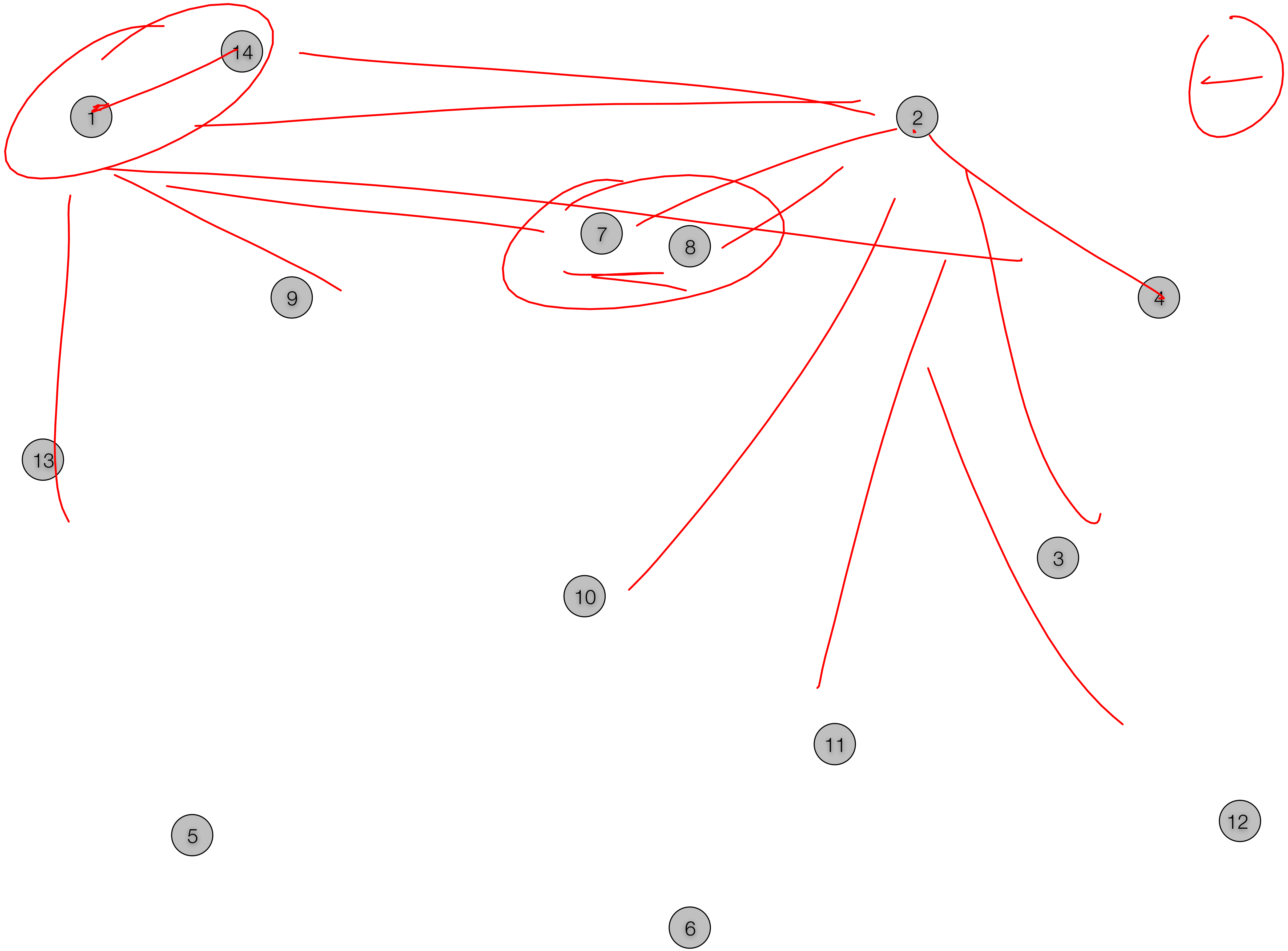




closest pair

of points

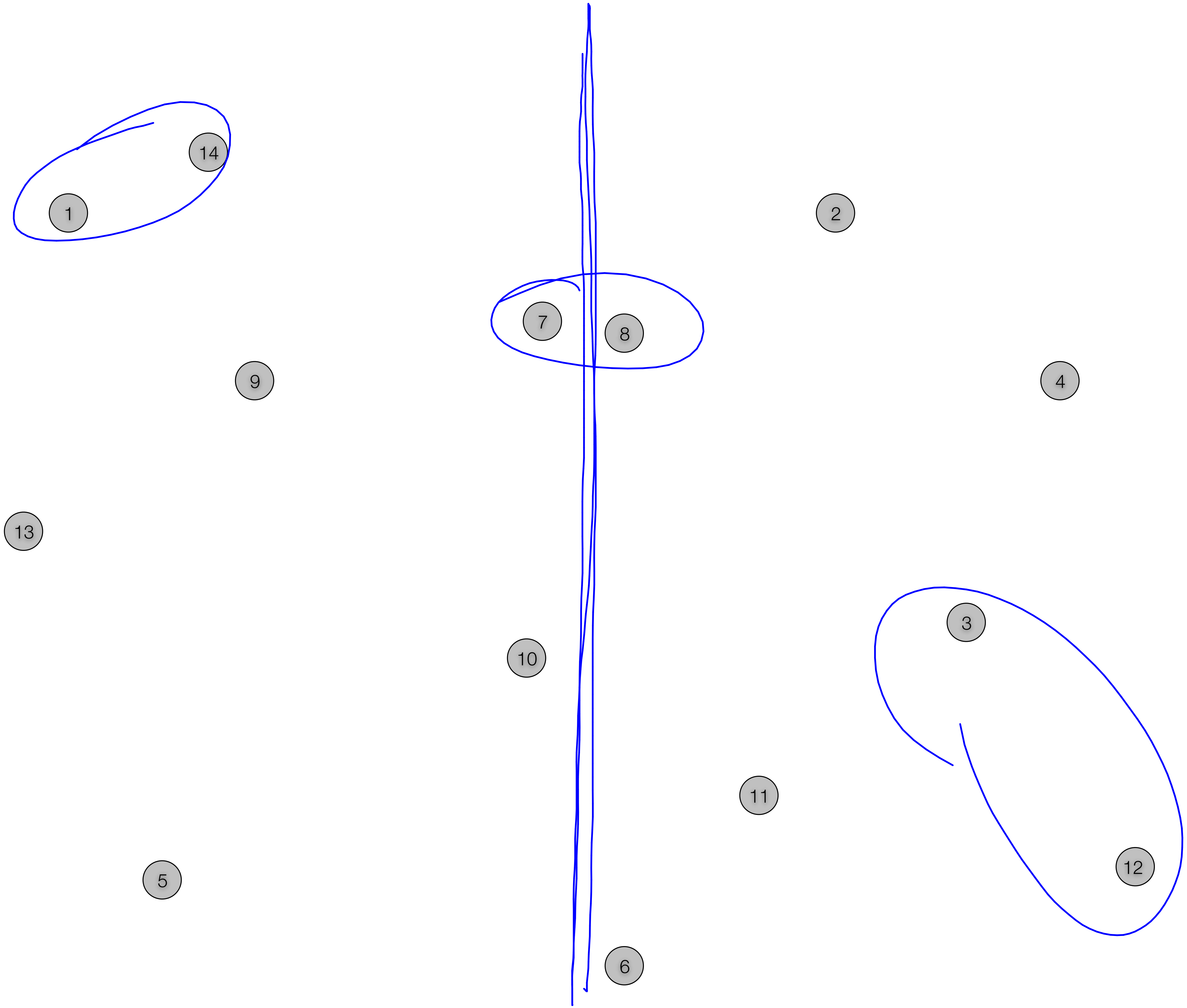
simple brute force approach takes



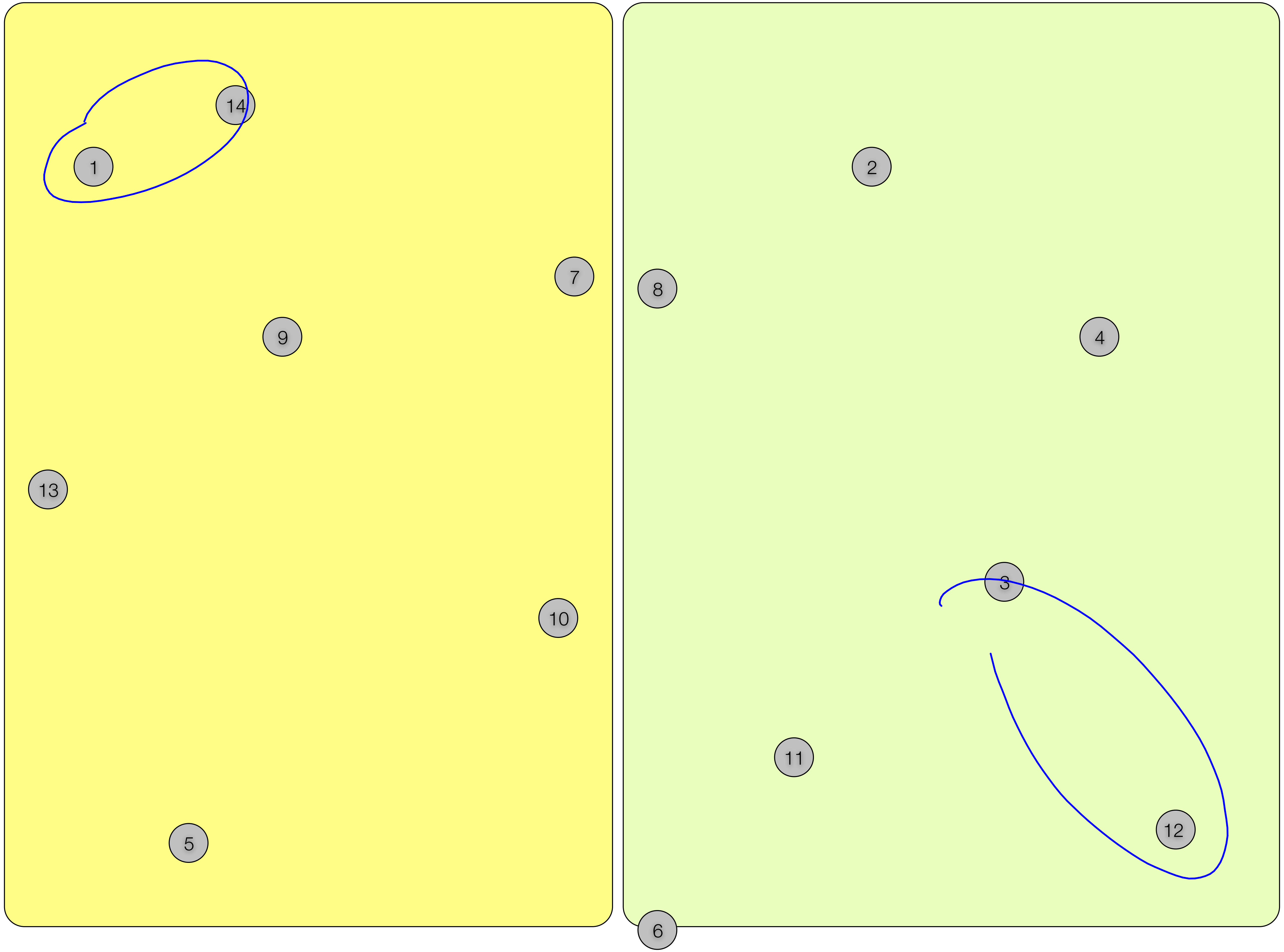
$\Theta(n^2)$

solve the large problem by

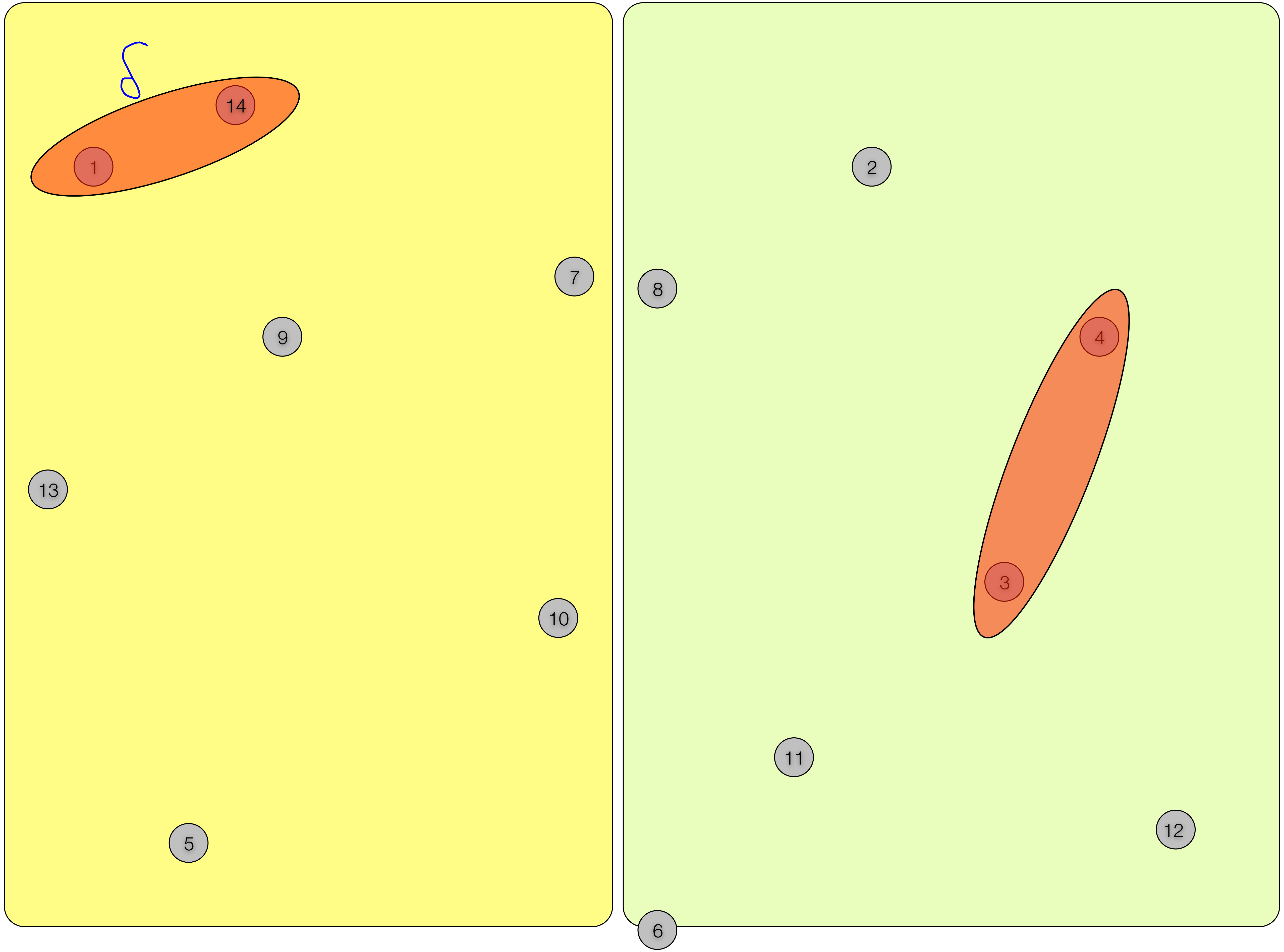
solving smaller problems
and combining solutions


