Randomized Algorithms: 2-dim Linear Programming

Deterministic algorithms: same behaviour on a fixed input

Randomized algorithms: random decisions

→ running time → random variable

→ output can be random

- With high probability, running time is small
- W.h.p., Output is correct.

Randomized Quicksort (randomly choose pivot)

Sampling, Avoid worst cases

- with some prob, hardware can fail.
- Algorithms: repeat multiple times
  $\Rightarrow$ error probability very small

Many cases: randomized algorithms are the fastest known.
Also, simplest.

2-dim Linear Programming

x, y

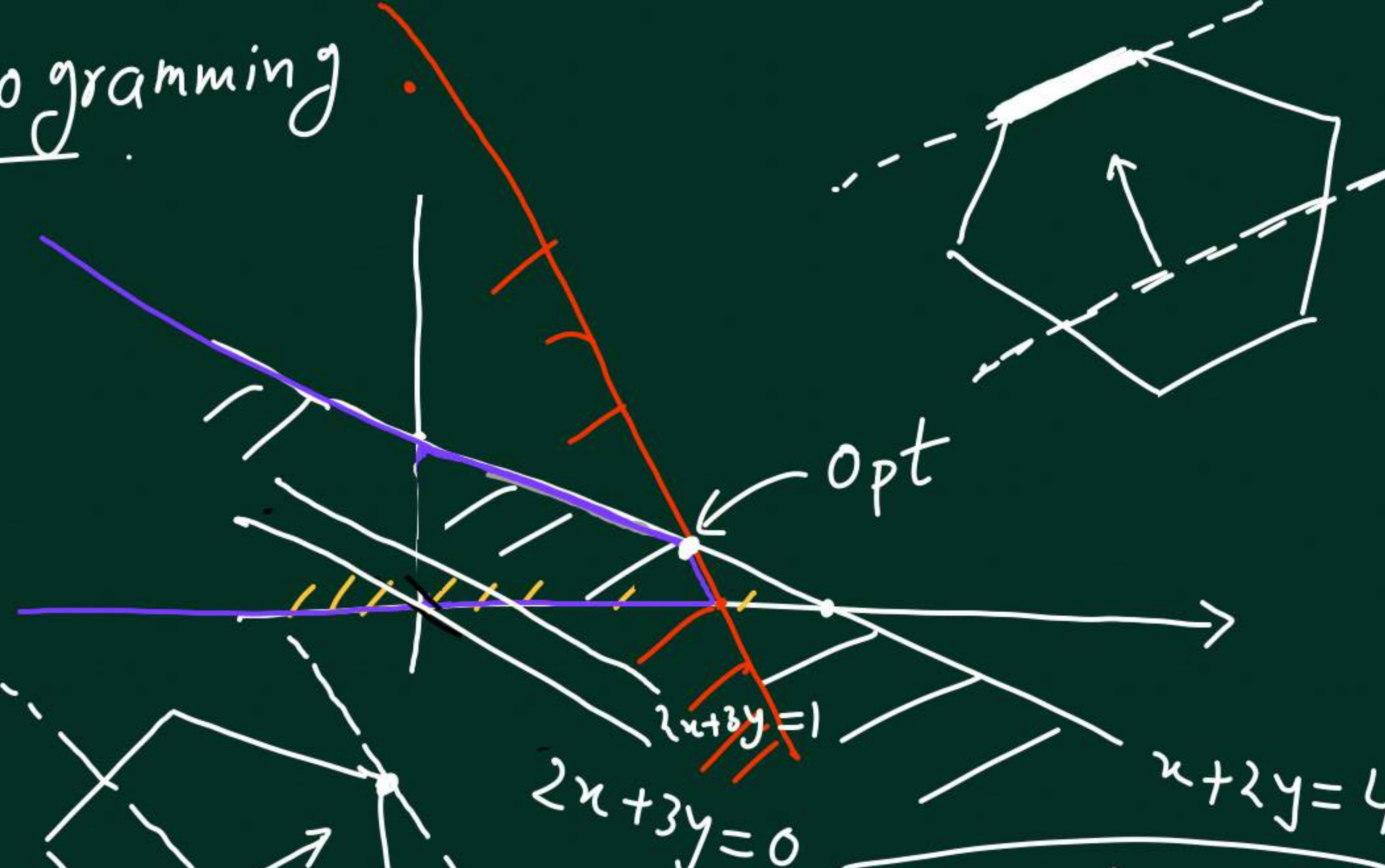
$x + 2y \le 4$

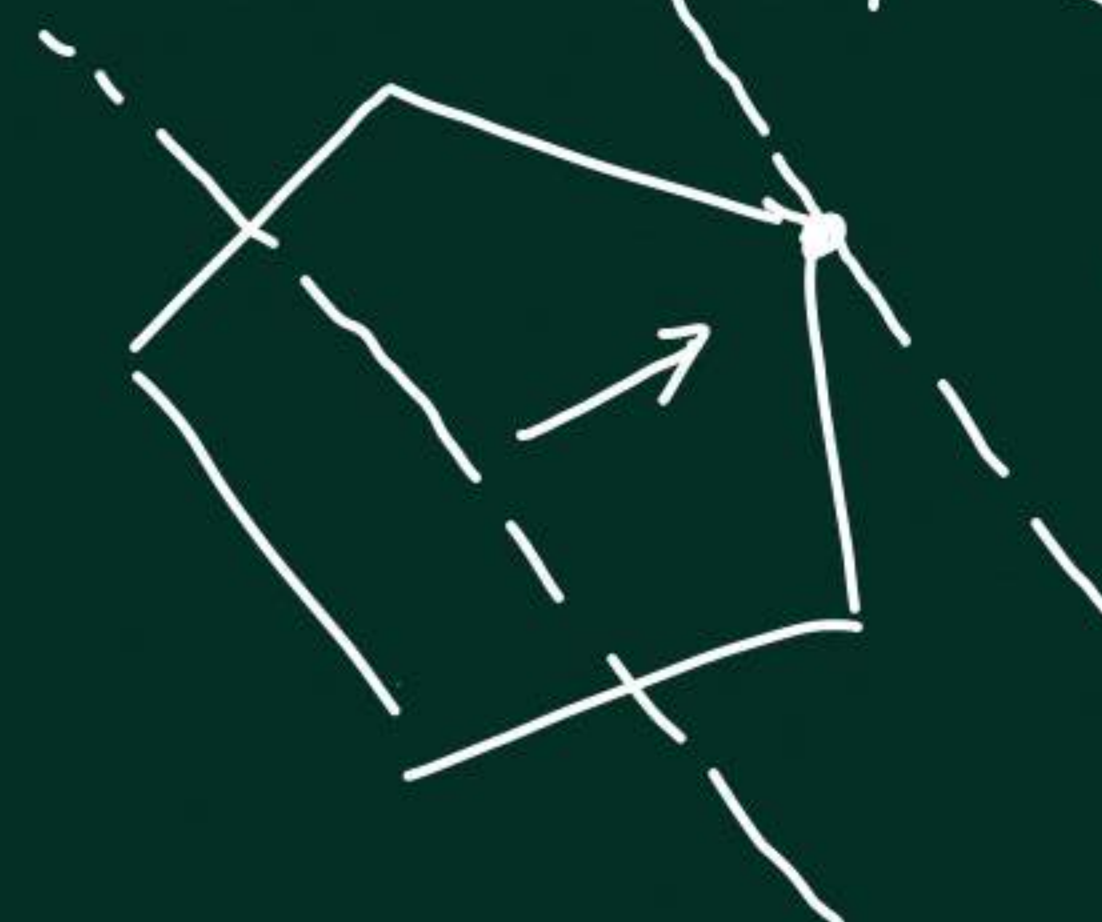
$2x + y \le 6$

$y \ge 0$

max $2x + 3y$

$(8/3, 2/3)$ Opt $\frac{22}{3}$





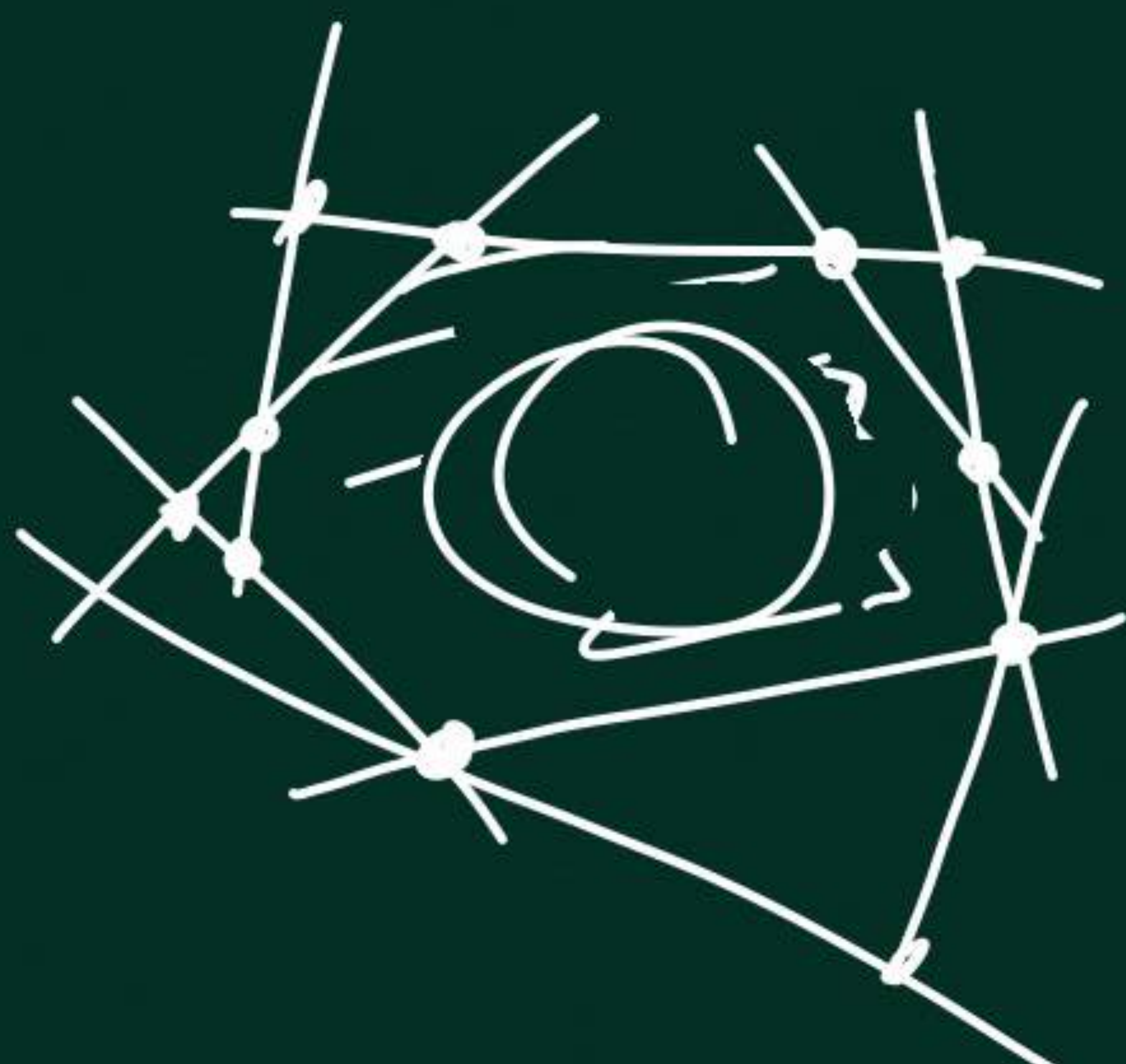
One of corners is an optimal point.

