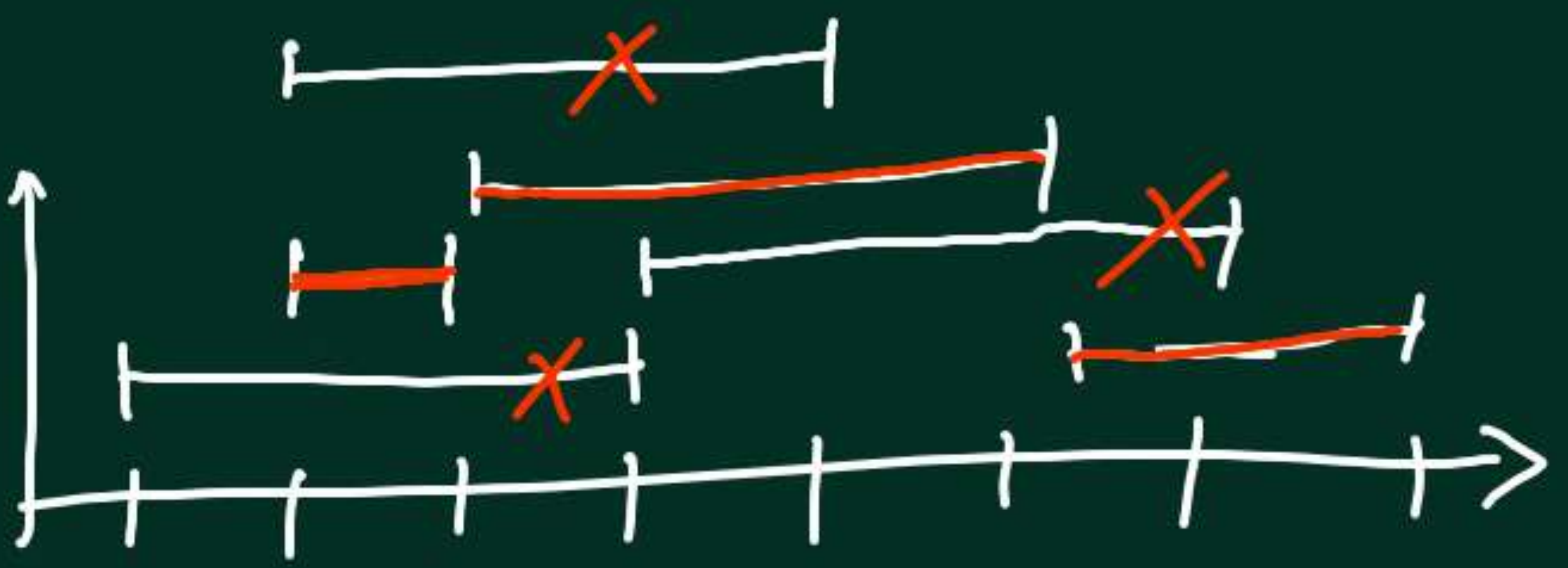
Interval Scheduling

Given a set of intervals, find the largest subset of disjoint intervals.



Greedy approach: min finish time

sort w.r.t. finish time

$\mathcal{I} \leftarrow$ input set

$I_0 \leftarrow$ min finish time $\mathcal{I}$

$\mathcal{I}' \leftarrow \mathcal{I} - \{$ intervals intersecting with $I_0 \}$

Output $I_0 +$ GreedyAlgo$(\mathcal{I}')$

Claim: There always exists an optimal solution that contains I_0 (earliest finish time)

Proof: Consider some optimal solution

$$I_1, I_2, \ldots, I_k$$

New solution $I_0, I_2, I_3 \ldots I_k$

This is a valid solution. k intervals.

Claim: Greedy algorithm gives an optimal solution.

Proof (induction on number of intervals in the input)

Base case: $n = 1$. Clearly optimal.

Iduction Hypothesis: for any set of $n-1$ intervals the greedy gives an optimal so

Opt($\mathcal{I}$

Induction step:

Greedy · optimal for n intervals.

$\tilde{\mathcal{I}}$ $\quad I_0 \leftarrow$ min finish time $\quad \mathcal{I}' \leftarrow \mathcal{I} - \{$ intersectin I_0

$|\mathcal{I}'| \leq n-1$. $\quad$ Output $I_0 + \boxed{\text{Greedy}(\mathcal{I}')}$

Claim: $I_0 + Opt(\mathcal{I}')$ is an optimal solution for $\mathcal{I}$

Proof: There is an optimal solution for $\mathcal{I}$ which contains I_0.