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

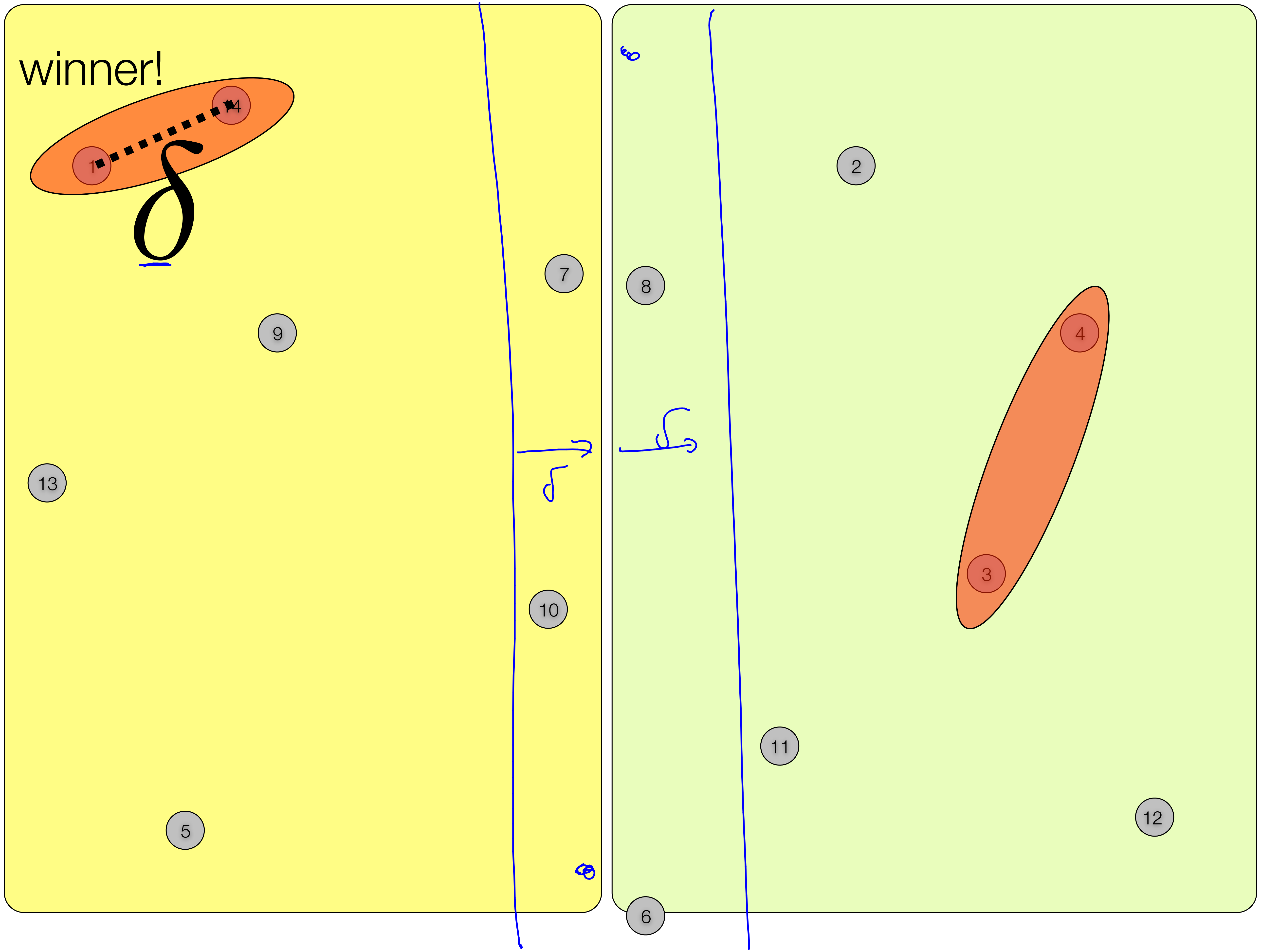
Divide & Conquer



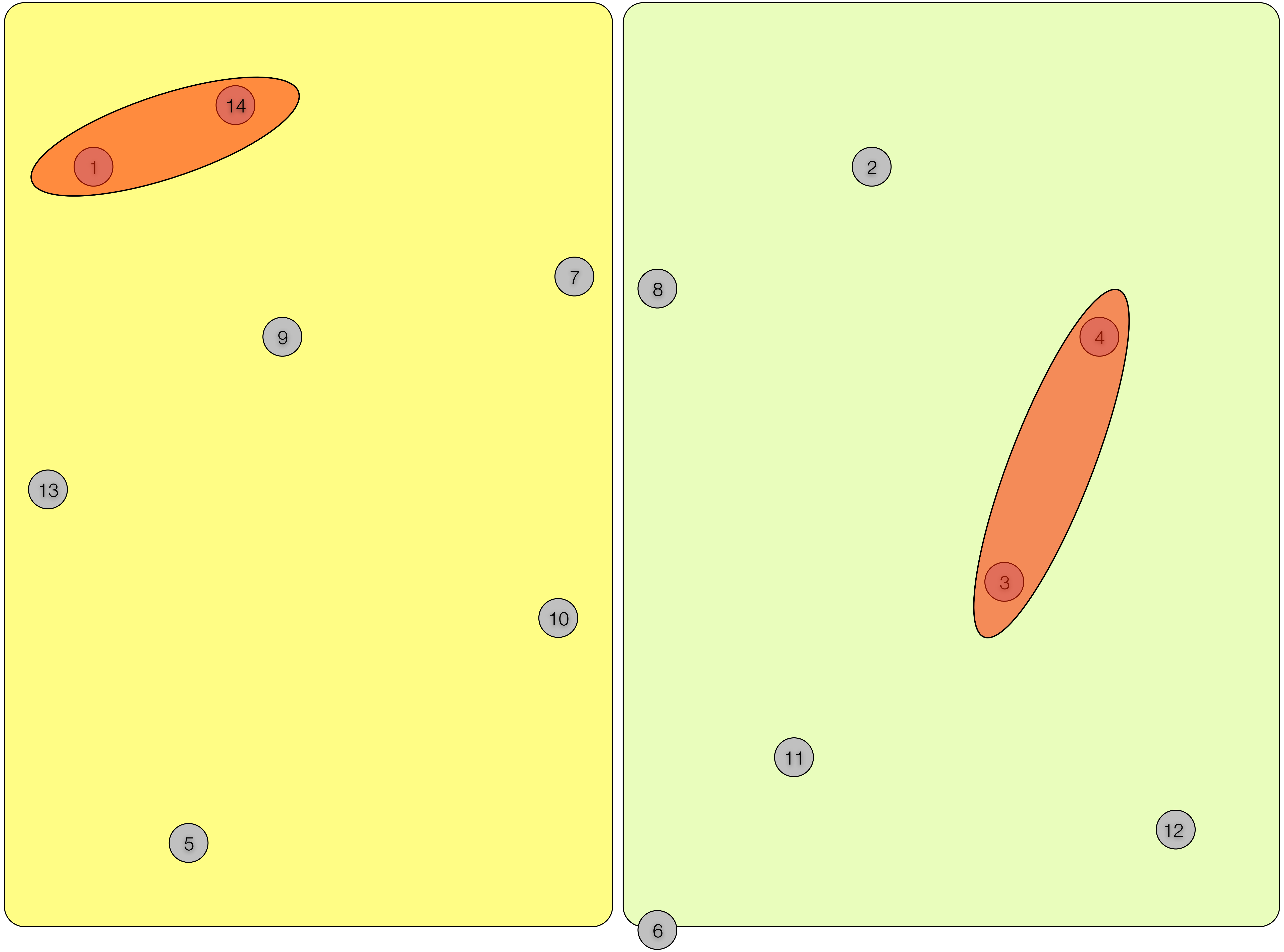
Divide & Conquer



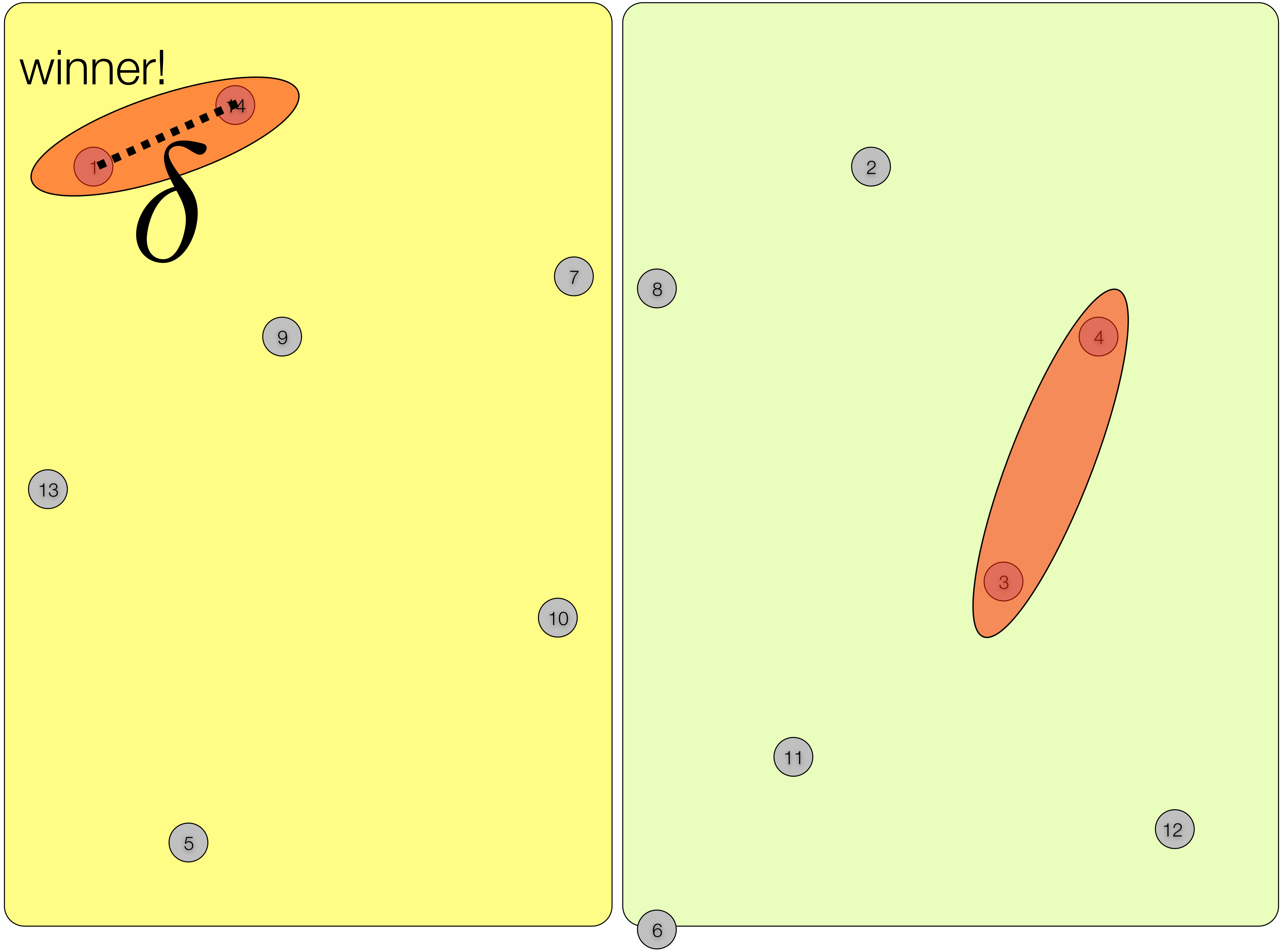
Divide & Conquer



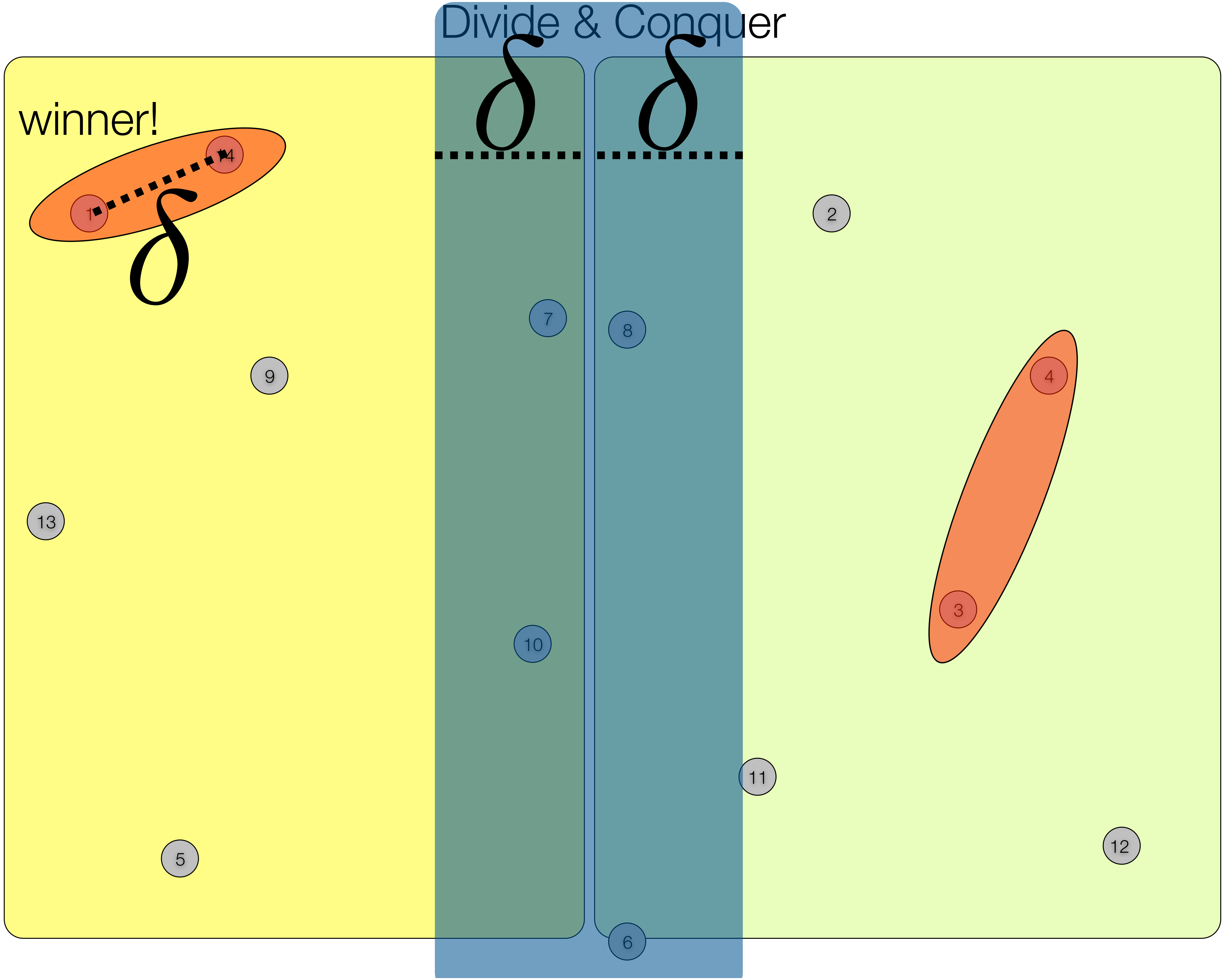
Divide & Conquer

