Divide and Conquer.

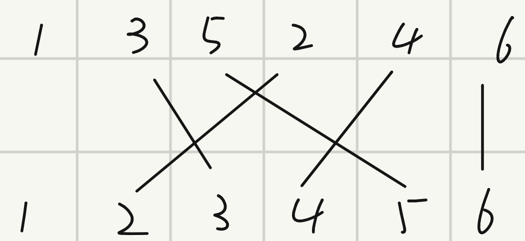
1. divide
2. conquer.
3. combine

Counting Inversion

1	3	5	2	4	6

A[1] A[2] A[6]

(i,j) is an inversion if $i<j$ and $A[i] > A[j]$.



each "x" implies an inversion

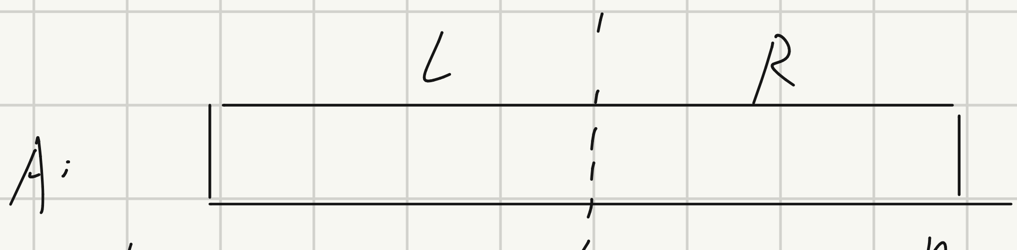
max inversions: C_n^2

Input an array A of n distinct integers,

Output the number of inversions in A

brute-force: $O(n^2)$

d&c: $O(n\log n)$



left inversion: $i<j\le \frac{n}{2}$

right inversion $\frac{n}{2} < i < j$

split inversion $i \le \frac{n}{2} \le j$

CountInv (A)

1. if (A) ≤ 2 , trivial.
2. leftInv = CountInv (left half of A)
3 rightInv = CountInv (right half of A)
4 splitInv = CountSplitInv (A).
5 return ... + ... + ...

max split: | 10 8 9 7 | | 4 1 3 2 |

$n/2 \times n/2 = n^2/4$

CountSplitInv (A)

```
splitInv = 0
i = 1
j = 1
for k = 1 to n
    if C[i] < D[j]
        i = i+1
    else
        splitInv = splitInv + (n/2 - i + 1)
        j = j+1
return splitInv.
```

Sort - and - CountInv (A)

1. if |A| ≤ 2 trivial.
2. (C, leftInv) = Sort - and - countInv (left half of A)
3 (D, rightInv) = sort - and - countInv (right half of A)
4 (B, SplitInv) = merge and - CountSplitInv

```
return (B, "  " + "  " + "  ")
```

```
Merge-and-CountSplitInv(C, D)
    i=1; j=1 ; splitInv = 0.
    for K=1 to n
        if C[i] < D[j]
            B[K] = C[i]
            i = i+1
        else  //  C[i] > D[j]
            B[K] = D[j]
            splitInv = splitInv + (n/2 - i + 1)
            j = j+1
    return (B, splitInv)
```

$$T(n) = 2T(n/2) + O(n) \Rightarrow T(n) = O(n \log n)$$
$$T(1) = C$$

Closest Pair Problem

Input: a set of n points on a plane

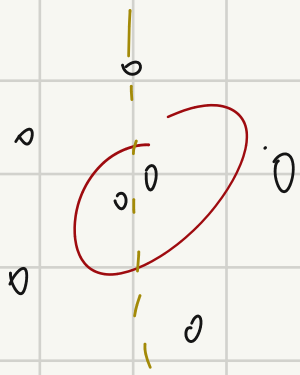
$p_i = (x_i, y_i)$ (assume $x_i \neq x_j$ and $y_i \neq y_j$ for any $i \neq j$)

Output: the pair (p_i, p_j) with smallest distance

$$d(p_i, p_j) := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

1-D  $O(n\log n)$ → sort

2-D



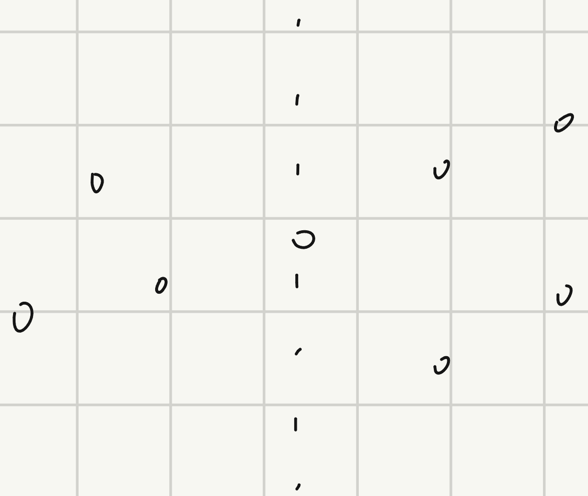
brute-force $O(n^2)$

d & c: $O(n\log n)$

left pair
right pair
split pair

closestPair (P)