HW

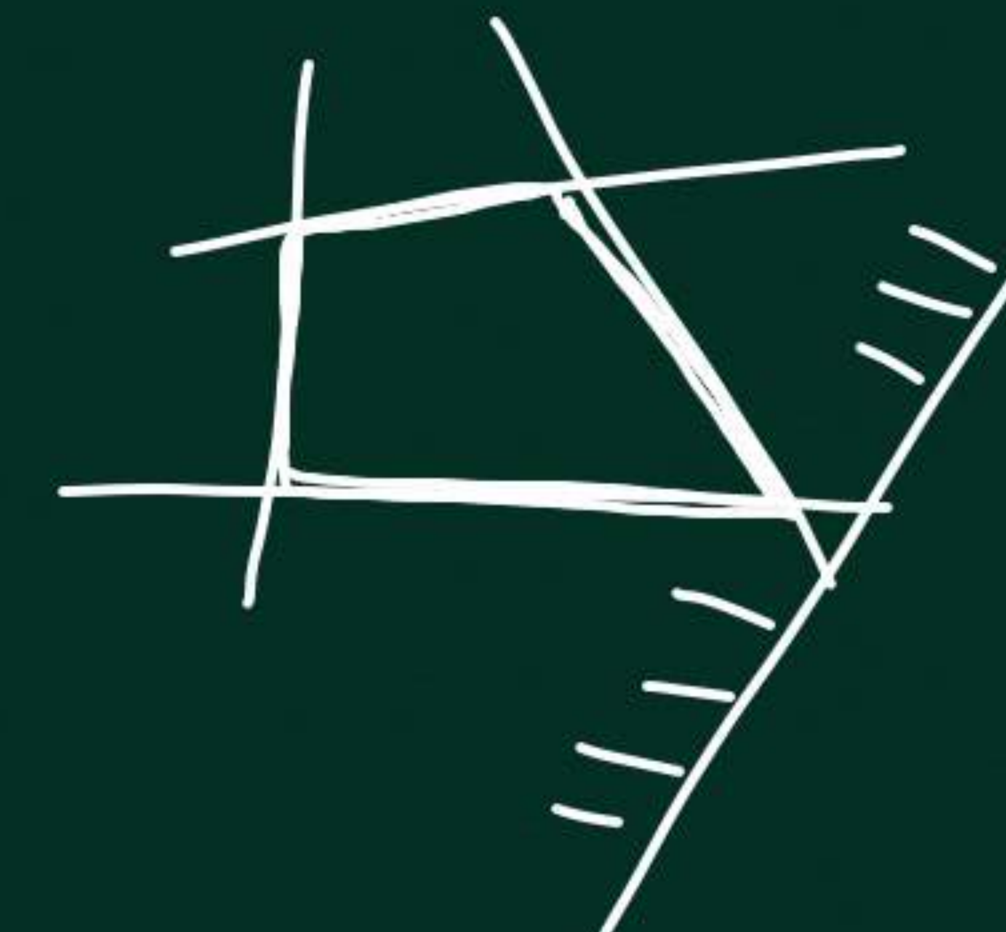
n linear constraints.

check all the corners $\quad O(n^3)$

at most n



no. of intersection points $O(n^2)$



HW $O(n \log n)$ compute all corners.