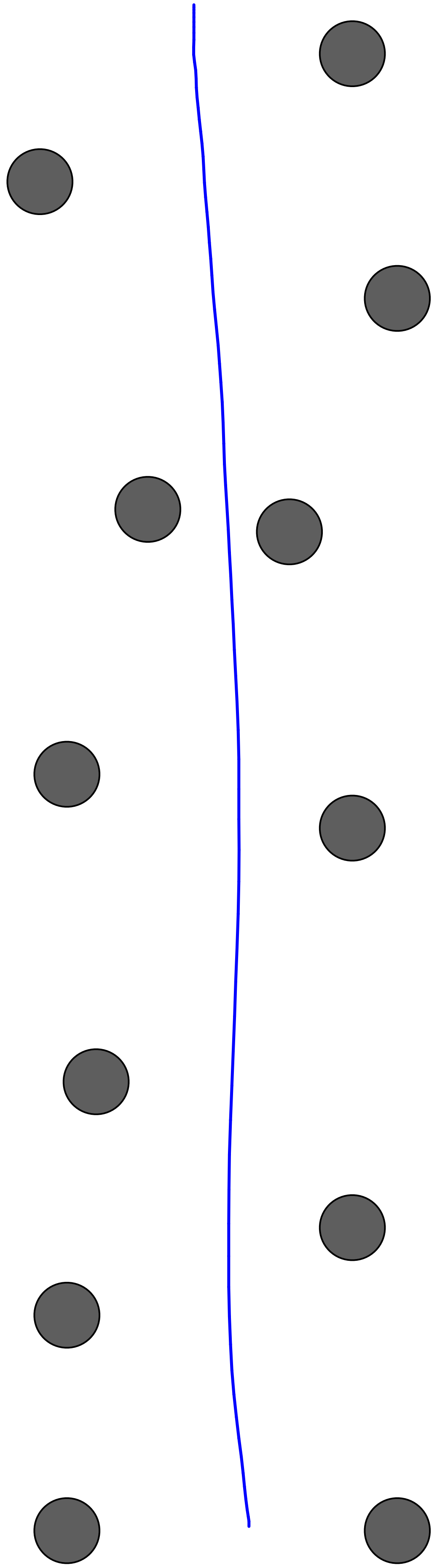


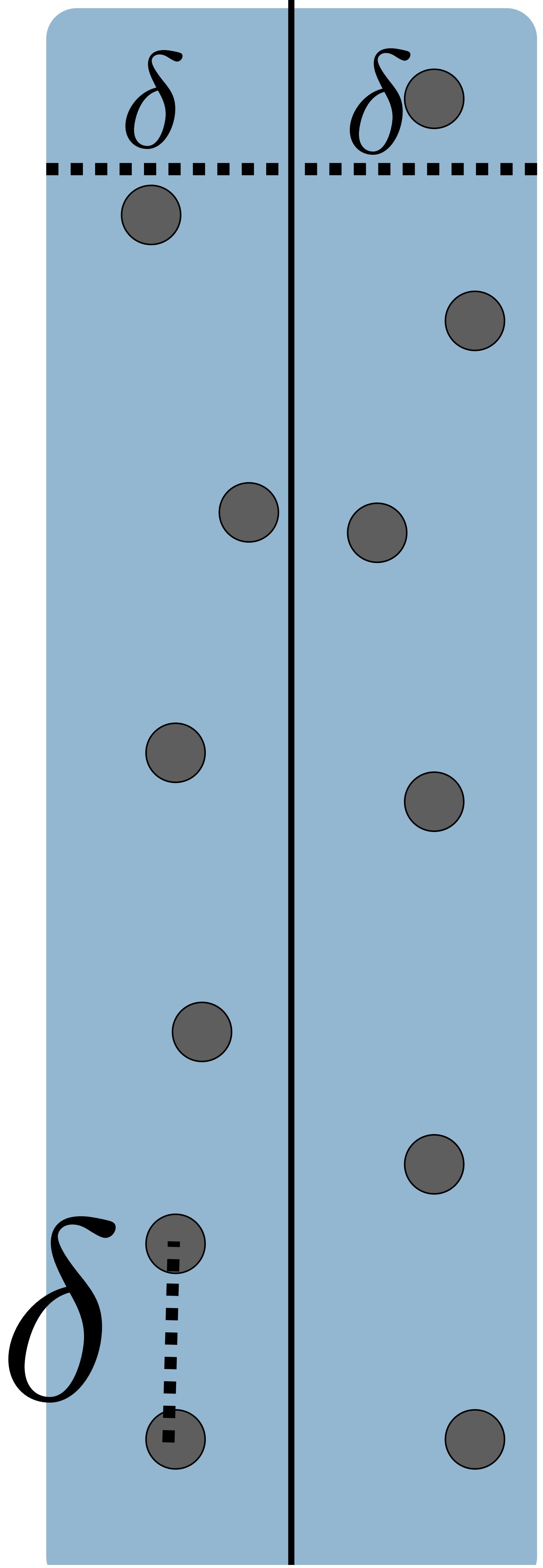
Divide & Conquer

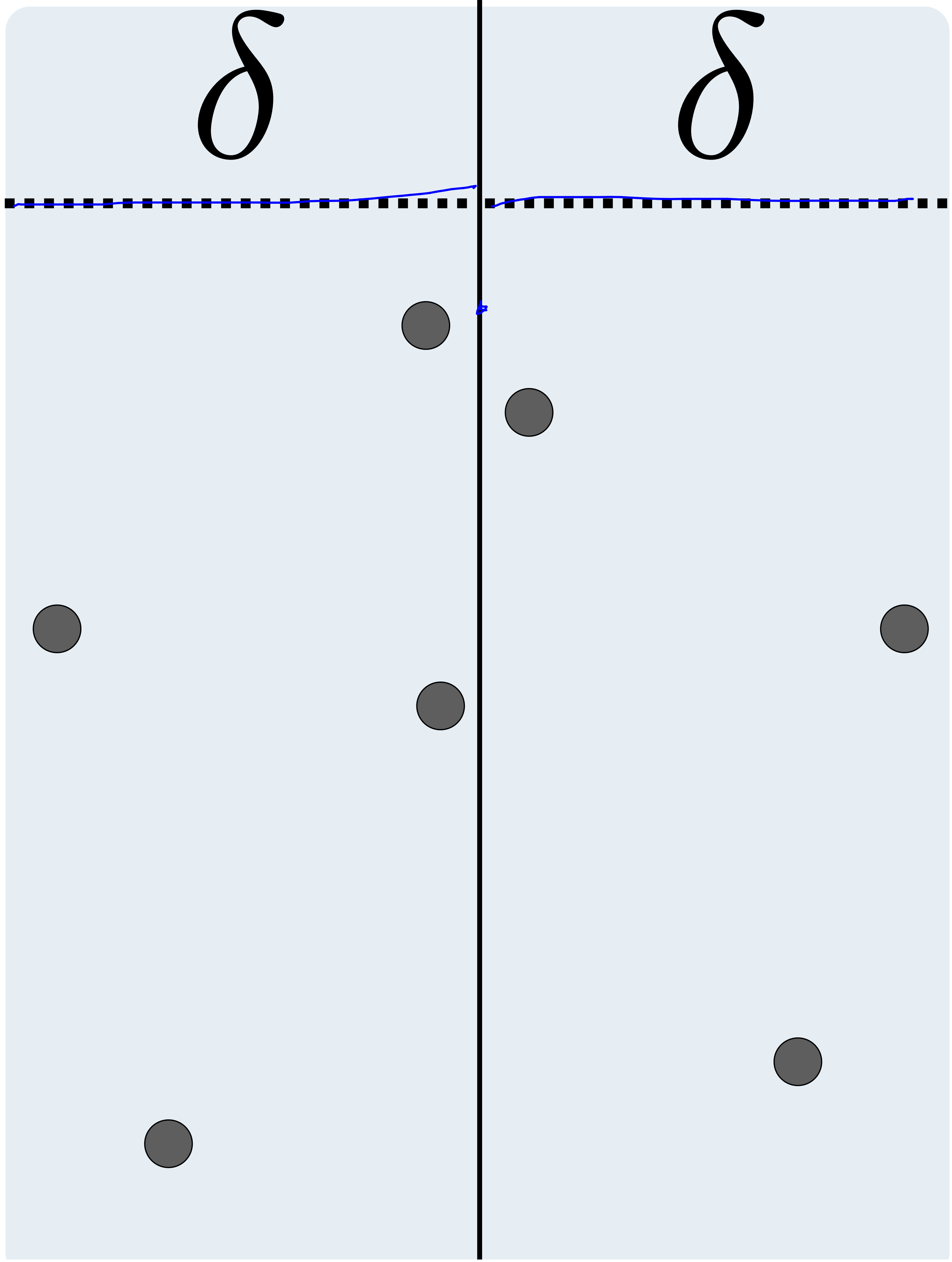


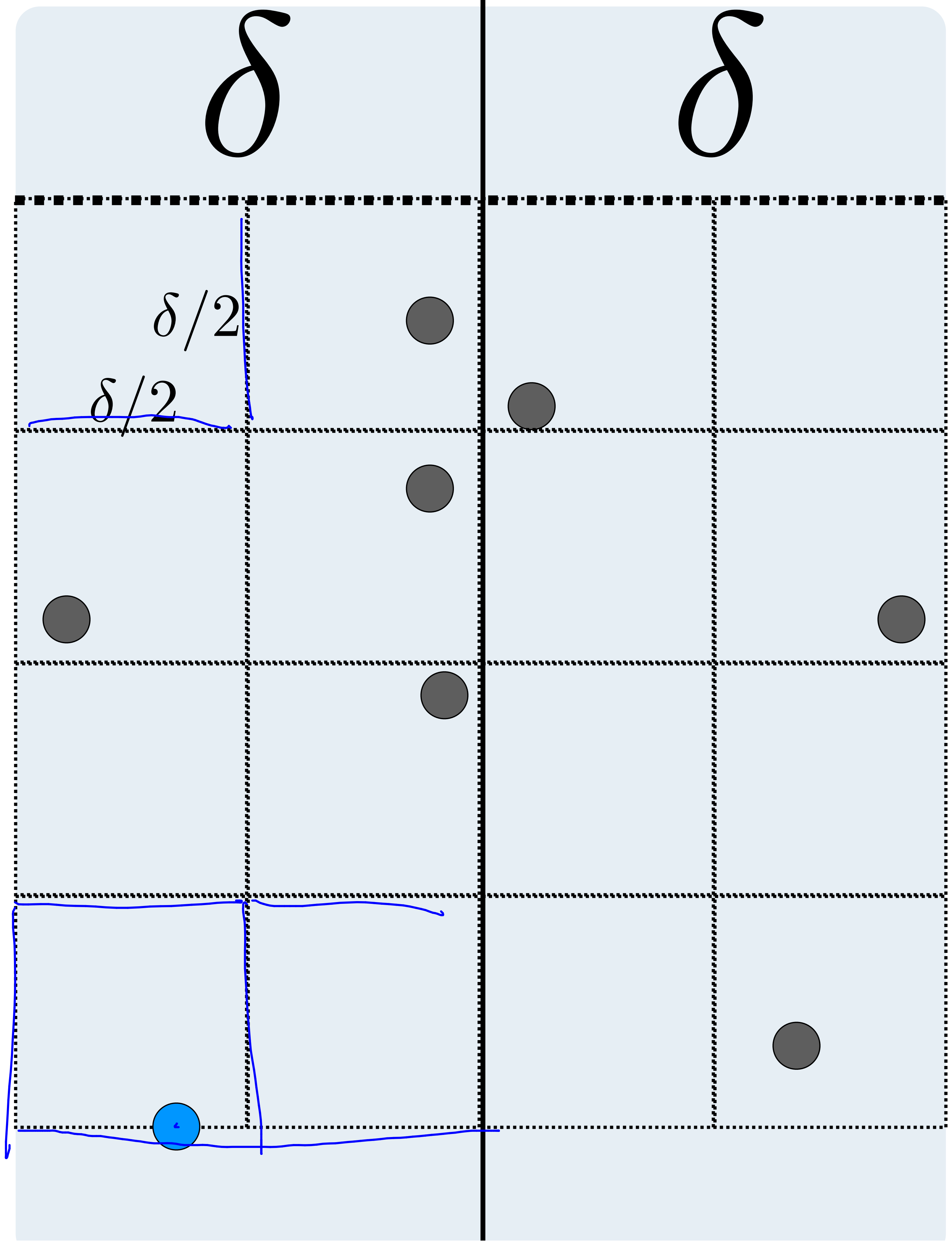
Divide & Conquer



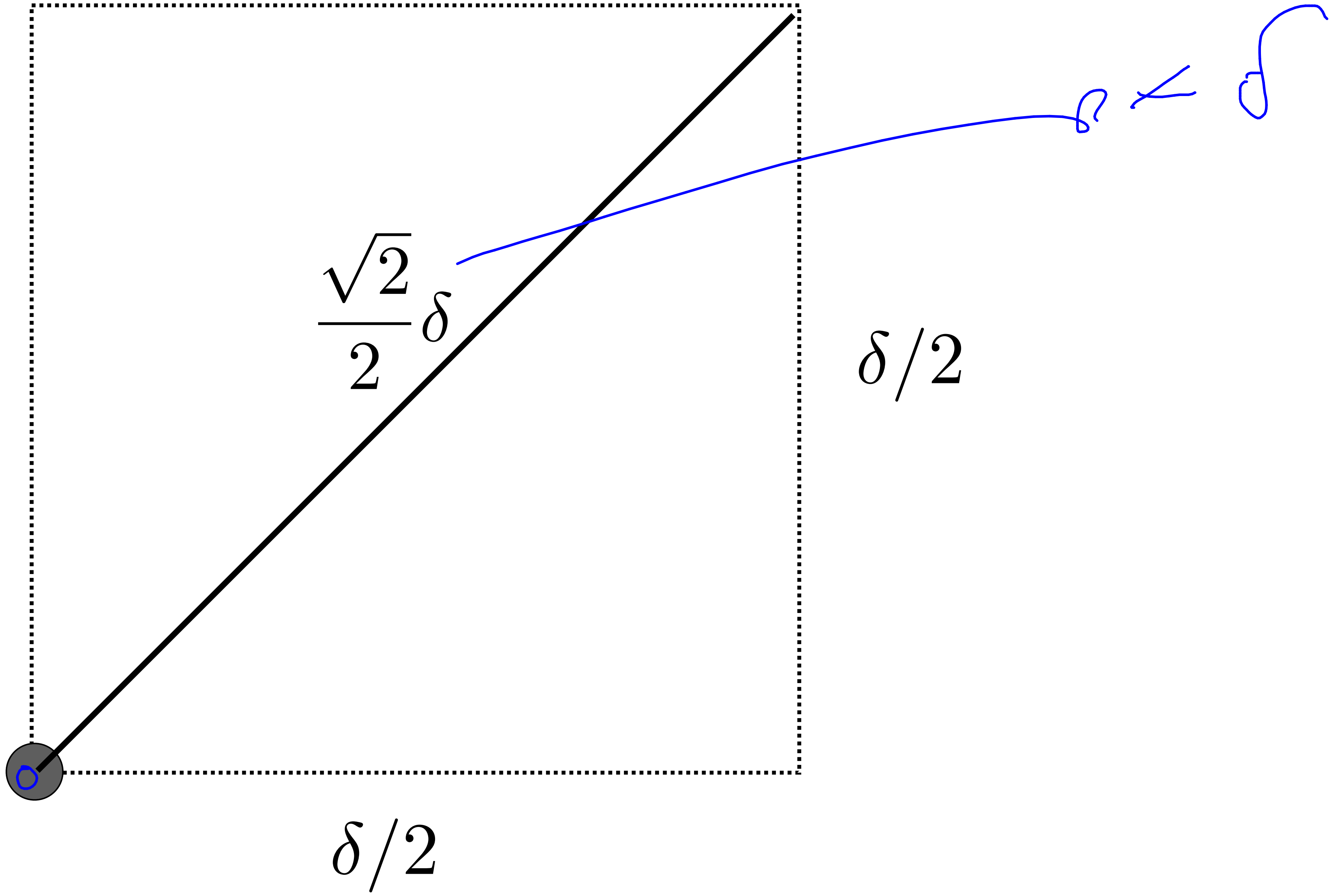


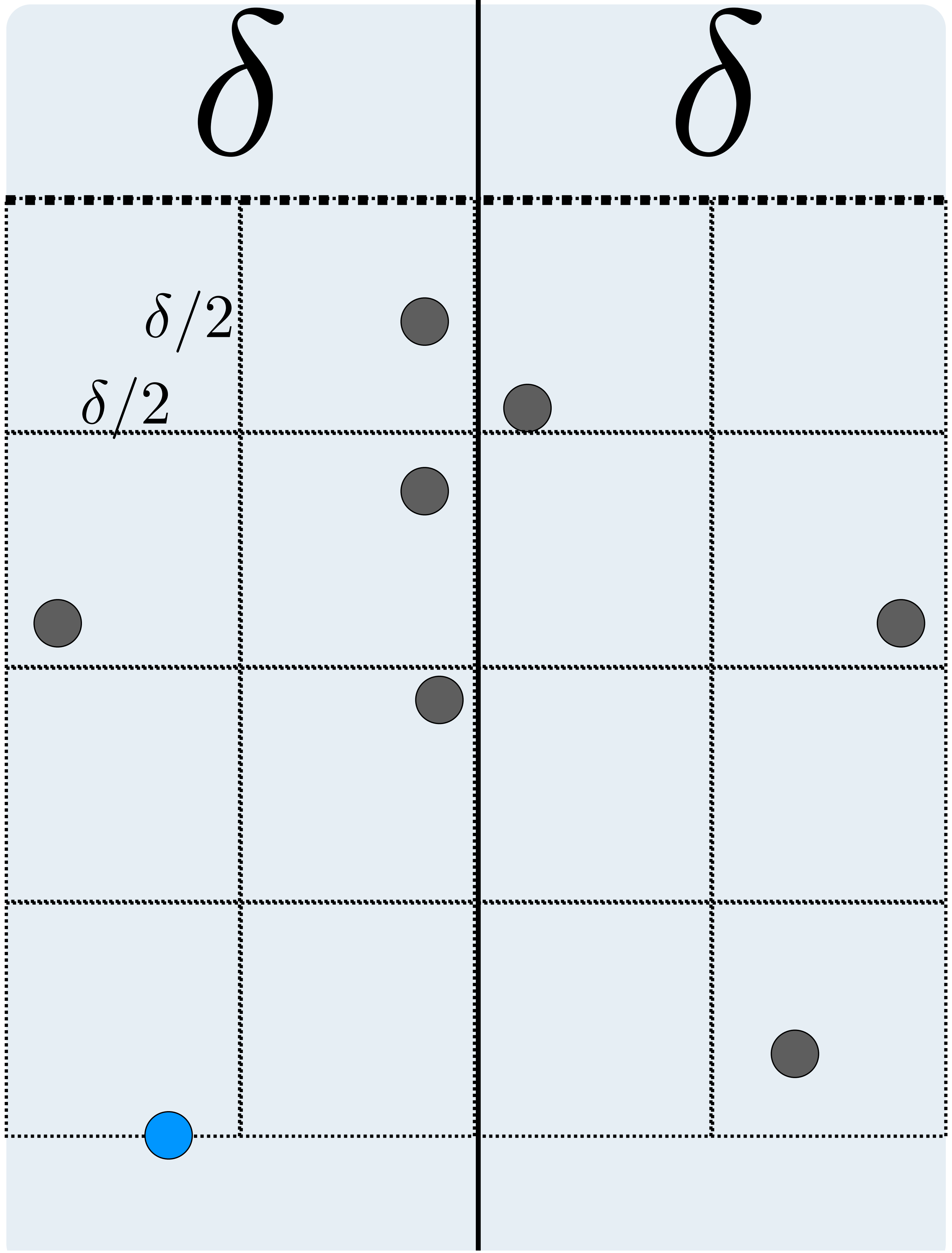

δ
δ
δ


δ
δ