1. if $|P| \le 3$ trivial
2. (p_1^l, p_2^l) = closestPair (left half of P)
3. (p_1^r, p_2^r) = closestPair (Right half of P)
4. (p_1^s, p_2^s) = closestSplitPair (P)
5. return the closest among (p_1^l, p_2^l) (p_1^r, p_2^r) (p_1^s, p_2^s)



\# splitpair : $\frac{n^2}{4}$

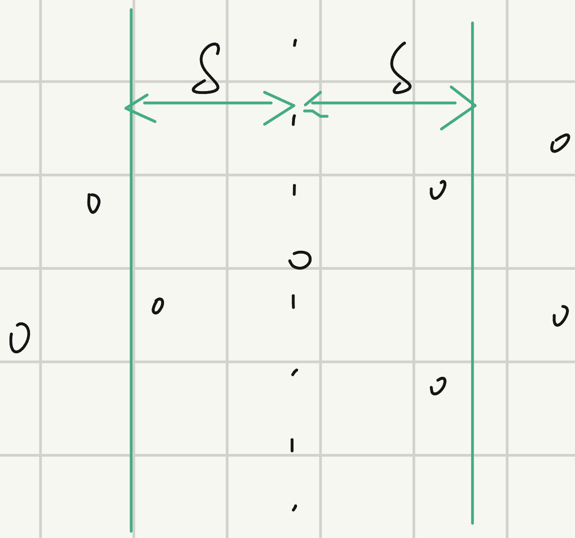
$\delta = \min (d(p_1^l, p_2^l), d(p_1^r, p_2^r))$

*: closestSplitPair consider only split pair with distance $< \delta$

↓

$O(n)$ such pair

$O(n)$ time

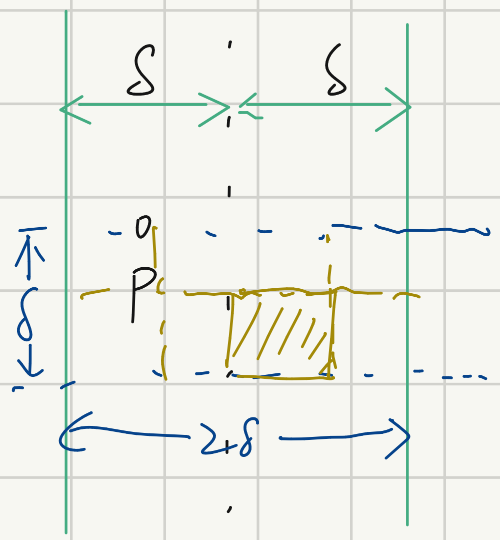


$d(p^l, p^r) < \delta$

$\Rightarrow |x^l - x^r| < \delta, \ |y^r - y^l| < \delta$

$\Downarrow$

...



each lattice has at most one dot

List of points in the strip sorted by y-coordinate

```
ClosestSplitPair (Sy)
    minDist := 0
    closestSplitPair = None
    for i=1 to K-1
        for j=1 to min {T, K-i}
            if d(qi, qi+j) < minDist
                minDist = d(qi, qi+j)
                closestSplitPair = (qi, qi+j)
return closestSplitPair
```

```
ClosestPair. (Px, Py)
if |P| ≤ 3, trival
```

obtained from P_x, P_y in $O(n)$ time

2. L_x = Points on left half, sort by x
   L_y = Points on y
   R_x = right sorted by x
   R_y = right y

3. (p_1^l, p_2^l) = closestPair (L_x, L_y)
4. (p_1^r, p_2^r) = closestPair (R_x, R_y)
5. δ = $\min\{d(p_1^l, p_2^l), d(p_1^r, p_2^r)\}$
6. S_y = points in the strip $(\bar{x}-\delta, \bar{x}+\delta)$, sorted by y
7. (p_1^s, p_2^s) = closestSplitPair (S_y)
8. return the closest pair of ( ), ( ), ( )

The result returned by split isn't necessarily the closest cause that the range is $\delta \times 2\delta$

