Lecture 4

SELECT $(\{a_1 .. a_n\}, k)$

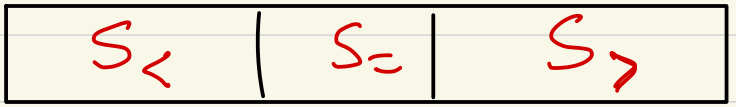
* PIVOT $v \leftarrow$ random element from $\{a_1 .. a_n\}$

* SPLIT $S_< \leftarrow \{a_i \mid a_i < v\}$

$S_= \leftarrow \{a_i \mid a_i = v\}$

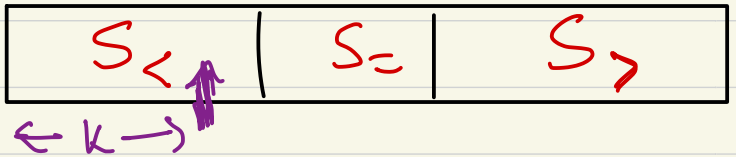
$S_> \leftarrow \{a_i \mid a_i > v\}$

IDEA: IMAGINE THE SORTED LIST



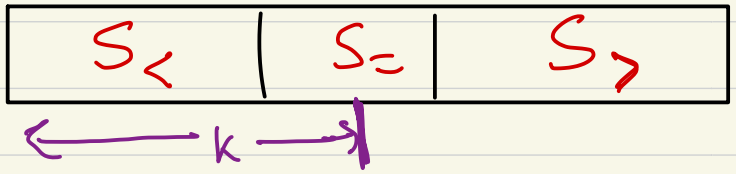
* CASES:

- $k \leq |S_<|$: return SELECT$(S_<, k)$



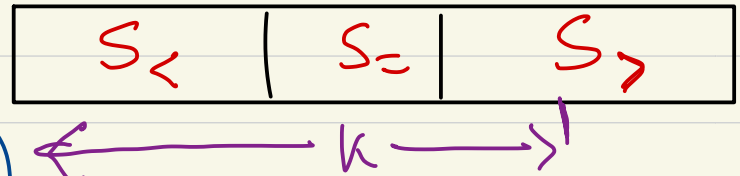
- $|S_<| < k \leq |S_<| + |S_=|$:

return v



- $|S_<| + |S_=| < k$

return SELECT$(S_>, k - |S_<| - |S_=|)$



Runtime Analysis

Randomised Algorithm: runtime depends on pivot choices.

For example:

[Best case]

→ If pivot $v = k^{th}$ smallest element

⇒ SELECT turns in CASE 2, returns $O(n)$ time

[Worst case]

→ Imagine $k = n/2$ and each pivot is largest or smallest element

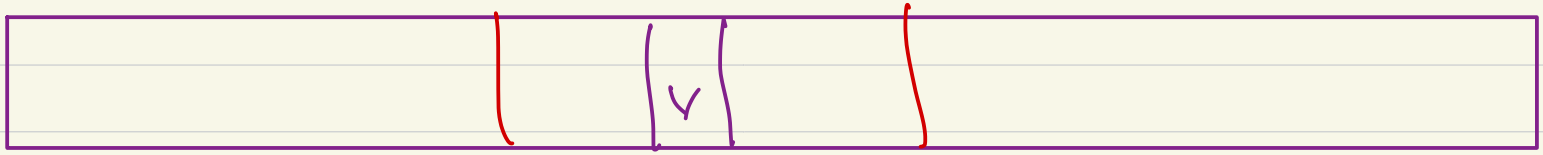
$\Theta(n^2)$ time

→ Happens with very low probability

Define $T[n] =$ Expected runtime of SELECT$(\{a_1 \ldots a_n\}, k)$

INTUITION: Why should SELECT terminate quickly?