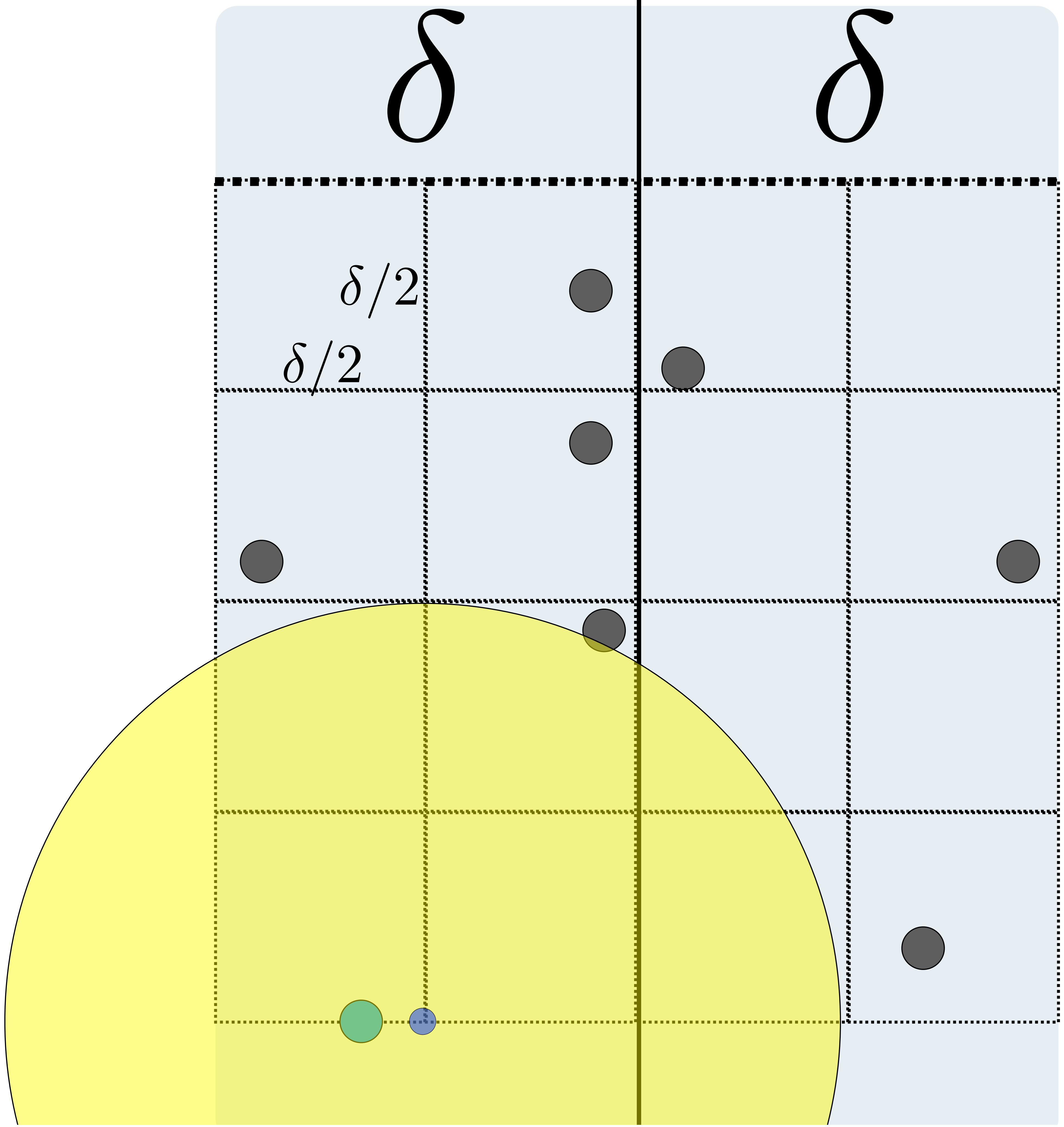


Imagine there is a grid of cubbies starting at the lowest Y point

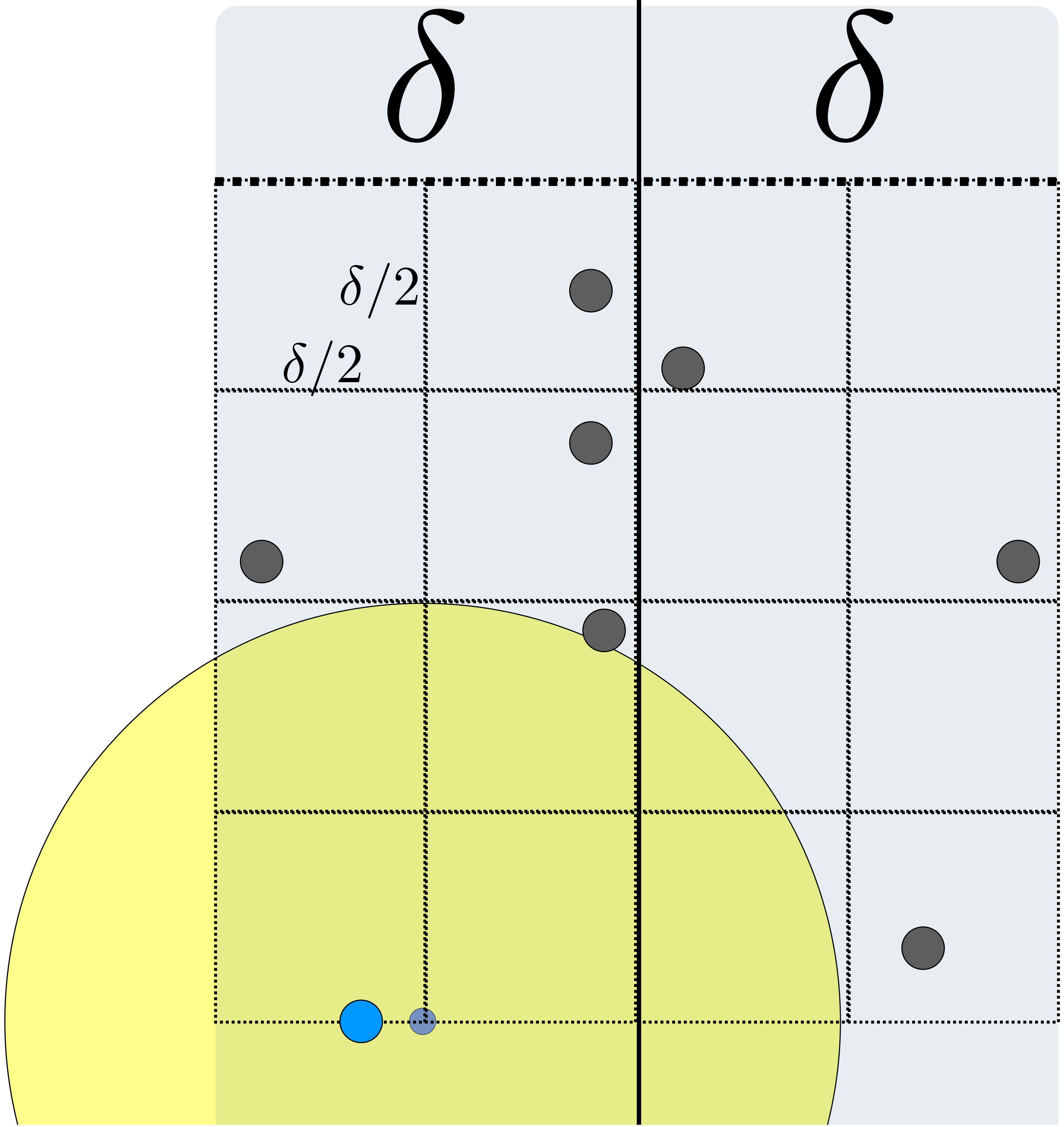

$\frac{\sqrt{2}}{2}\delta$
$< \delta$
$\delta/2$
$\delta/2$



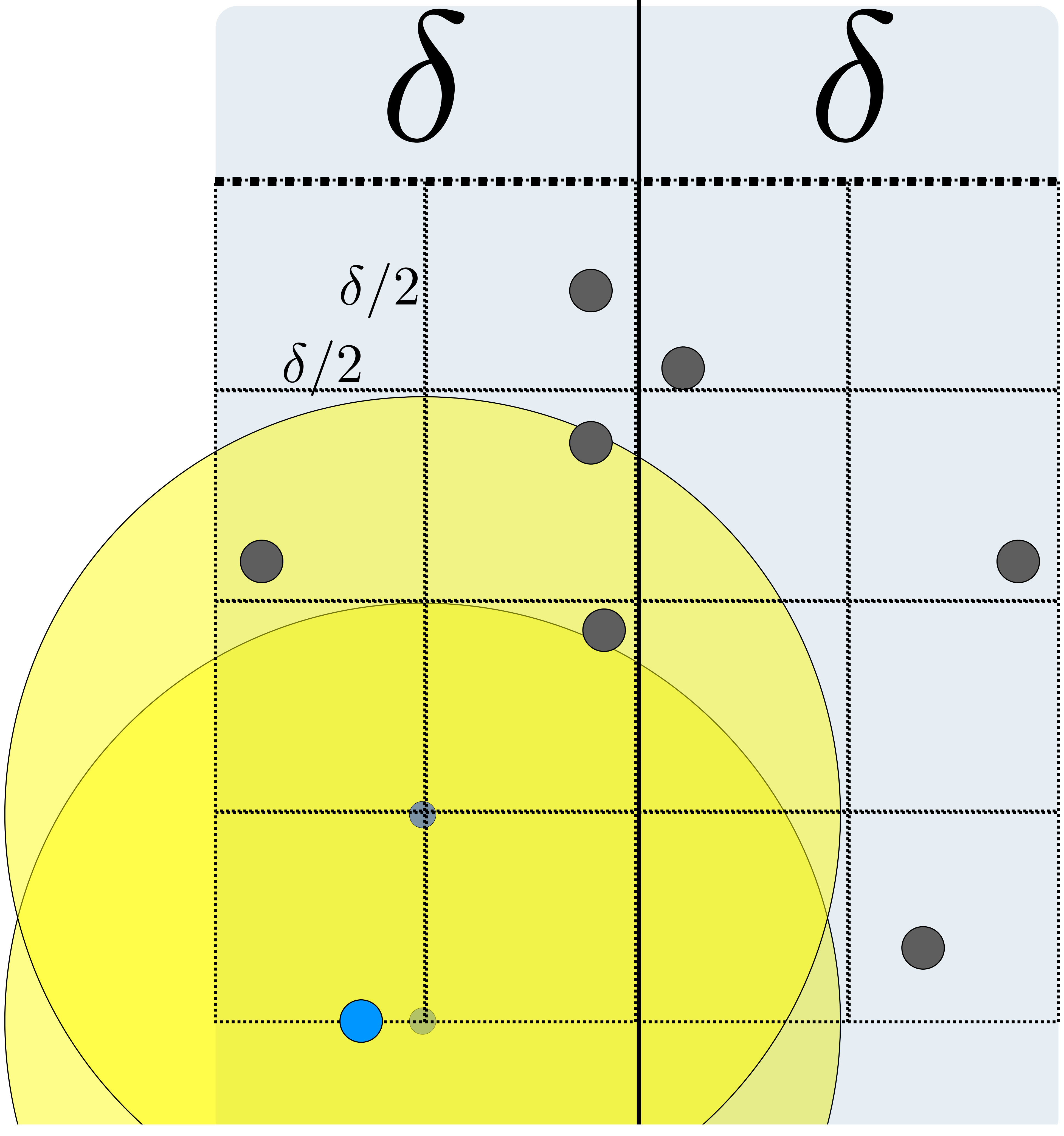
FACT: At most 1 point in each cubby



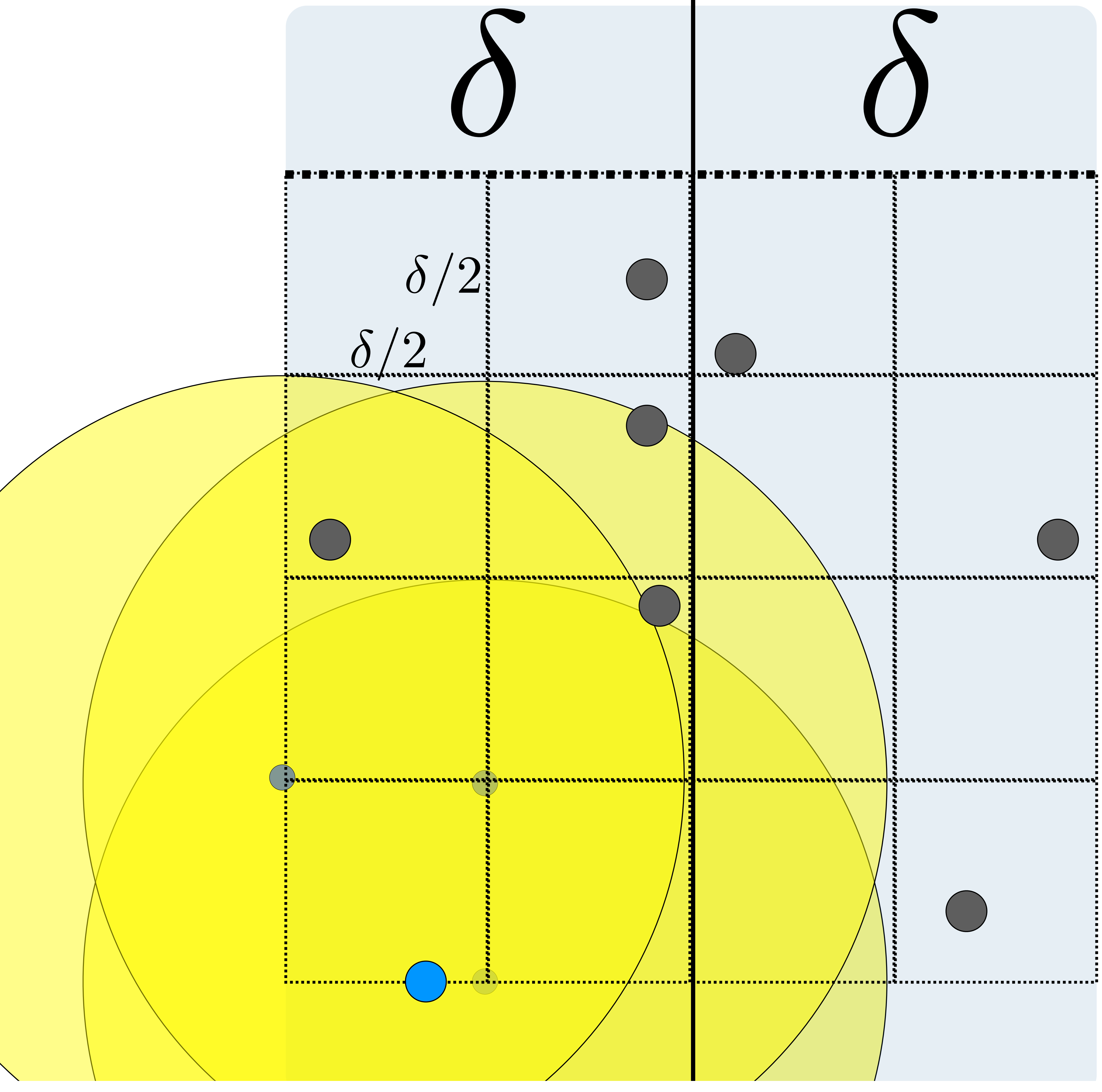