Randomization simple algorithm

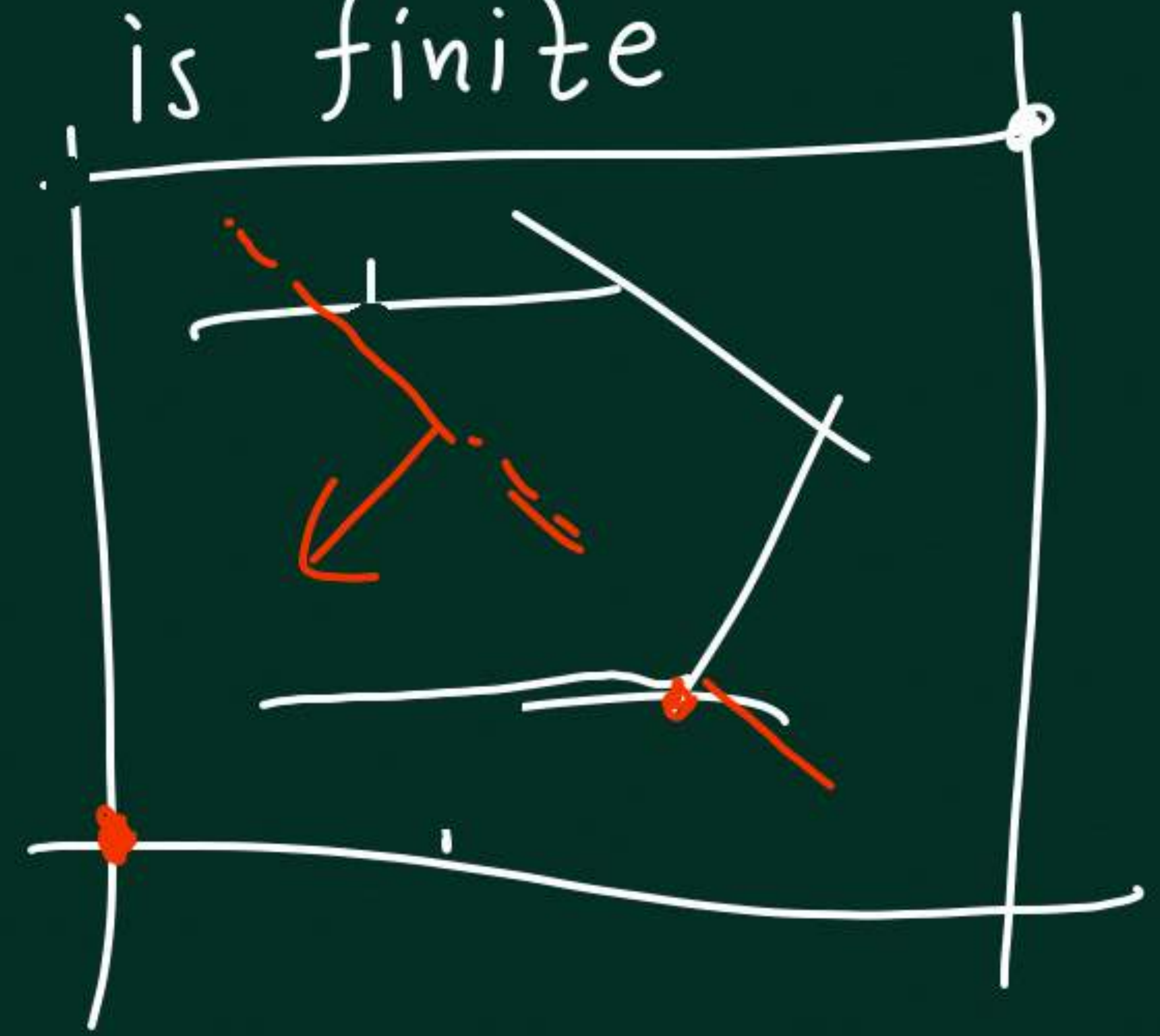
Find the optimal point $O(n)$ time (w.g.p.)

Assumptions: Optimal value is finite

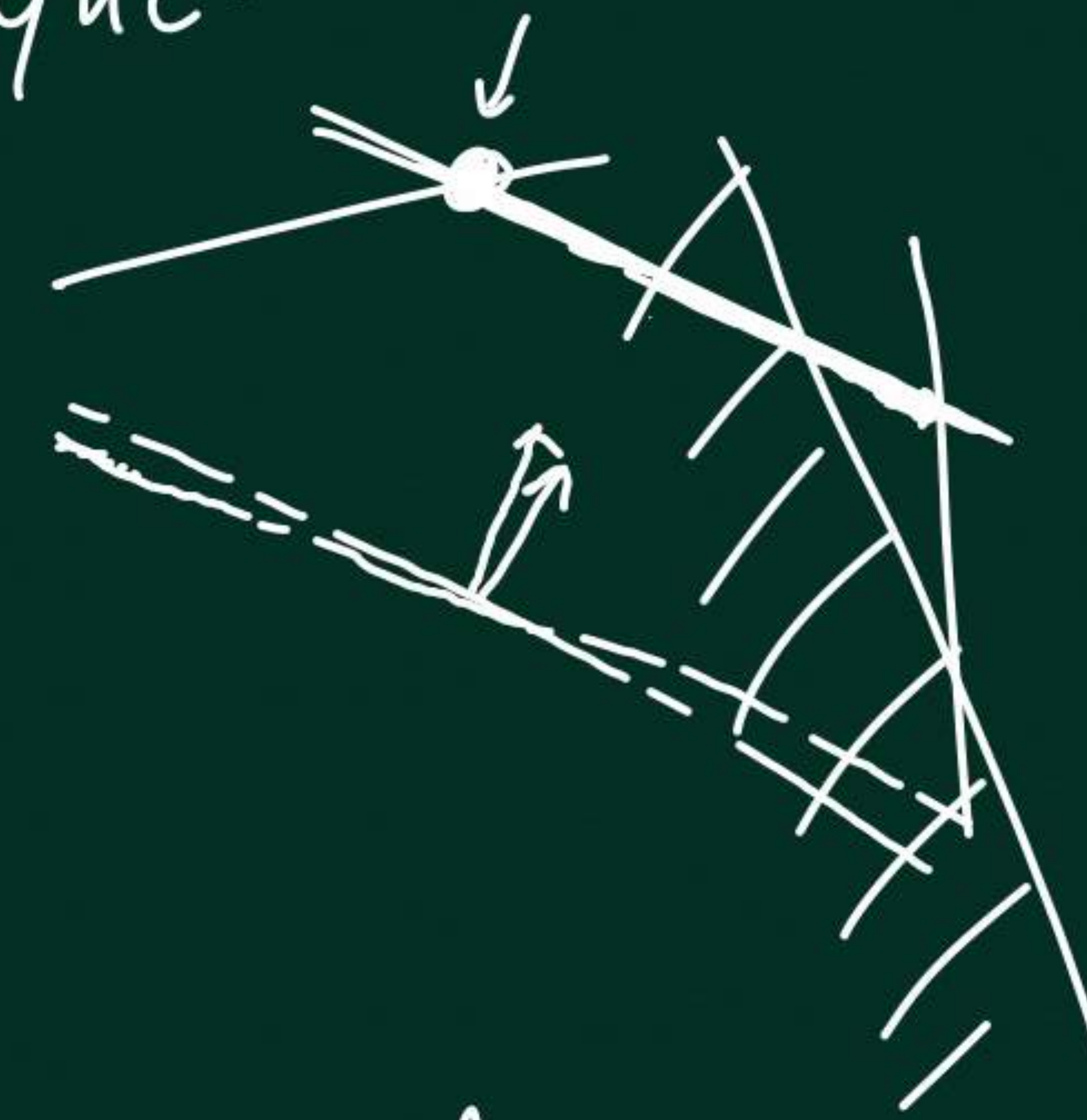
for every intersection point

$-L < x\text{-coord} < L$

$-L < y\text{-coord} < L$



Assumption: optimal solution is unique.