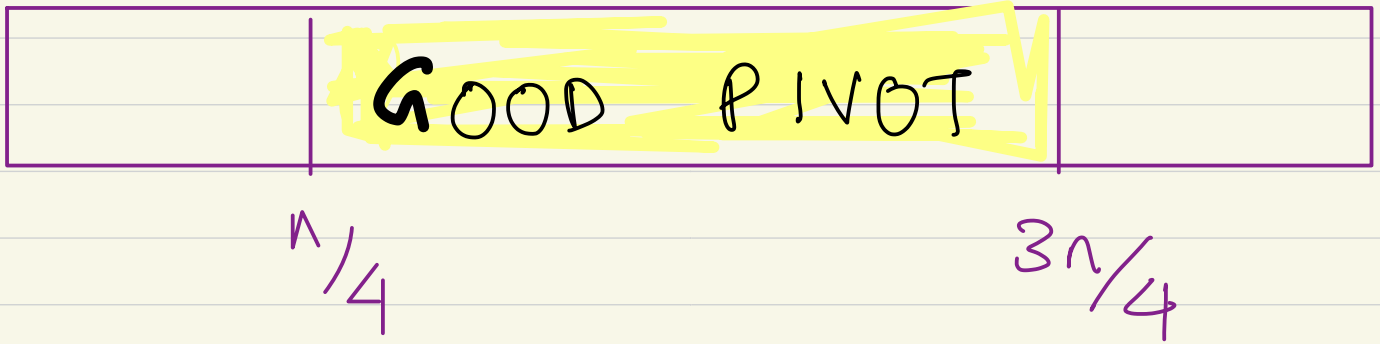
Imagine the sorted list



With good chance, random pivot v is somewhere in middle, $\Rightarrow$ $S_<, S_>$ are

substantially smaller.

Def: [GOOD PIVOT] A pivot element v is a good pivot if v is between $n/4^{th}$ smallest to $3n/4$ smallest



Obs: 1) List size decreases to $\leq 3n/4$

whenever pivot is GOOD.

2) # of good pivots $= n/2$

$T[n]$ = Expected runtime of Select $\{a_1, \ldots a_n\}$

$$= \left(\begin{array}{c}\text{Expected runtime before} \\ \text{first good pivot}\end{array}\right)$$

$$\leq \left(\begin{array}{c}\text{Expected \# of pivots before} \\ \text{first good pivot}\end{array}\right) * n$$

$$\| $$

$$\left(\begin{array}{c}\text{Expected \# of tosses before} \\ \text{HEADS}\end{array}\right) = 2$$

$$+ \left(\begin{array}{c}\text{Expected runtime after} \\ \text{first good pivot}\end{array}\right)$$