FACT: <=1 point per cubby



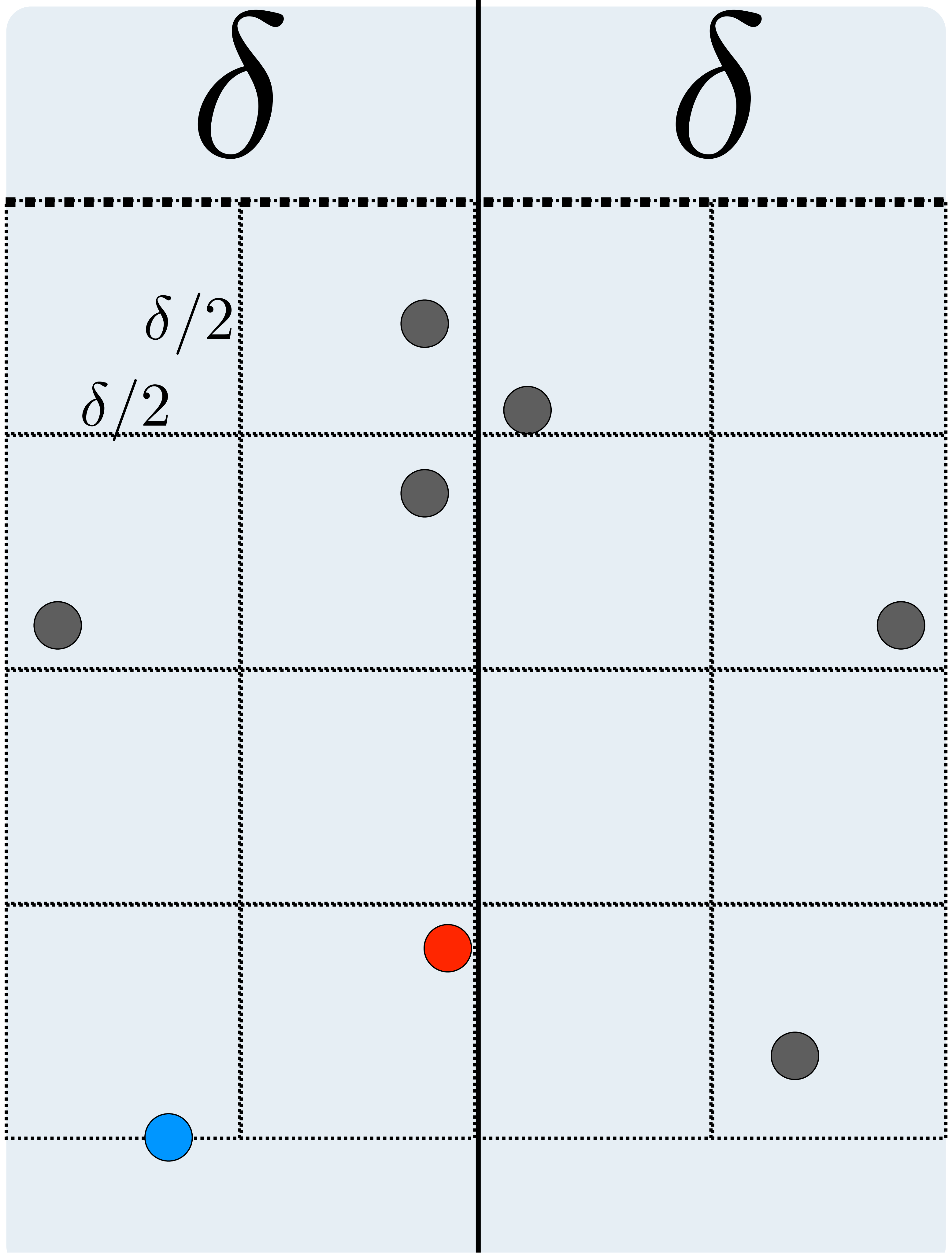
FACT: <=1 point per cubby

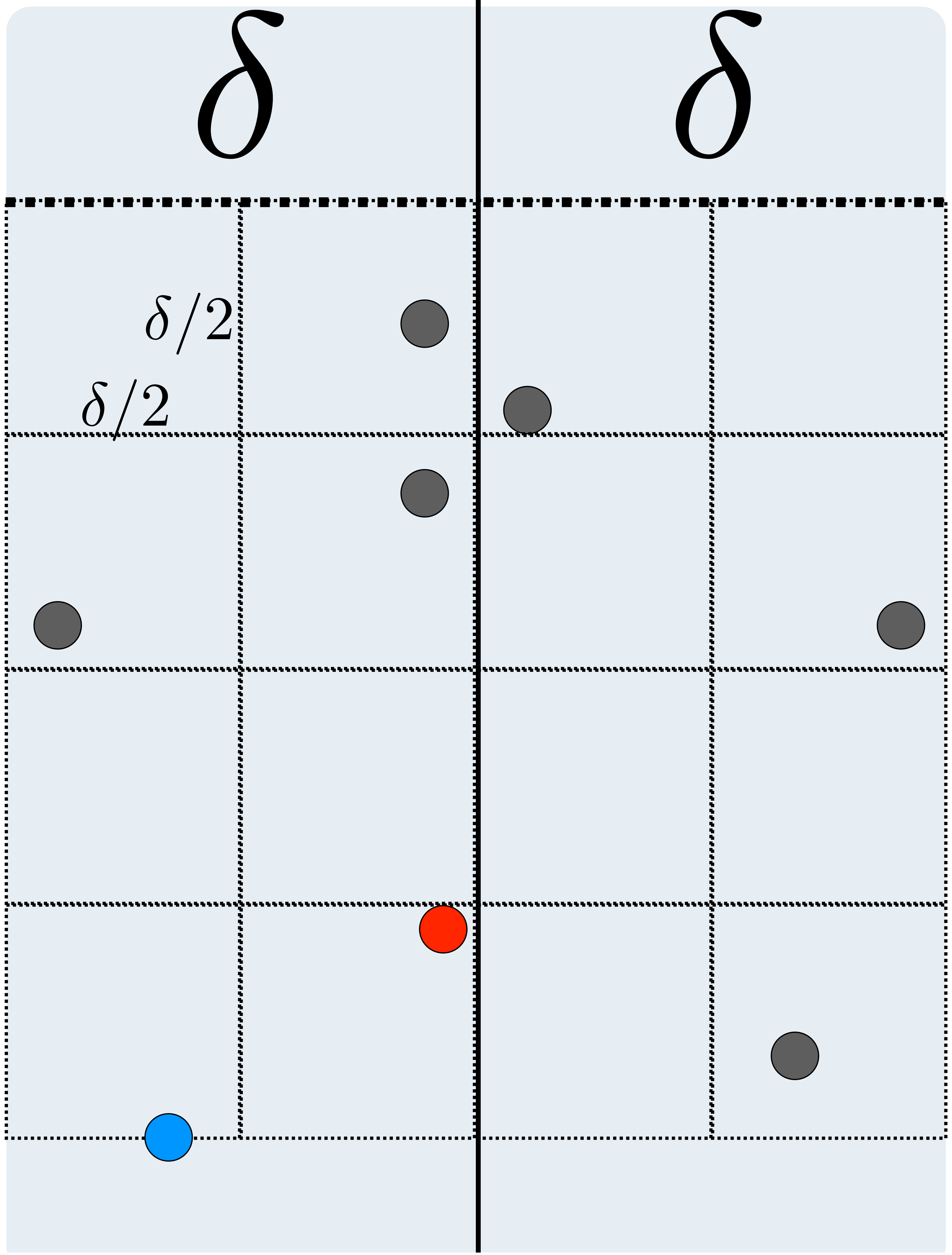


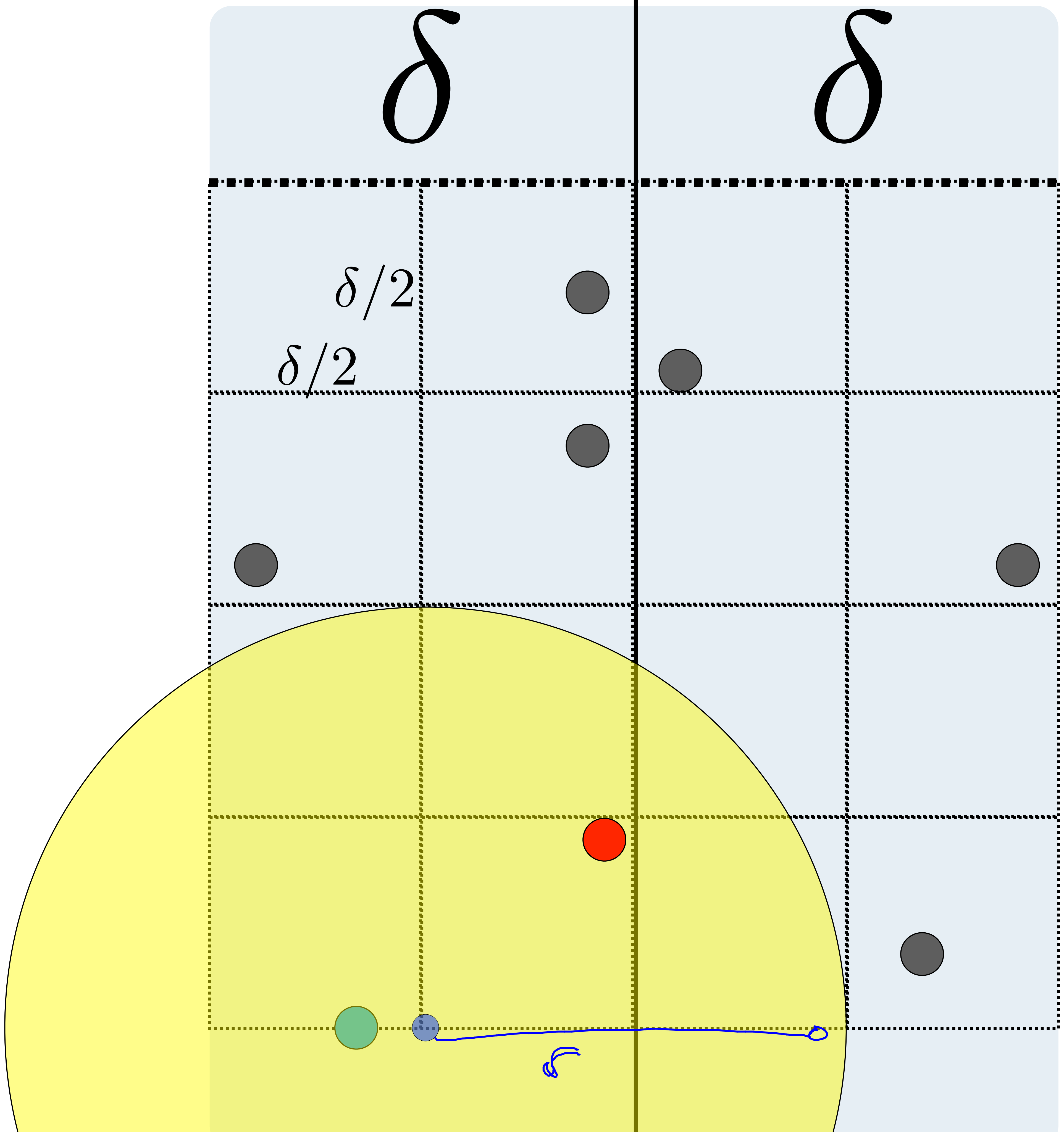
FACT: <=1 point per cubby

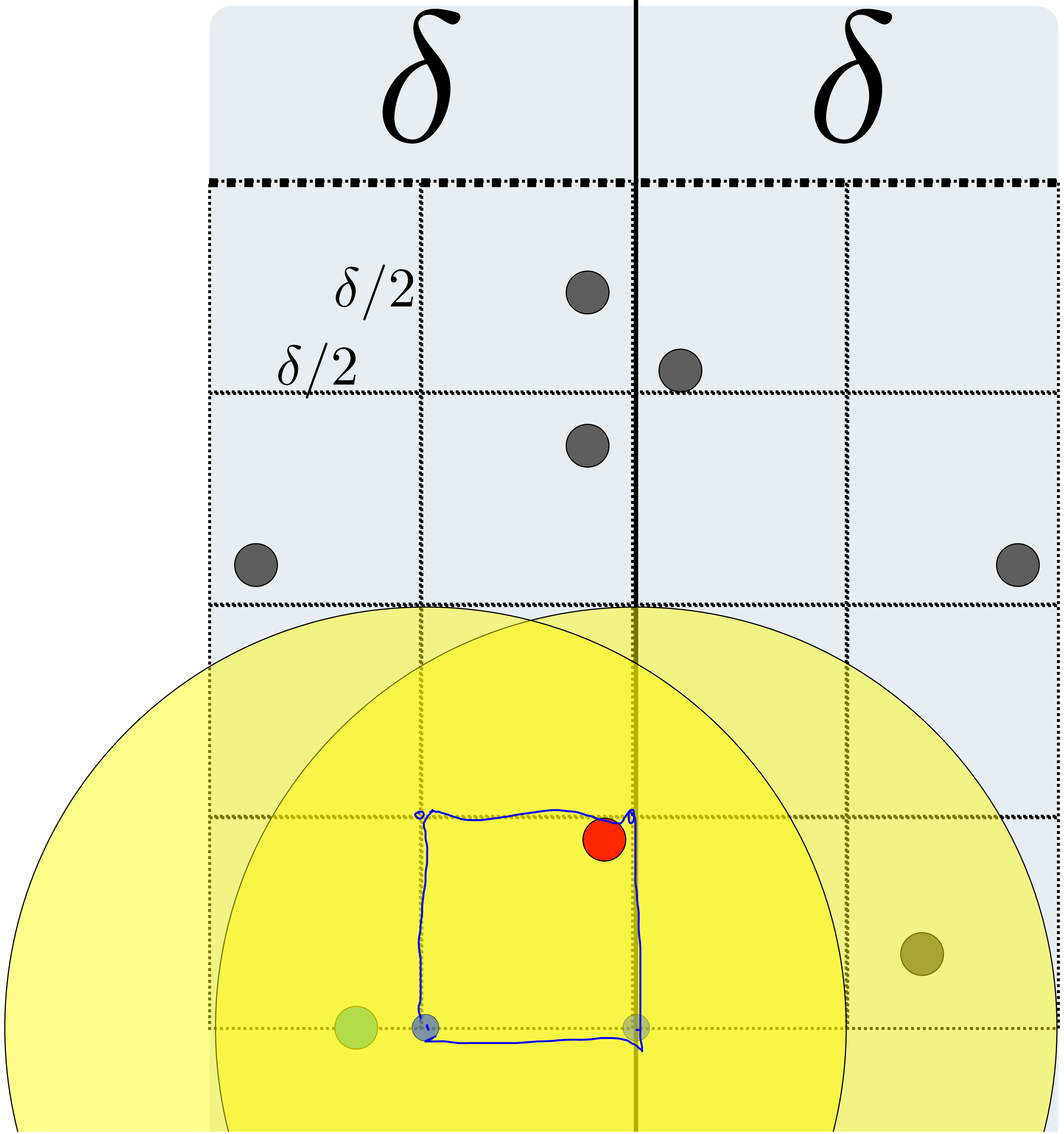


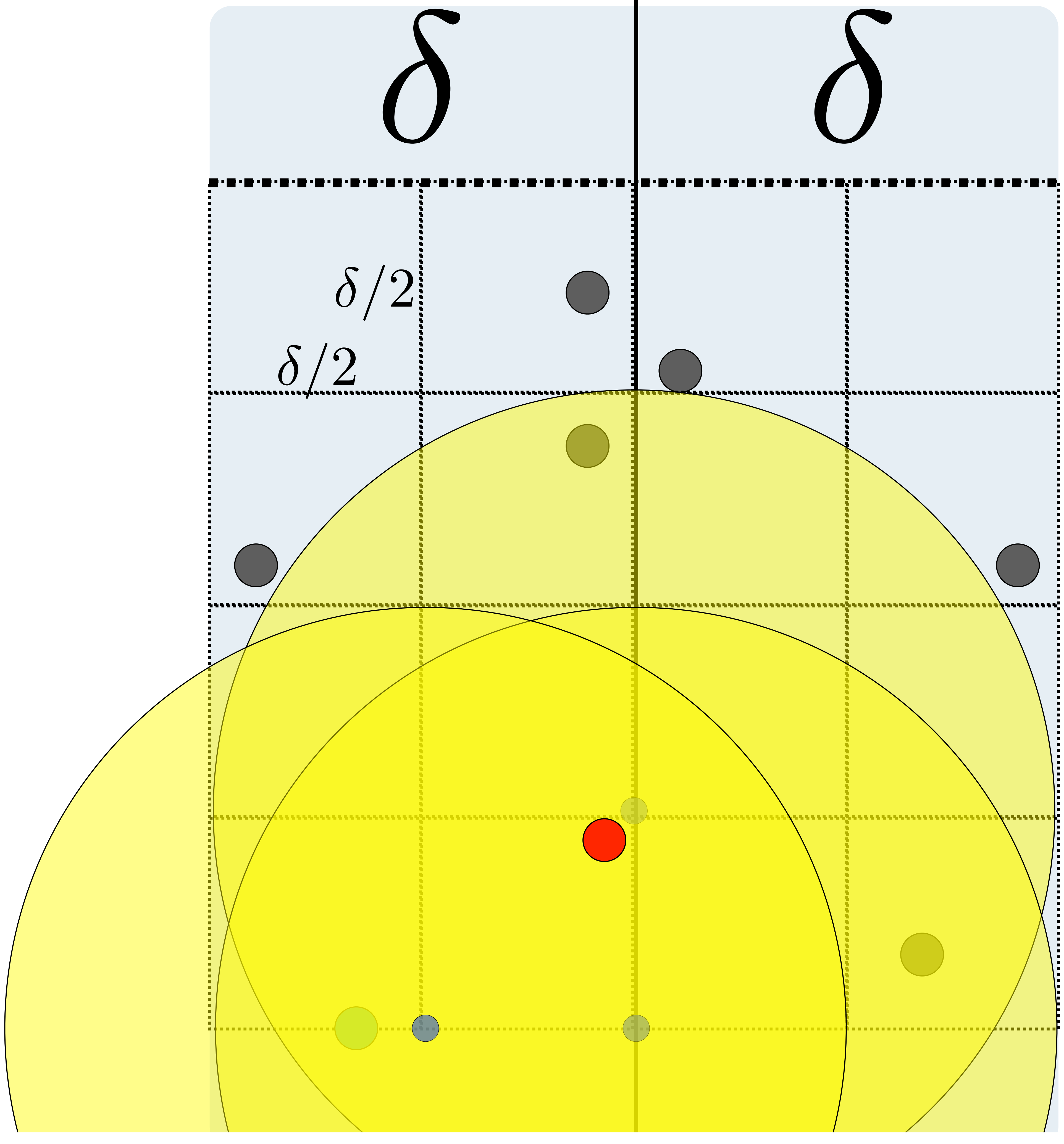