$C_0 \leftarrow$ Set of constraints $\leftarrow$ $-L \le x \le L$

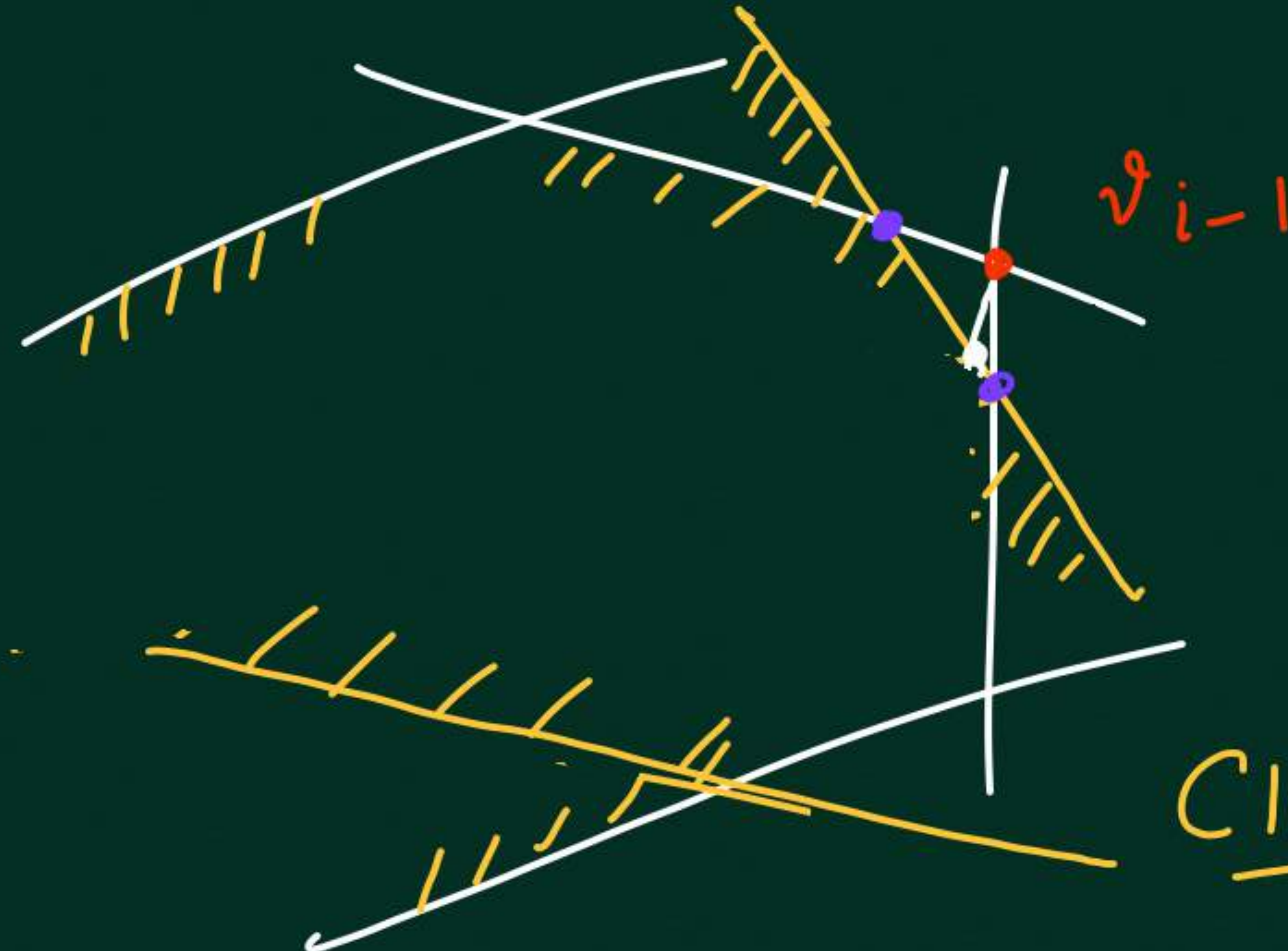
$-L \le y \le L$

$v_0 \leftarrow$ Optimal point $\leftarrow (L, L)$

Add constraints one by one and update the optimal point

$$(C_{i-1}, v_{i-1}) \longrightarrow (C_i, v_i)$$