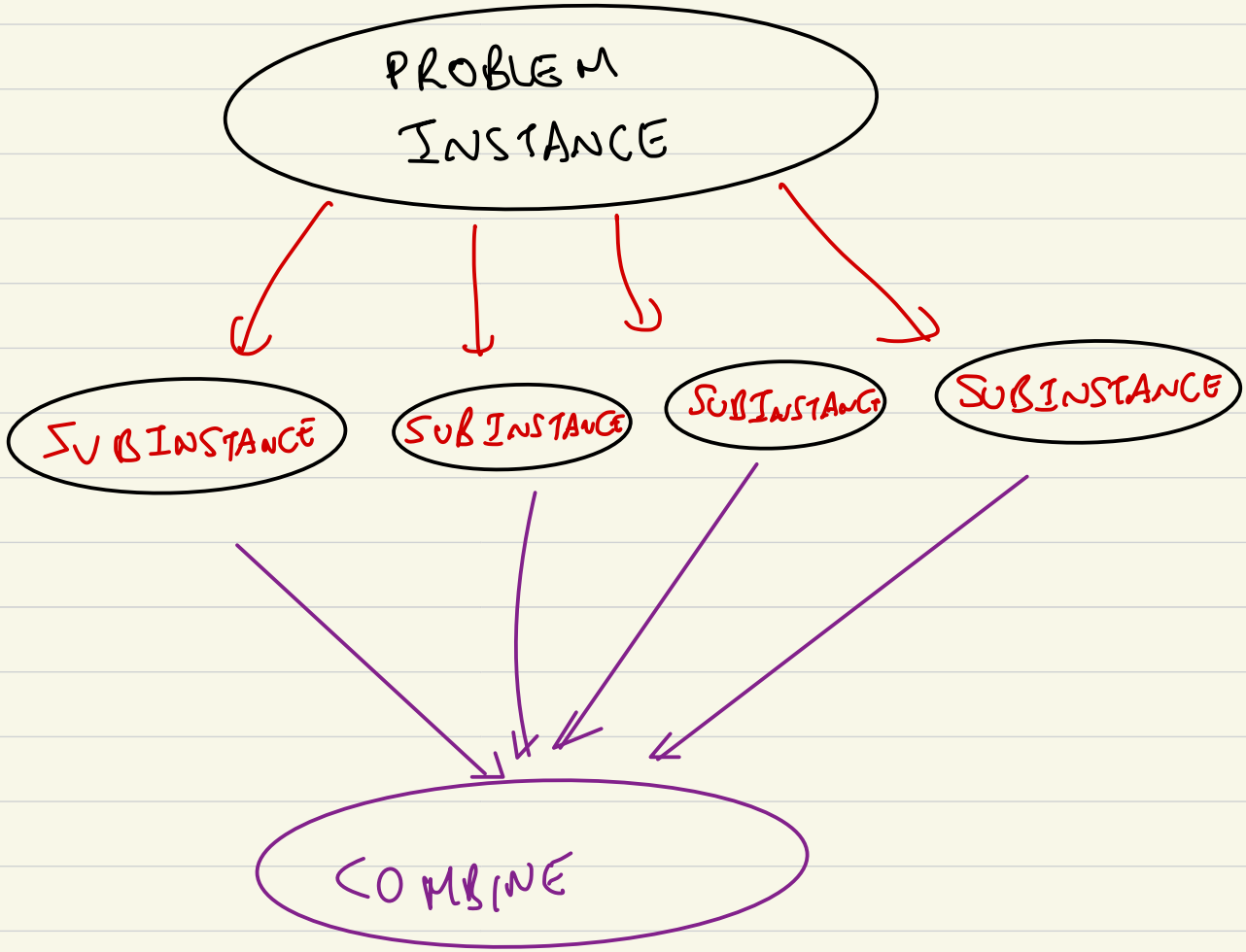
$$\leq T[3n/4]$$

$$T[n] \leq T[3n/4] + O(n)$$

By Master's theorem

$$\Rightarrow T[n] = \Theta(n)$$

Divide and conquer



Solve → recursively

Exponentiation:

INPUT: Number n

OUTPUT: 2^n in decimal (array of digits)

$$2^{100} = \underbrace{2 \cdot 2 \cdot 2 \cdots 2}_{100}$$

$$= (2^{50})^2$$

$$2^{50} = (2^{25})^2$$

$$2^{25} = (2^{12})^2 \cdot 2$$

$$2^{12} = (2^6)^2$$

$$2^6 = (2^3)^2$$

$$2^3 = (2^1)^2 \cdot 2$$

$$2^1 = 2$$

```
Exp( a, n : integer)
    If  n=1   return  a
    b =   Exp(a, ⌊n/2⌋)
    if (n even)   return  b*b
       (n odd)    return  b*b*a
```

RUN TIME: $T[n] = T\left[\left\lfloor \frac{n}{2} \right\rfloor\right] + \Theta(M(n))$

where $M(n)$ = time for a n-digit multiplication

Since $M(n) > \Theta(n)$

$$T[n] = \Theta(M(n)).$$

BINARY TO DECIMAL

INPUT : $B[1..n]$: an array of bits representing number B in binary

OUTPUT: $D[1..m]$: decimal representation of number B as a sequence of digits

Naive Algorithm

$$(1011011) = 1\cdot 2^6 + 0\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2 + 1\cdot 1$$

$$= 1\cdot 64 + 0\cdot 32 + 1\cdot 16 + 1\cdot 8 + 0\cdot 4 + 1\cdot 2 + 1$$

$$= 91$$

$$B = 2^{n-1}\cdot B[1] + 2^{n-2}\cdot B[2] + 2^{n-3}\cdot B[3] + \ldots + B[n]$$

← n additions →

RUNTIME : $\Theta(n)$ additions $\Rightarrow$ $\Omega(n^2)$ time.

How many digits? $B \leq 2^n$

$\Rightarrow \quad m = \log_{10} B = \log_2 B / \log_2 10 = (0.3010)\, n$

BINARY TO DECIMAL

INPUT: $B[1..n]$: an array of bits representing number B in binary

OUTPUT: $D[1..m]$: decimal representation of number B as a sequence of digits

Divide & Conquer Approach:

$$(1011\ 1100)_2 = (1011)_2 \cdot 2^4 + (1100)_2$$

a_L a_R

$$= (11) \cdot 16 + (12)$$

$$= 176 + 12$$

$$= 188$$

BINARY TO DECIMAL ($a[1..n]$: array of bits)

SPLIT: $a_L \leftarrow a[1..\lceil n/2 \rceil]$

$a_R \leftarrow a[\lceil n/2 \rceil + 1, .. n]$

RECURSE: $d_L \leftarrow$ BINARY TO DECIMAL (a_L)