FACT: <=1 point per cubby

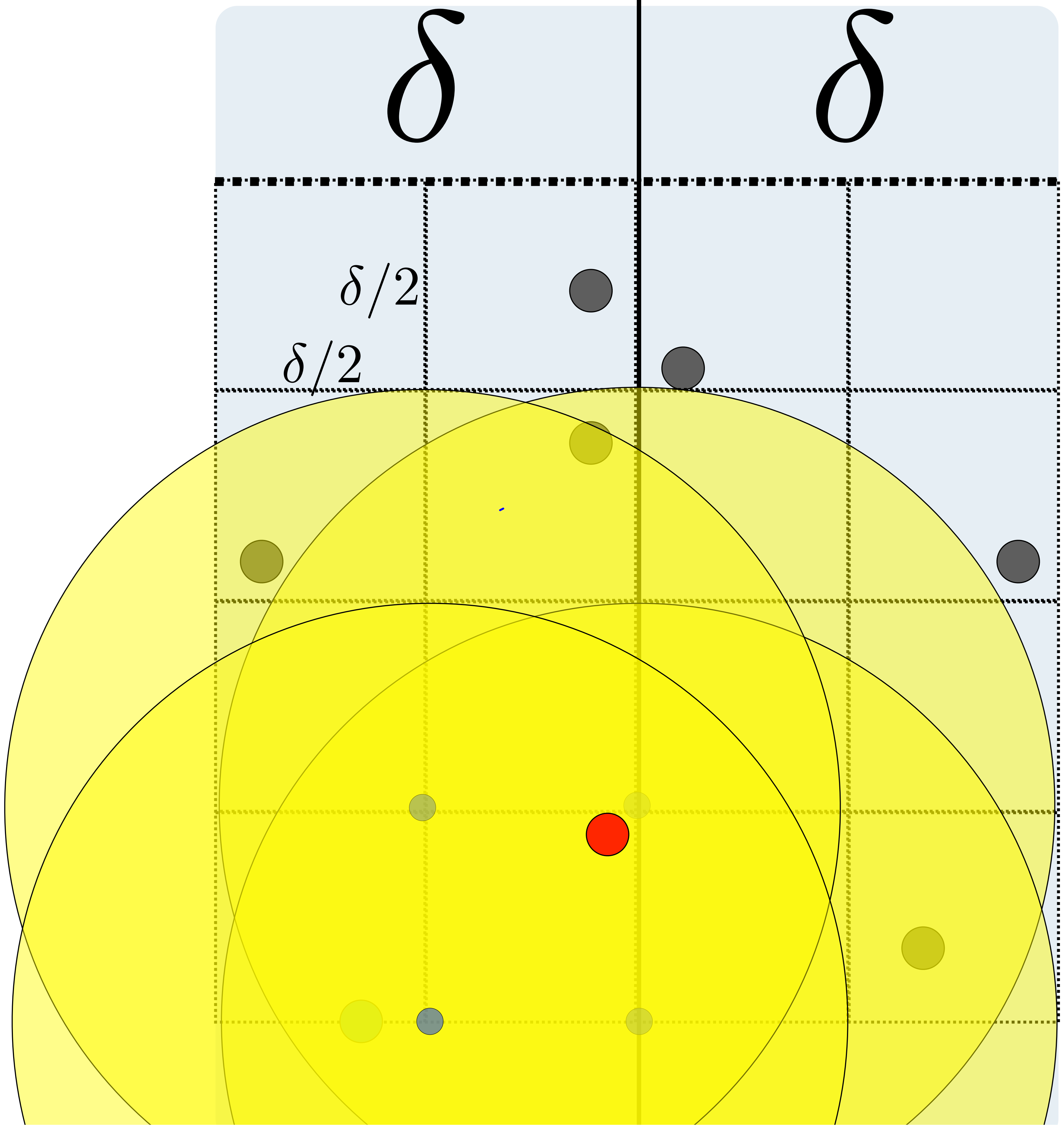

δ
δ
$\delta/2$
$\delta/2$

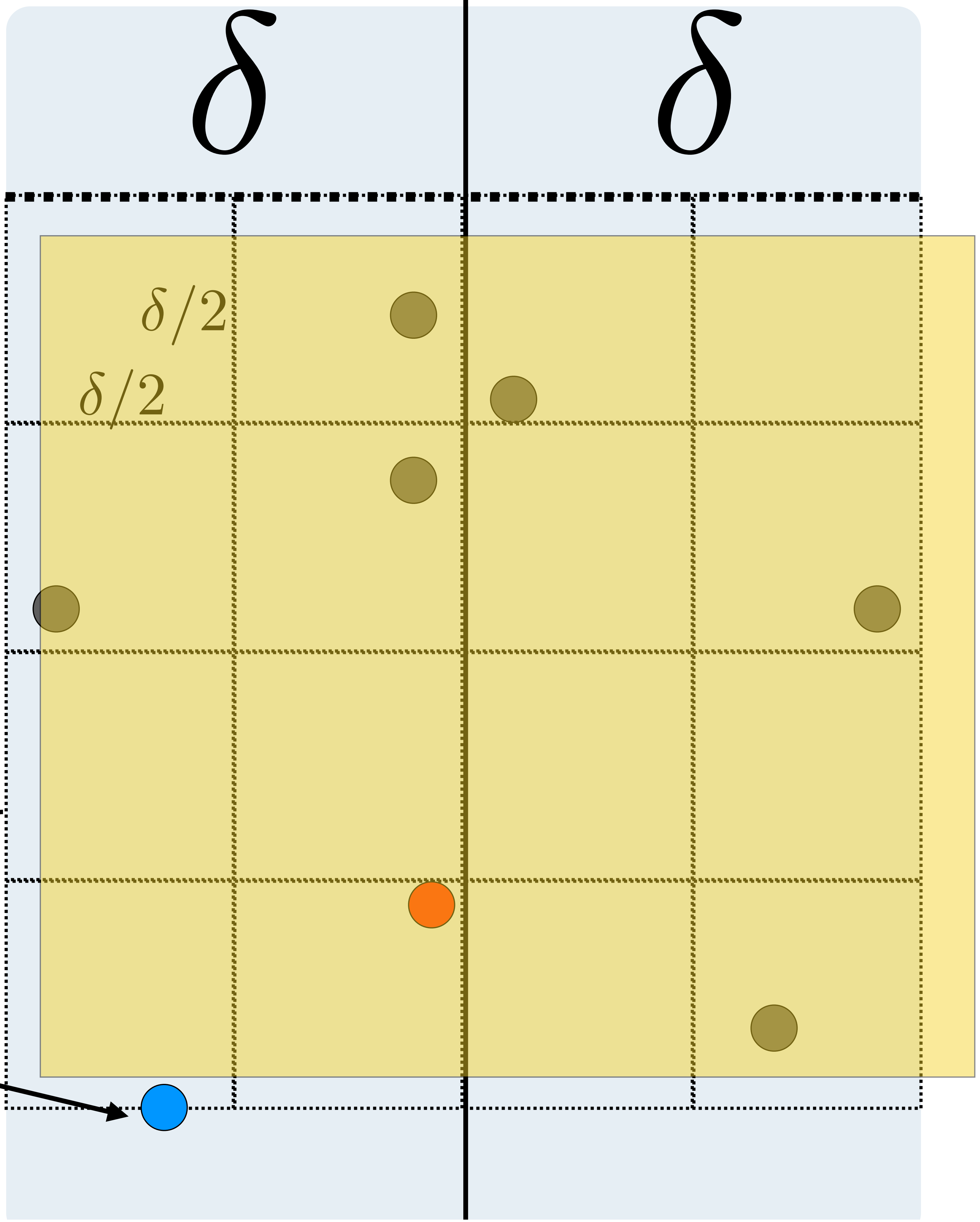

δ
δ
$\delta/2$
$\delta/2$


δ
δ
$\delta/2$
$\delta/2$


δ
δ
δ/2
δ/2


δ
δ
$\delta/2$
$\delta/2$


δ
δ
δ/2
δ/2

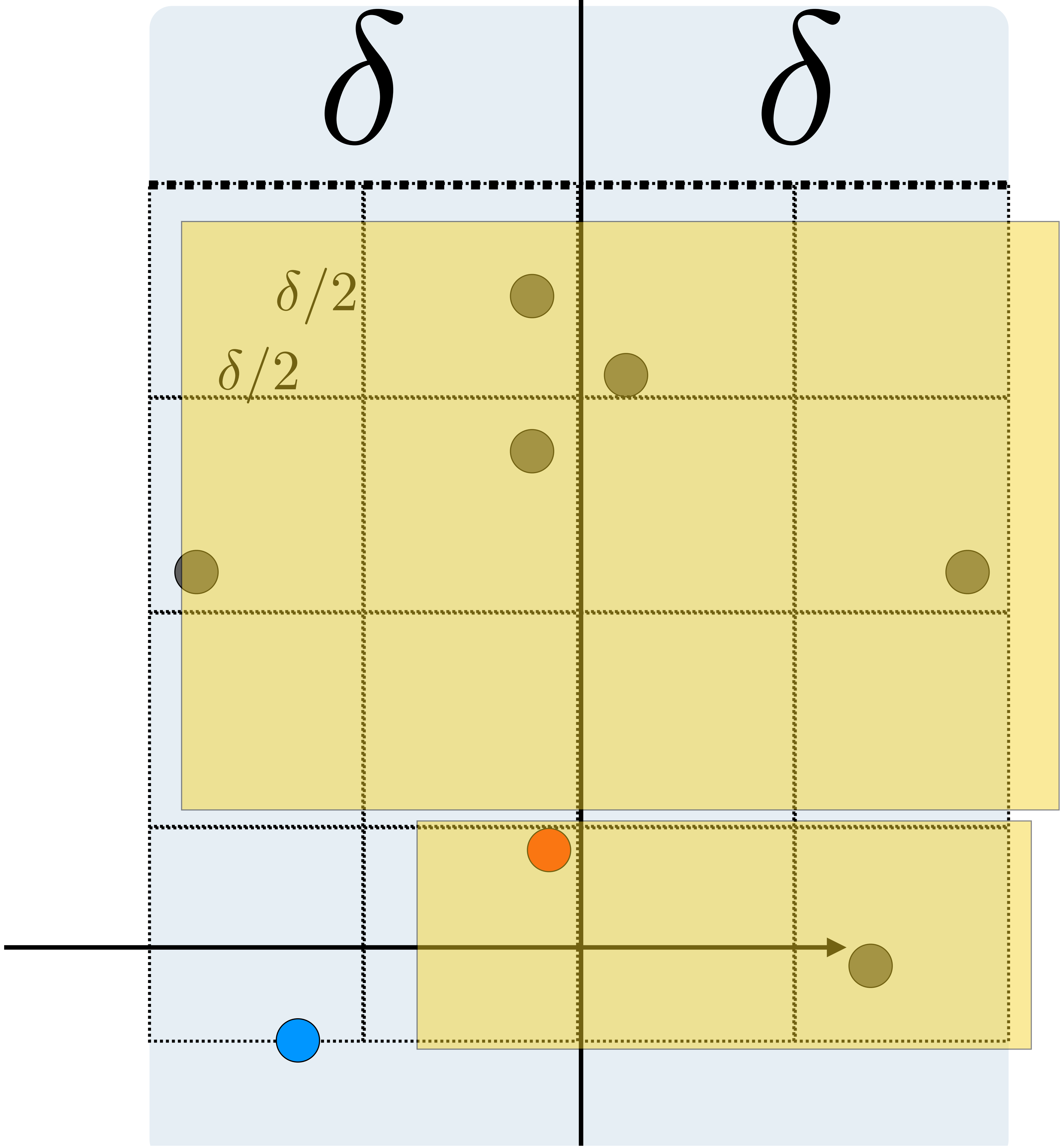


Visit its
by y-order