$$C_i = C_{i-1} \cup \{h_i\}$$



If v_{i-1} satisfies h_i $O(1)$
then $v_i = v_{i-1}$

If v_{i-1} doesn't satisfy h_i $O(i)$

Claim: v_i will satisfy h_i with equality

Maximization

$f(p) = c\,p_x + d\,p_y$

objective function

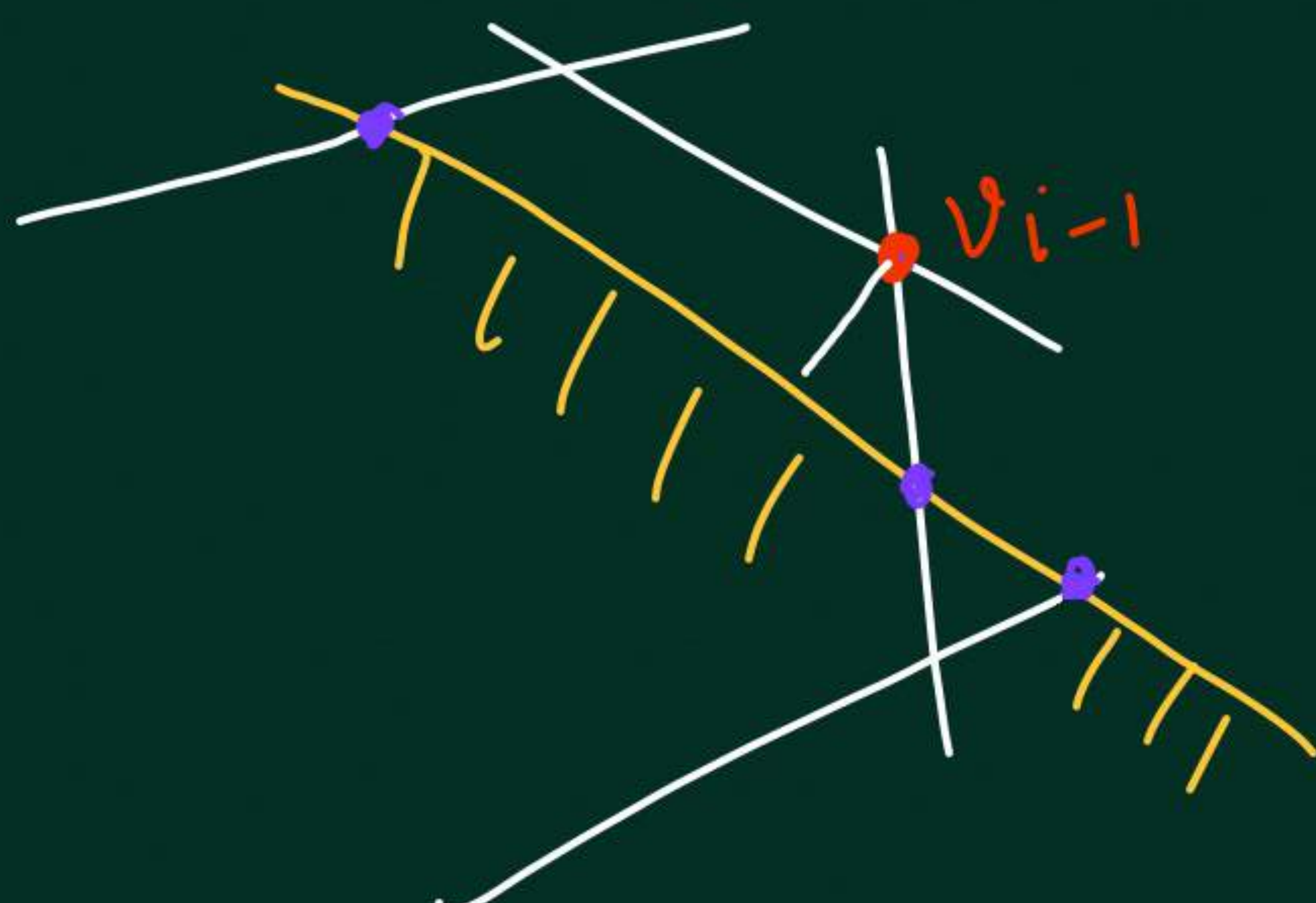