$d_R \leftarrow$ BINARY TO DECIMAL (a_R)

$c \leftarrow$ EXP $(2, \lceil n/2 \rceil)$

COMBINE: return $c * d_L + d_R$

RUNTIME:

$$T[n] = 2\,T[n/2] + \Theta(\text{n-digit multiplication}) +$$

$$\Theta(\text{exponentiation}) + \Theta(\text{addition})$$

$$= 2\,T[n/2] + \Theta(M(n))$$

where $M(n)$ = time for n-digit multiplication

If $M(n) > n^{1.00001}$ then

$$T(n) = \Theta(M(n))$$

CLOSEST PAIR

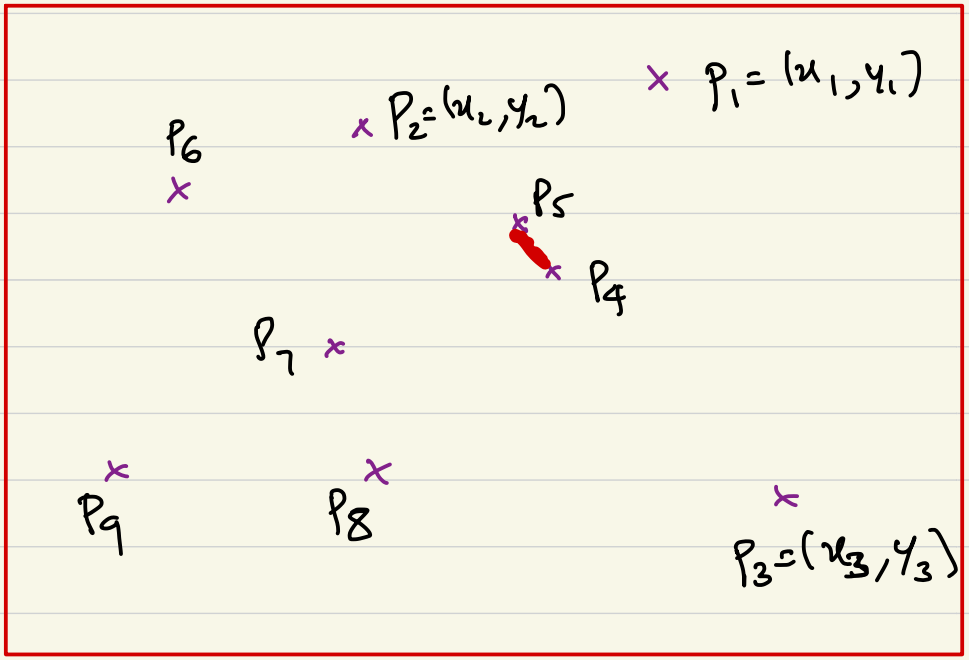
INPUT: n points on plane $\{ p_i = (x_i, y_i) \mid i = 1 \ldots n \}$

OUTPUT: Closest pair of points (p_i, p_j)

Naive Algorithm:

Try every pair of points

$\rightarrow \Theta(n^2)$ time algorithm.



CLOSEST $\left(\{p_i = (x_i, y_i) \mid i = 1..n\}\right)$

- Sort $\{p_1 \ldots p_n\}$ in increasing x.
- $(p_L, p_{L'}) \leftarrow$ CLOSEST$(\{p_1 \ldots p_{n/2}\})$

  $(p_R, p_{R'}) \leftarrow$ CLOSEST $(\{p_{n/2+1} \ldots p_n\})$

  let $d_L =$ distance $(p_L, p_{L'})$

  $d_R =$ distance $(p_R, p_{R'})$

  let $d = \min\{d_L, d_R\}$
- $A_L \leftarrow S_L$ restricted to strip $[x_{n/2} - d, x_{n/2} + d]$

  $A_R \leftarrow S_R$ restricted to strip $[x_{n/2} - d, x_{n/2} + d]$
- Sort A_L, A_R in increasing y.