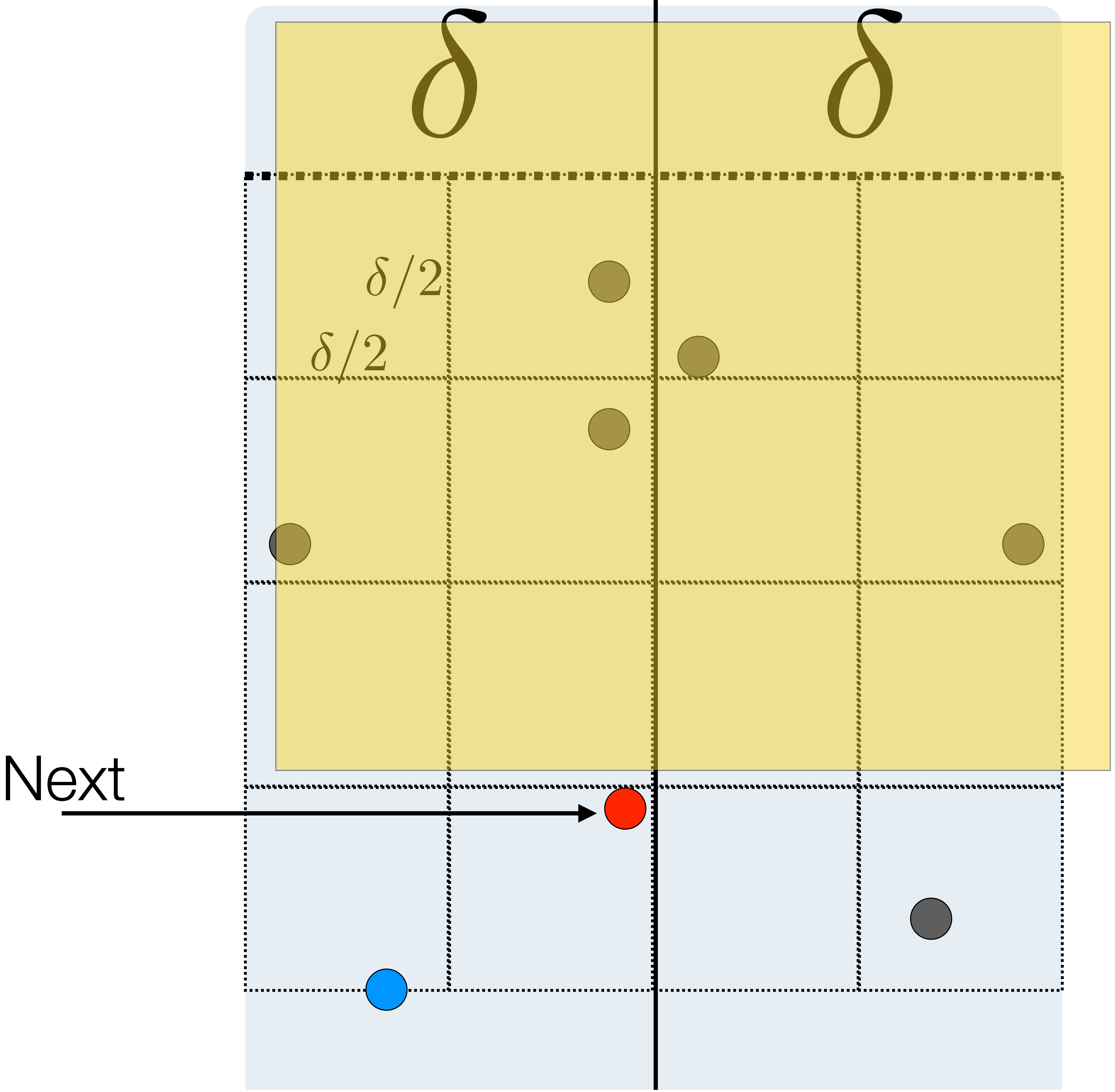
Start

Check the
next 15
boxes

Next



Check the next <15 boxes



Check the next <15 boxes

Closest(P)

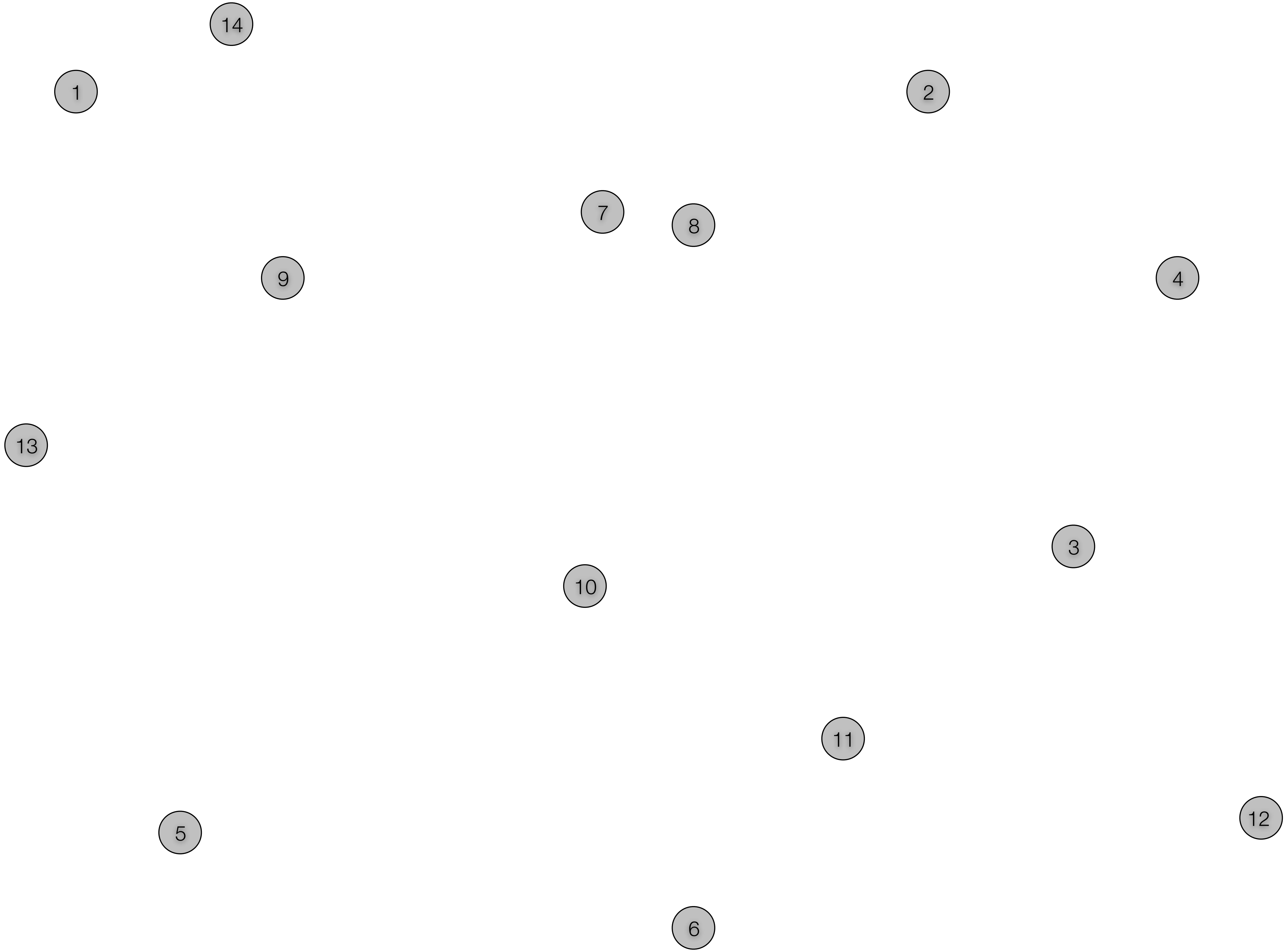
)

```
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)
```

```
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)
```