Obs $f(v_{i-1}) \geq f(v_i)$

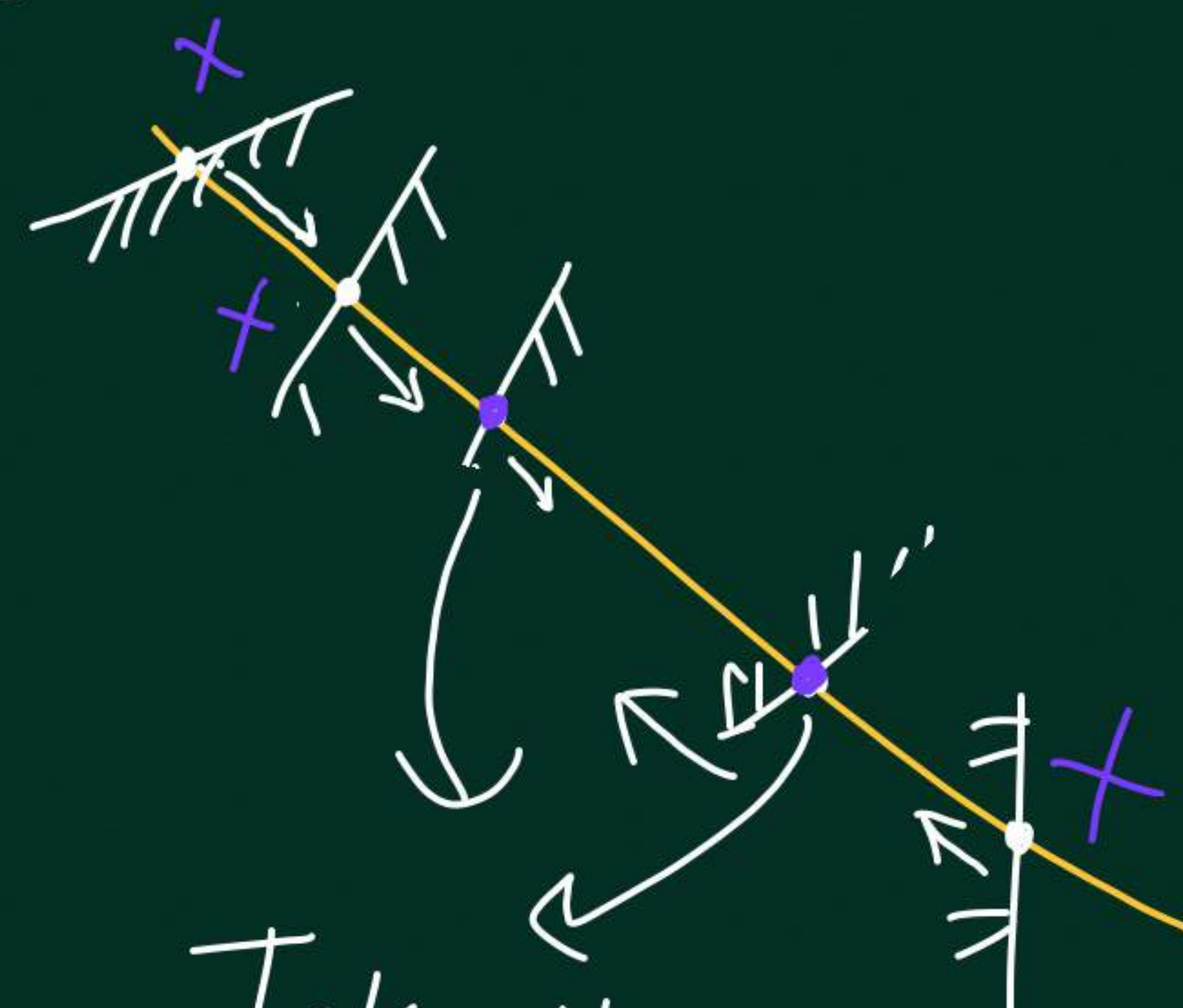
Consider line segment joining v_i and v_{i-1}.
Intersection of line segment with h_i is q.

$f(v_{i-1}) \geq f(q) \geq f(v_i)$

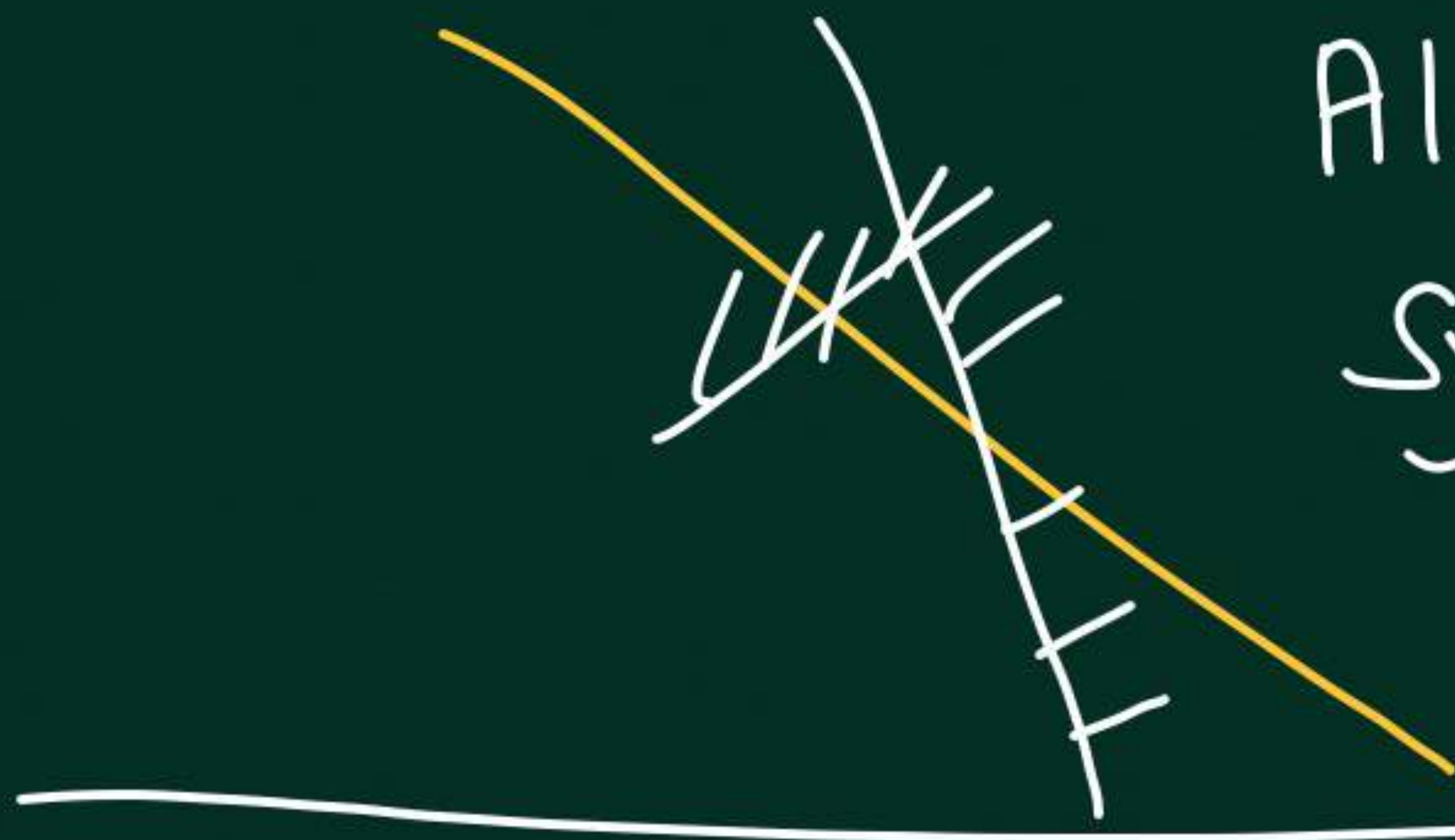
$O(i)$ time update algorithm?