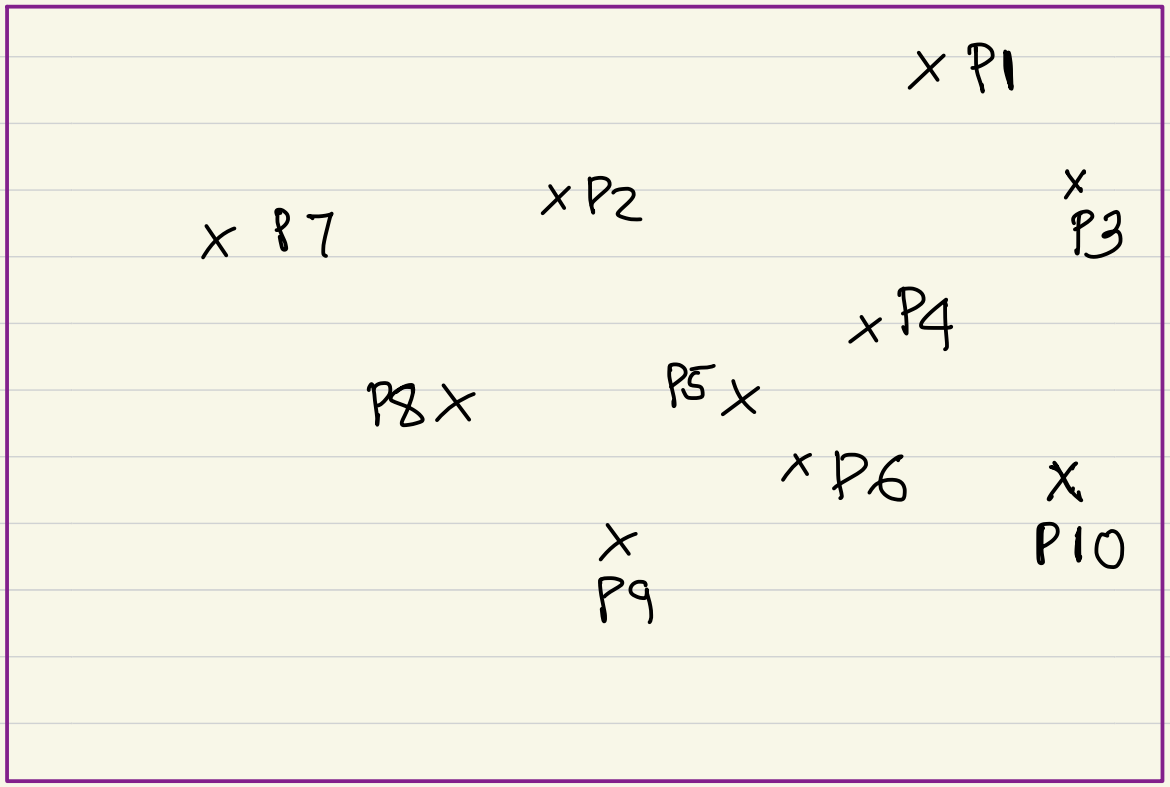
- For each $p=(x,y)$ in A_2

  $R \leftarrow \{ q=(x',y') \text{ in } A_R \text{ with } y-d \leq y' \leq y+d \}$.

  Compute all distances between p AND $q \in R$ and update minimum.

DIVIDE AND CONQUER

INPUT: {P1, P2, P3, .. P9, P10}



How to divide?

* Split the list

A = {P1, P2, P3, P4, P5}

B = {P6, P7, P8, P9, P10}

* SPLIT THE PLANE :