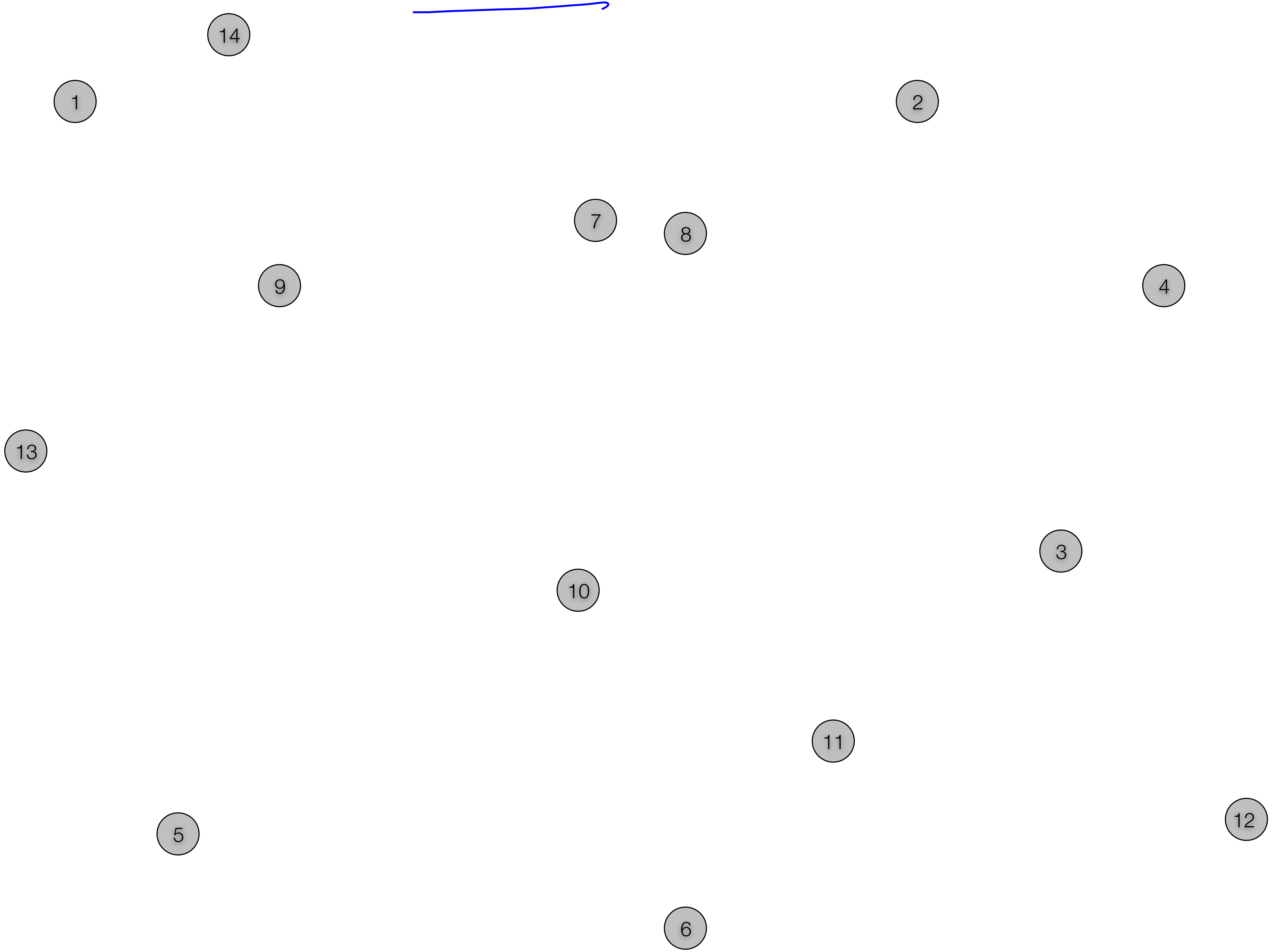
Can be reduced to 7!

Details: How to do step 1?



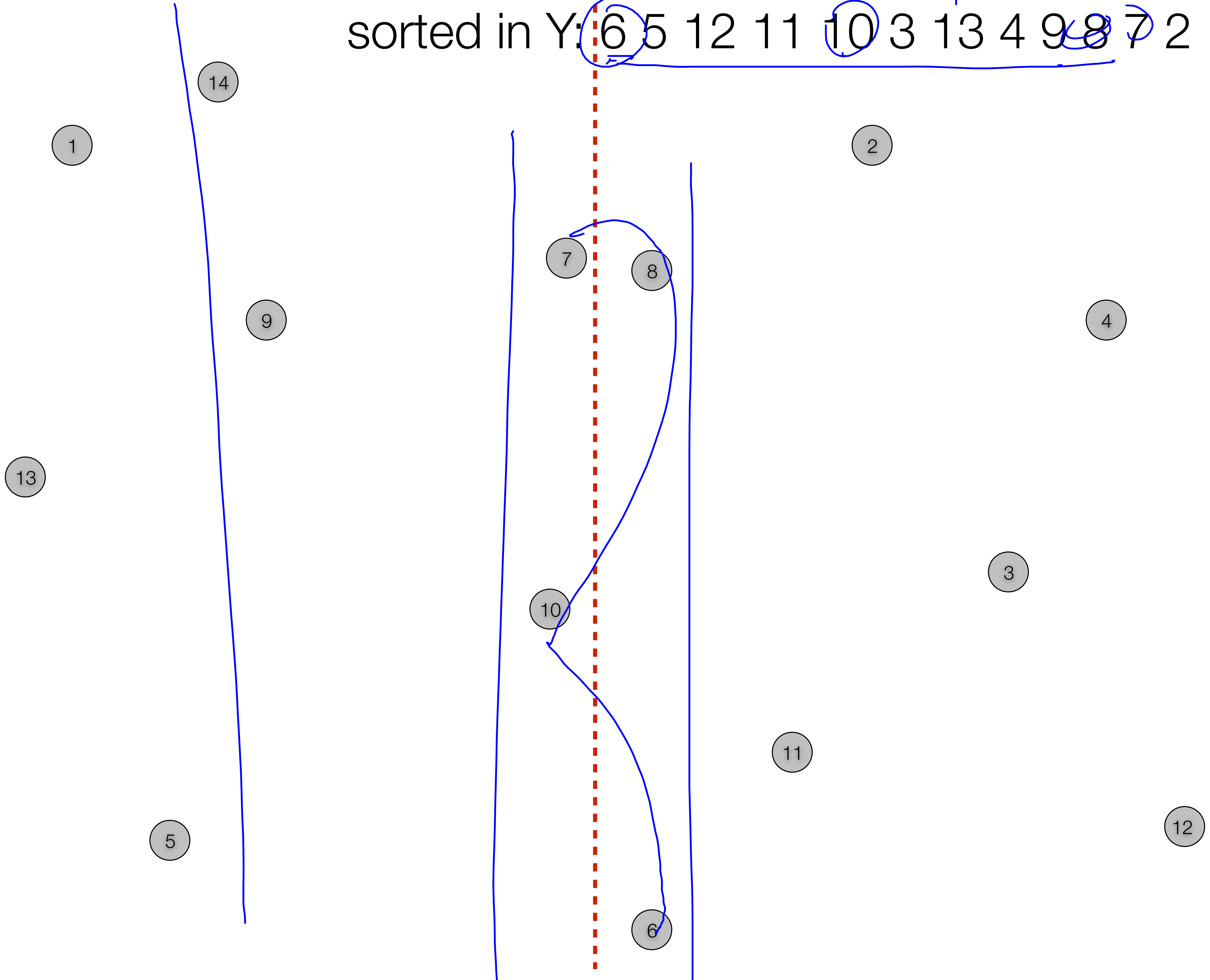
sorted in X: 13 1 5 14 9 10 7 9 8 11 2 3 4 12

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 $\Theta(n \log n)$
- Compute Sorted-in-Y list SY
- Closest(P,SX,SY) → $\Theta(n \log n)$

```
Closest(P,SX,SY)
    Let q be the middle-element of SX
    Divide P into Left, Right according to q
    delta,r,j = MIN(Closest(Left, LX, LY)  Closest(Right, RX, RY))   → $2T(\frac{n}{2})$

    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
                                                         $\Theta(n)$
    Return (delta,r,j)
```

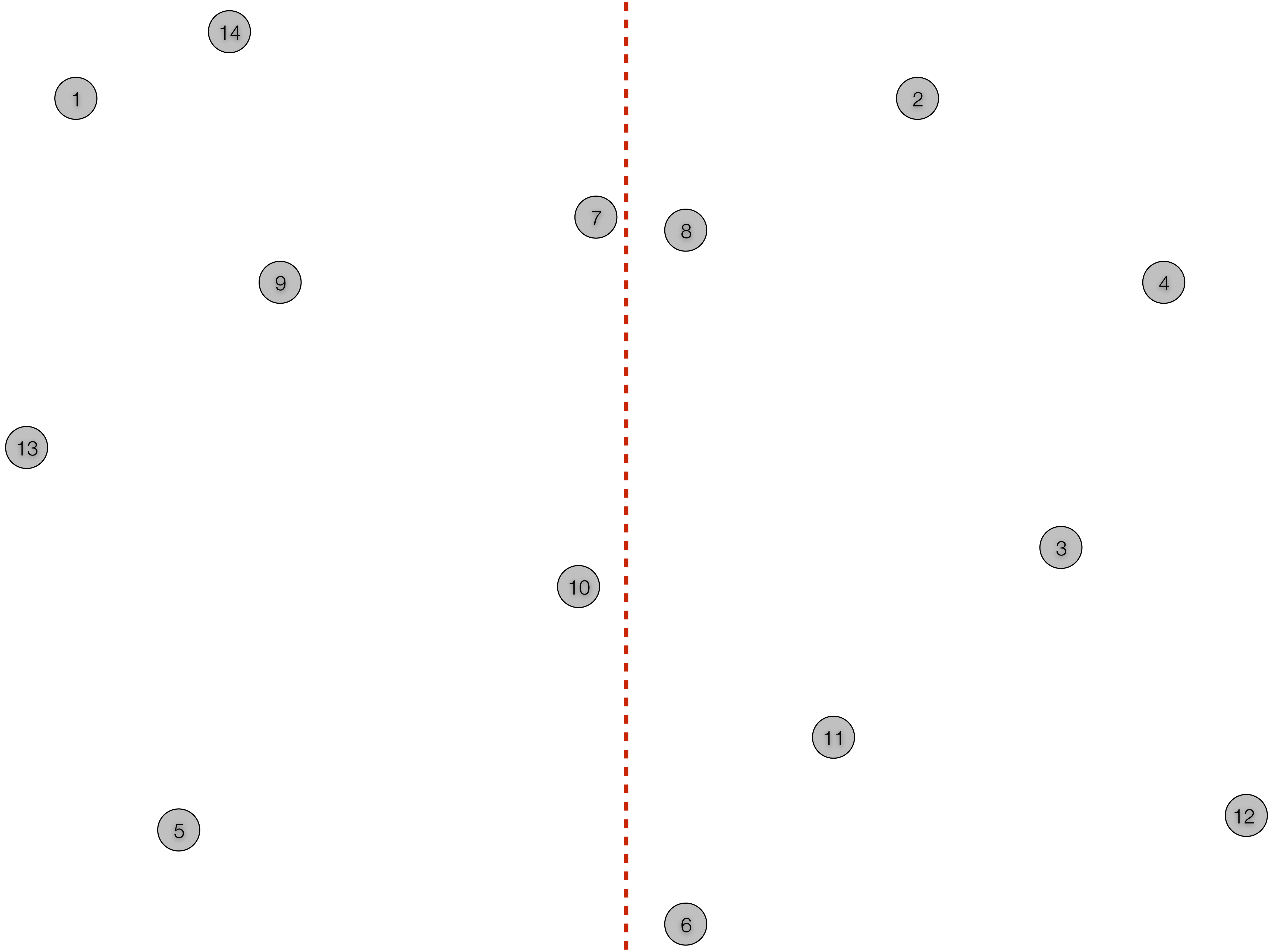
$$\underline{T(n) = 2T(\tfrac{n}{2}) + \Theta(n) = \Theta(n \log n)}$$

```
Closest(P,SX,SY)
    Let q be the middle-element of SX
    Divide P into Left, Right according to q
    delta,r,j = MIN(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

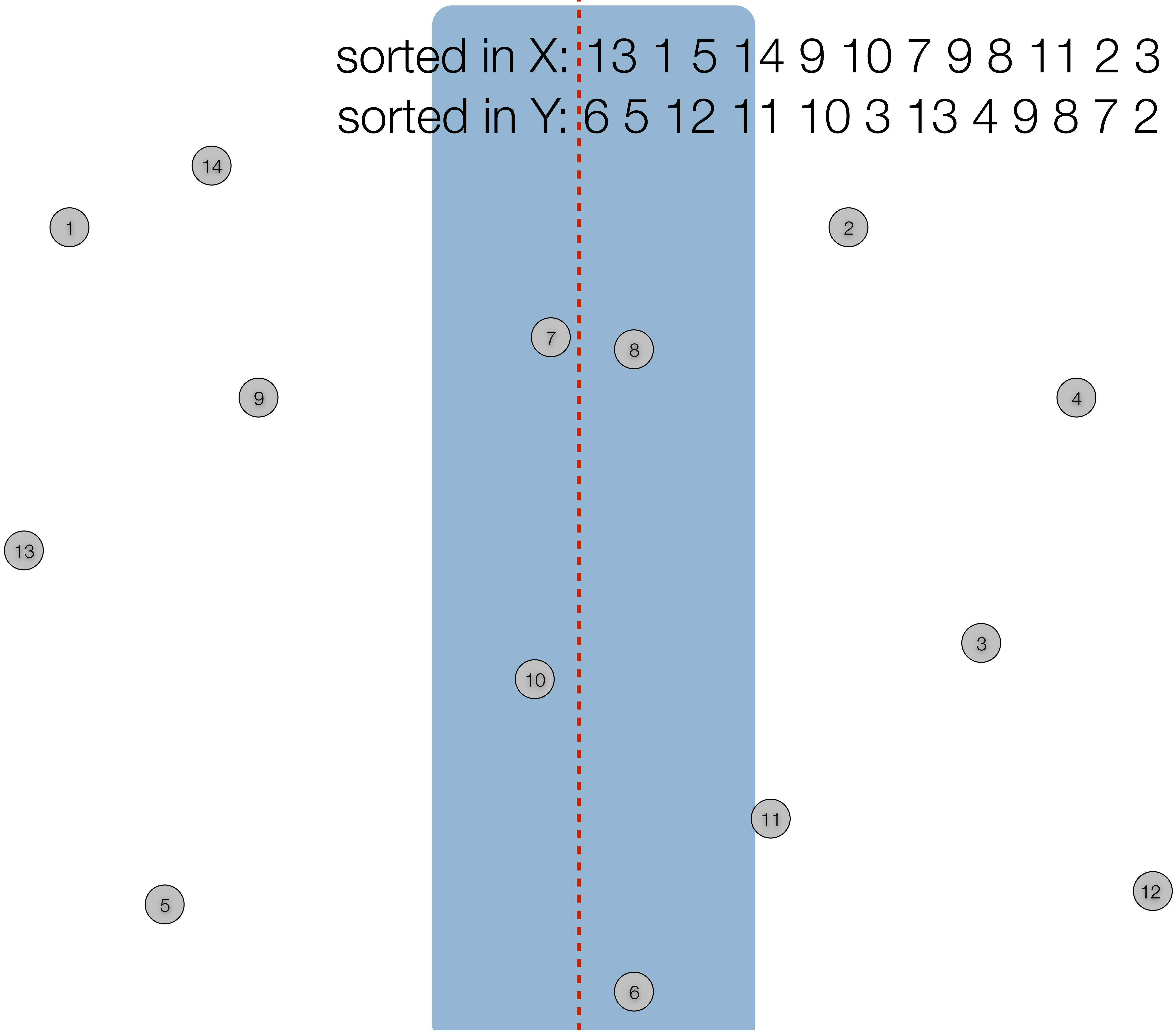
sorted in Y: 6 5 12 11 10 3 13 4 9 8 7 2 1 14



```
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!



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

```
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);
}
```

```
// 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;
}
```