<u>Obs</u> v_i is the point of intersection of h_i and Some h_ℓ $1 \leq \ell \leq i-1$.





Take the best of two $\leftarrow v_i$



Also detect if system becomes infeasible.

i^{th} iteration $\leftarrow$ $i-1$ intersection point

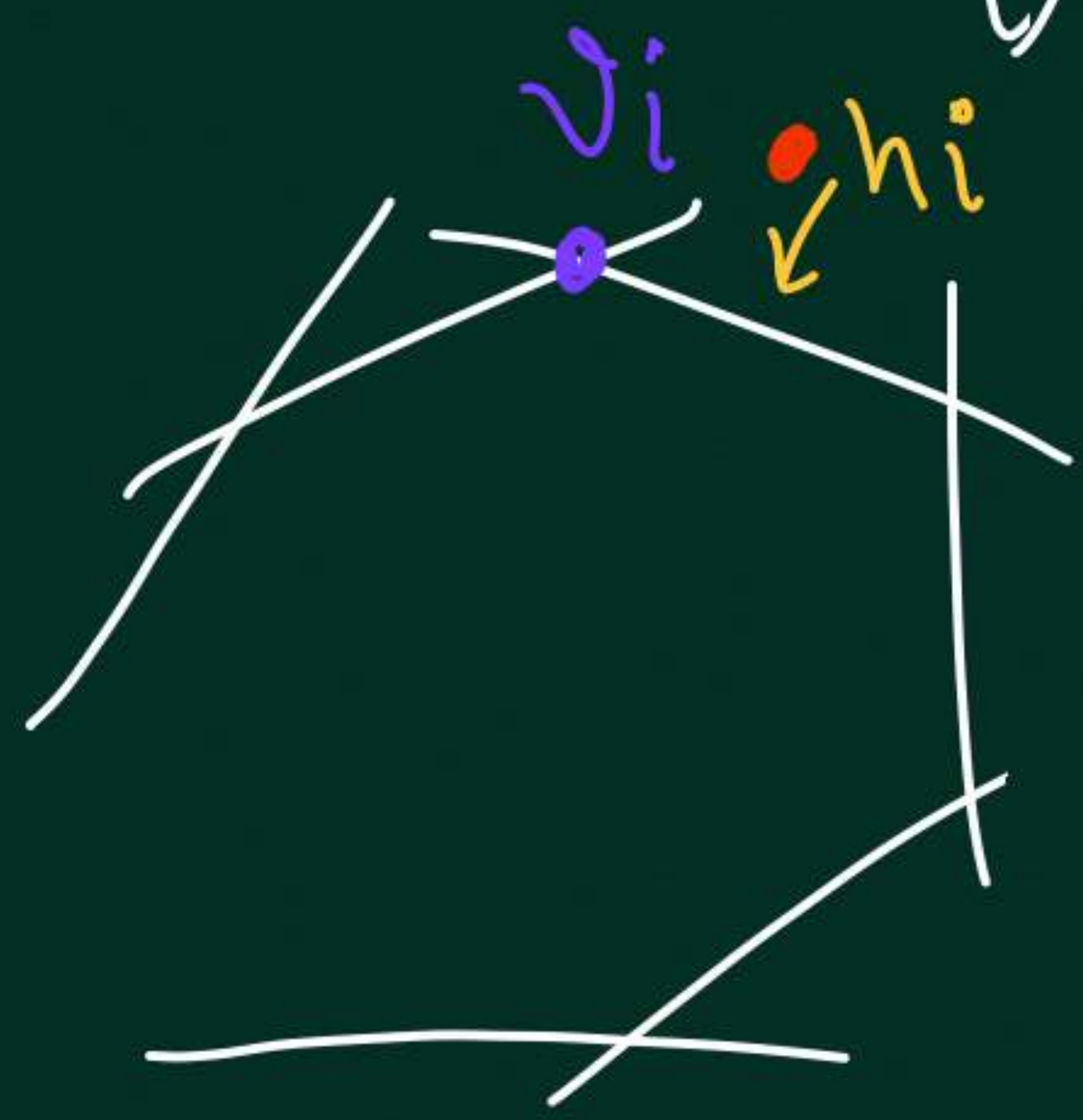
Total running time $= O(1) + O(2) + \dots + O(i) \dots + O(n)$

$= O(n^2)$

We arrange the lines in a random order and then run the algorithm.

<u>Que</u> What is the probability that

$v_i \neq v_{i-1}$ (bad event)



= prob that i^{th} line is one of two lines passing through v_i

$= \frac{2}{i}$

Expected number of times i have computed intersections of lines.

X_i = the no. of intersections computed in the i^{th} iteration.

$$X_i = \begin{cases} 0 & \text{with pr } 1 - \frac{2}{i} \\ i-1 & \text{pr } 2/i \end{cases}$$

$$X = \sum_{i=1}^{n} X_i \qquad \mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i} \frac{2}{i}(i-1) \le 2n$$

$$\boxed{\text{Expected running time } O(n)}$$