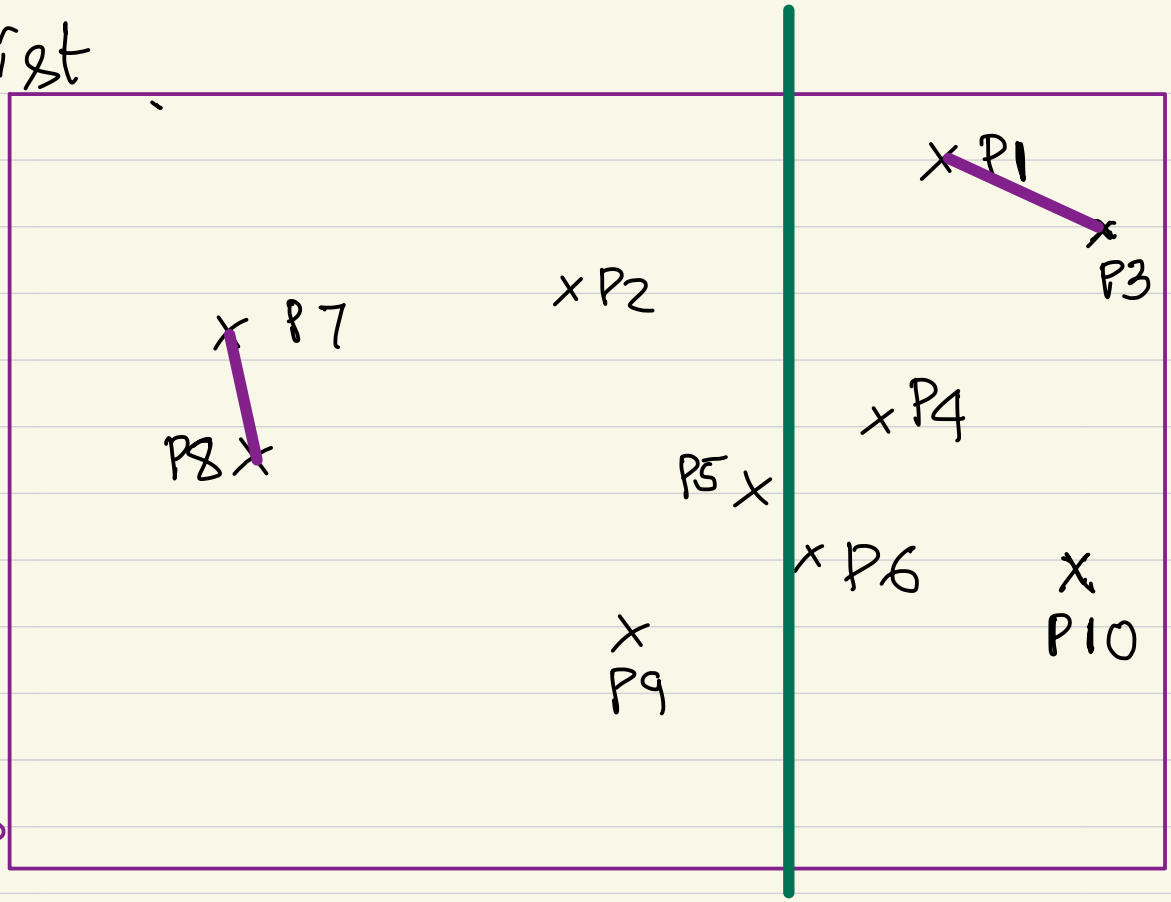
1) Sort the points {$p_i = (x_i, y_i)$} based on x-co-ordinate

{P7, P8, P2, P9, P5, P6, P4, P1, P10, P3}

2) Split the sorted list

$S_{LEFT} = \{P7, P8, P2, P9, P5\}$

$S_{RIGHT} = \{P6, P4, P1, P10, P3\}$



IMAGINE THAT:

1) We solved two subproblems recursively

and found closest pair in S_{LEFT} & S_{RIGHT}.

$CLOSEST\ PAIR\ (S_{LEFT}) = (P_L, P_{L'})$ (P_7, P_8)