$T(n) = T(n/2) + O(n) \Rightarrow T(n) = O(\log n)$
$T(3) = c$

\# recursive calls

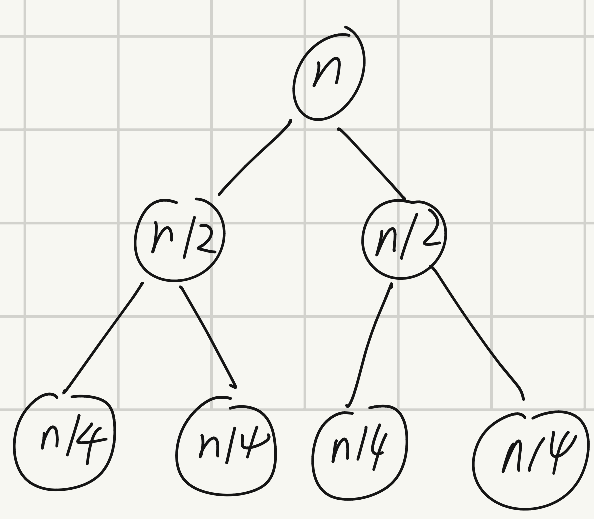
$T(n) = a\,T(n/b) + n^d$ ⟹ time requiring except for recursion

b ⟹ input size shrinking factor

$T(1) = c$ ⟹ constant 1

$T(n) = 2T(n/2) + n$
$T(1) = 1$

$\log_2^n$

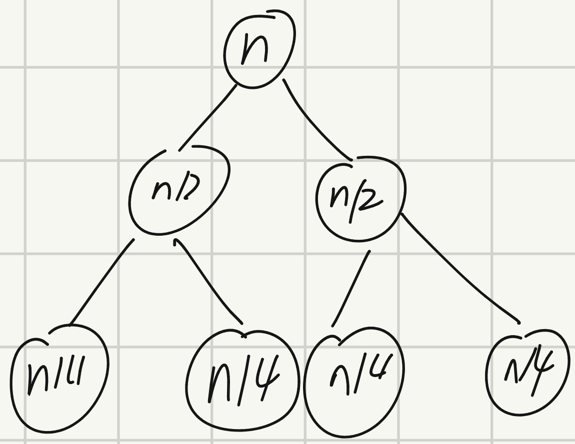


n
n
n

$n(\log n)$

$$\begin{cases} T(n) = 2T(n/2) + n^2 \\ T(1) = 1 \end{cases}$$

recursion tree



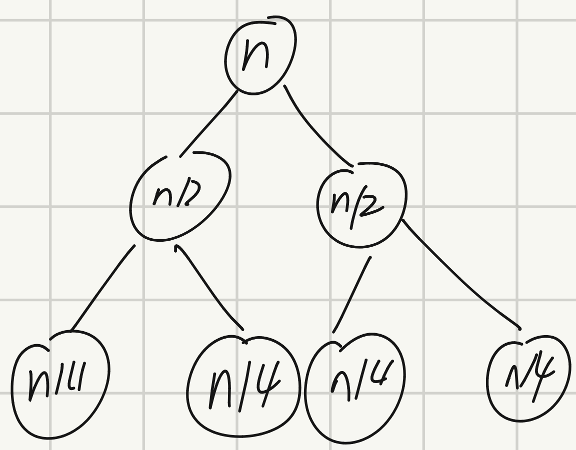
n^2

$(\frac{n}{2})^2 \times 2 = \frac{n^2}{2}$

$(\frac{n}{4})^2 \times 4 = \frac{n^2}{4}$

$(\frac{n}{2})^i \times 2^i = \frac{n^2}{2^i}$

$$\text{total} = \sum_{i=0}^{\log n} \frac{n^2}{2^i} < 2n^2 = O(n^2)$$



$\sqrt{n}$

$\sqrt{n/2} \times 2 = \sqrt{2n}$

$\sqrt{n/4} \times 4 = \sqrt{4n}$

$\sqrt{n/2^i} \times 2^i = \sqrt{2^i n}$

$$\text{total} = \sum_{i=0}^{\log_2 n} \sqrt{n} \cdot (\sqrt{2})^i$$

$$= \sqrt{n}\,\frac{1-(\sqrt{2})^{\log_2 n+1}}{1-\sqrt{2}} \qquad \underbrace{\sqrt{2}^{\log_2 n} \cdot \sqrt{n} = n}_{\text{last one}}$$

$$= \frac{\sqrt{2}}{\sqrt{2}-1}\sqrt{n}\cdot\sqrt{n} = O(n)$$

① 等 ： 由层数决定

② 等比 $\downarrow$ 最上层

③ 等比 $\uparrow$ ： 最底层决定

Form 1: $T(n) = aT(\frac{n}{b}) + O(n^{\alpha})$ $\quad T(1) = O(1)$

1) $a = b^{\alpha}$

$T(n) = O(n^{\alpha} \log n)$

2) $a < b^{\alpha}$

$T(n) = O(n^{\alpha})$

3) $a > b^{\alpha}$

$T(n) = O(n^{\log_b a})$

Form 2. $T(n) = aT(\frac{n}{b}) + f(n)$ $\quad T(1) = O(1)$

1) $a \cdot f(\frac{n}{b}) = f(n)$

$T(n) = O(f(n) \cdot \log n)$

2) $a \cdot f(\frac{n}{b}) = r \cdot f(n)$ $\quad r < 1$

$T(n) = O(f(n))$

3) $a \cdot f(\frac{n}{b}) = r f(n)$ $\quad r > 1$

$T(n) = O(n^{\log_b a})$