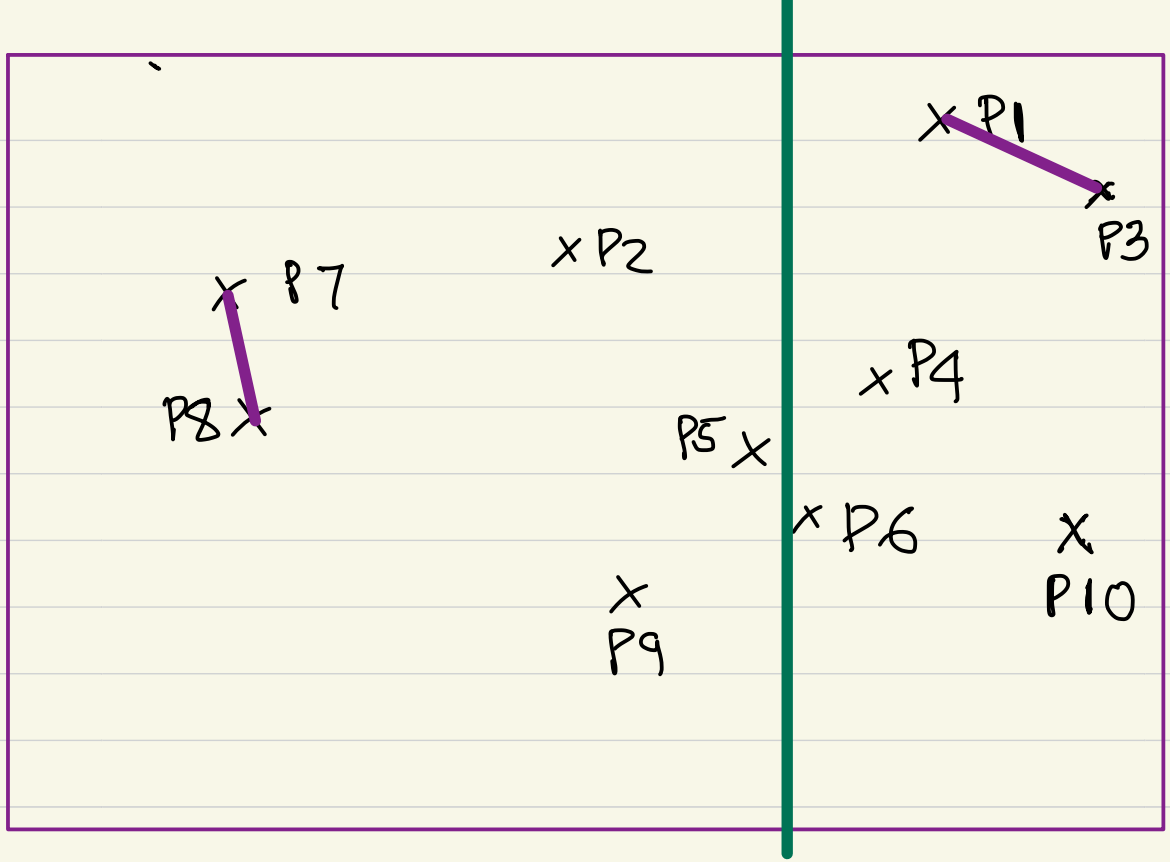
$CLOSEST\ PAIR\ (S_{RIGHT}) = (P_R, P_{R'})$ (P_1, P_3)

Among the pair, say (P_7, P_8) is closer,

let

$d \leftarrow$ distance (P_7, P_8)

Closest pair might be across S_{LEFT}, S_{RIGHT}

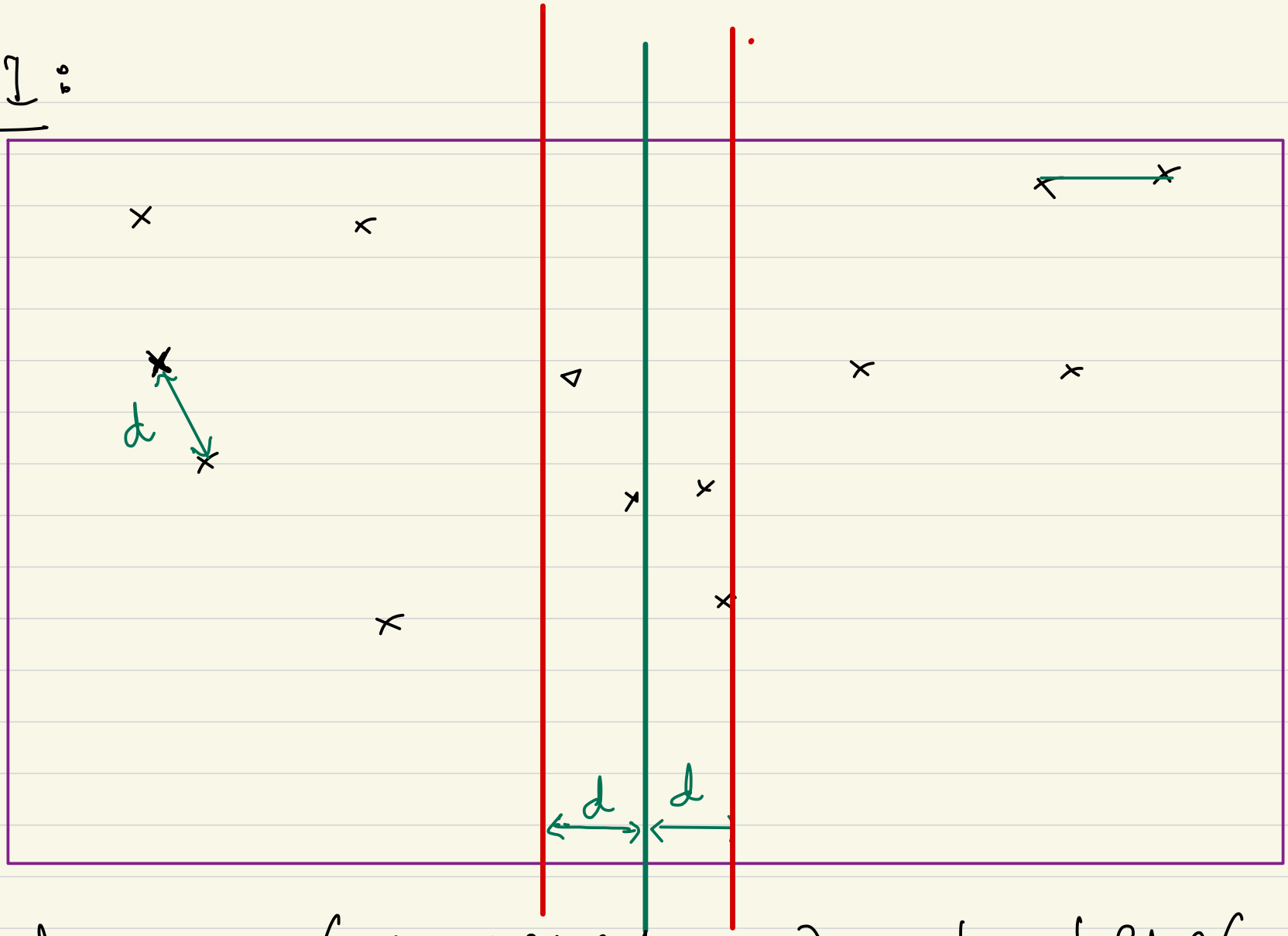


Naive: Try every pair

$P_L \in S_{LEFT}$ $P_R \in S_{RIGHT}$

$n/2 \times n/2 = \Theta(n^2)$ pairs

IDEA I:



If $d \leftarrow \min(\text{closestPAIR}(S_{LEFT}), \text{closestPAIR}(S_{RIGHT}))$

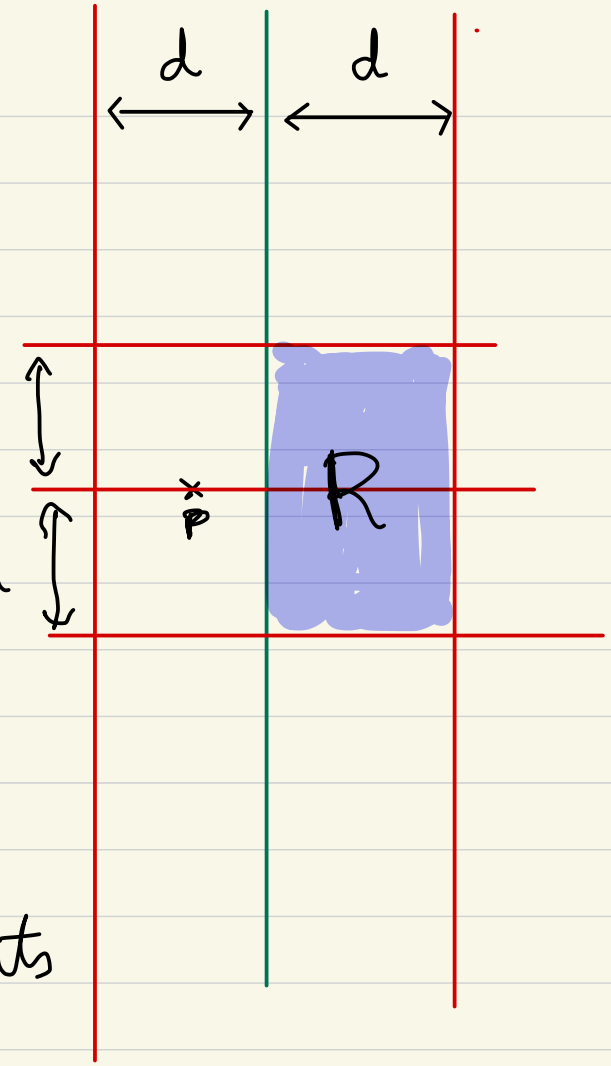
then restrict attention to a strip of width d on each side.

i.e. Throw away points outside the strip !!

IDEA 2:

A point P only needs to be compared with points in region R

How many points in R?



Claim: At most 8 points in region R.

Assuming Claim, every point only needs comparision with 8 other points.

1) All points are at least d - from each other

2) Every pair inside a square $< d$

$\Rightarrow$ At most 1 point per square

$\Rightarrow$ At most 8 points in region.