Let this be $\overbrace{I_0, J_1, J_2, \dots J_\ell}^{\ell+1}$

Obs: $J_1, J_2, \dots J_\ell$ disjont from I_0

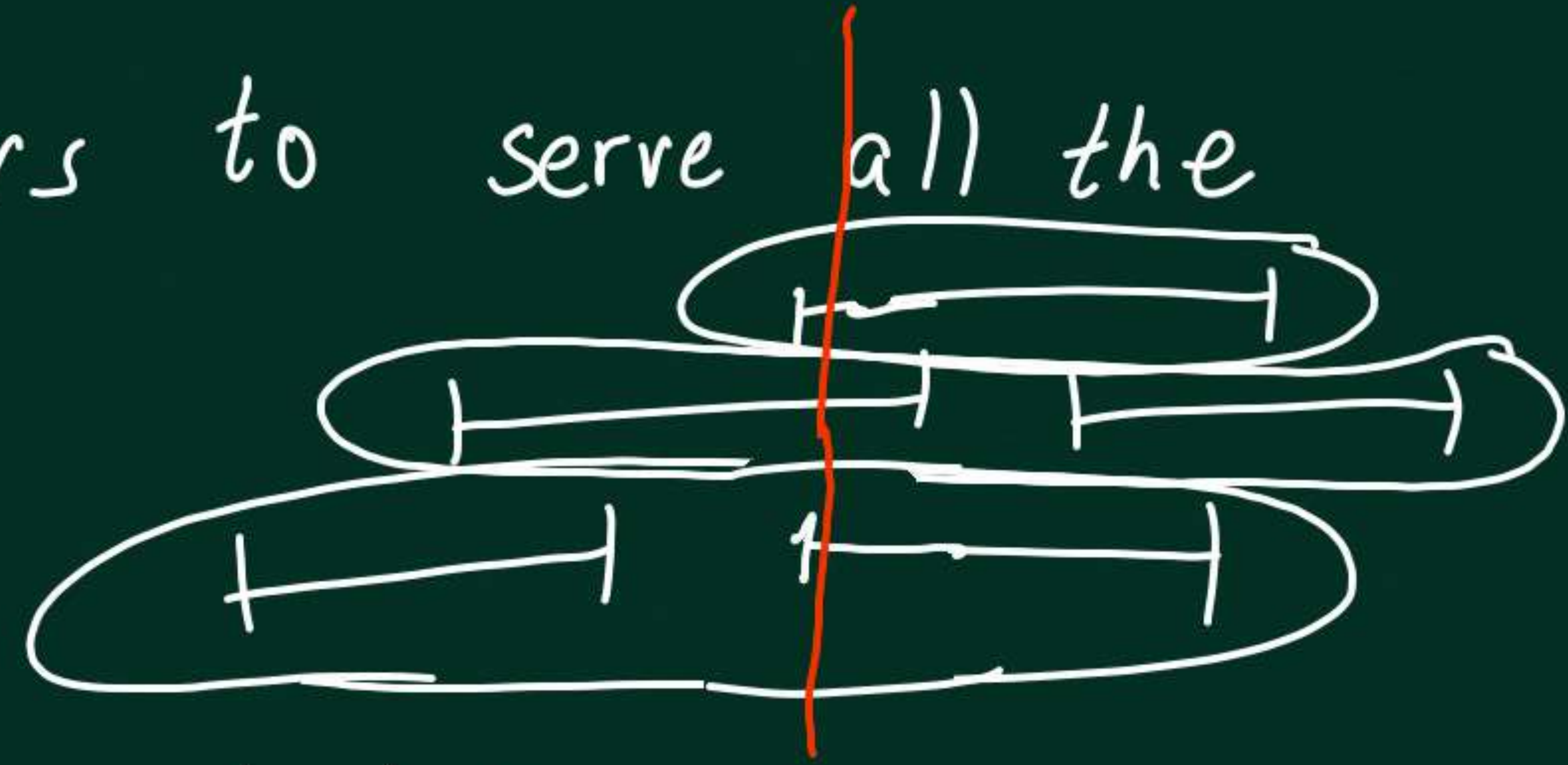
$\in \mathcal{I}'$

$J_1, J_2, \dots J_\ell \leftarrow$ valid solution for $\mathcal{I}'$

$$\ell \le |opt(\mathcal{I}')| \Rightarrow \ell+1 \le |I_0 + opt(\mathcal{I}')|$$

Problems

Minimum number of servers to serve all the requests (intervals)



Minimum number of platforms

= Maximum no. trains at a given time instant.

$A_1\ A_2\ A_3$

✓ A_1	A_2 ✗	A_3	← Max lateness 4
A_3	A_2	A_1	✓ 0

Minimize max lateness

n Assignments deadlines : $d_1, d_2, \ldots, d_n$

time : $t_1, t_2, \ldots, t_n$

Que: Is it possible to do all assignments within deadlines.

Que: lateness

$$l_i = \max(f_i - d_i, 0)$$

	A_1	A_2	A_3
time	3	2	1
deadln	6	4	2

Que: Maximize the number of assignments within deadline.

ess

Largest duration Interval scheduling

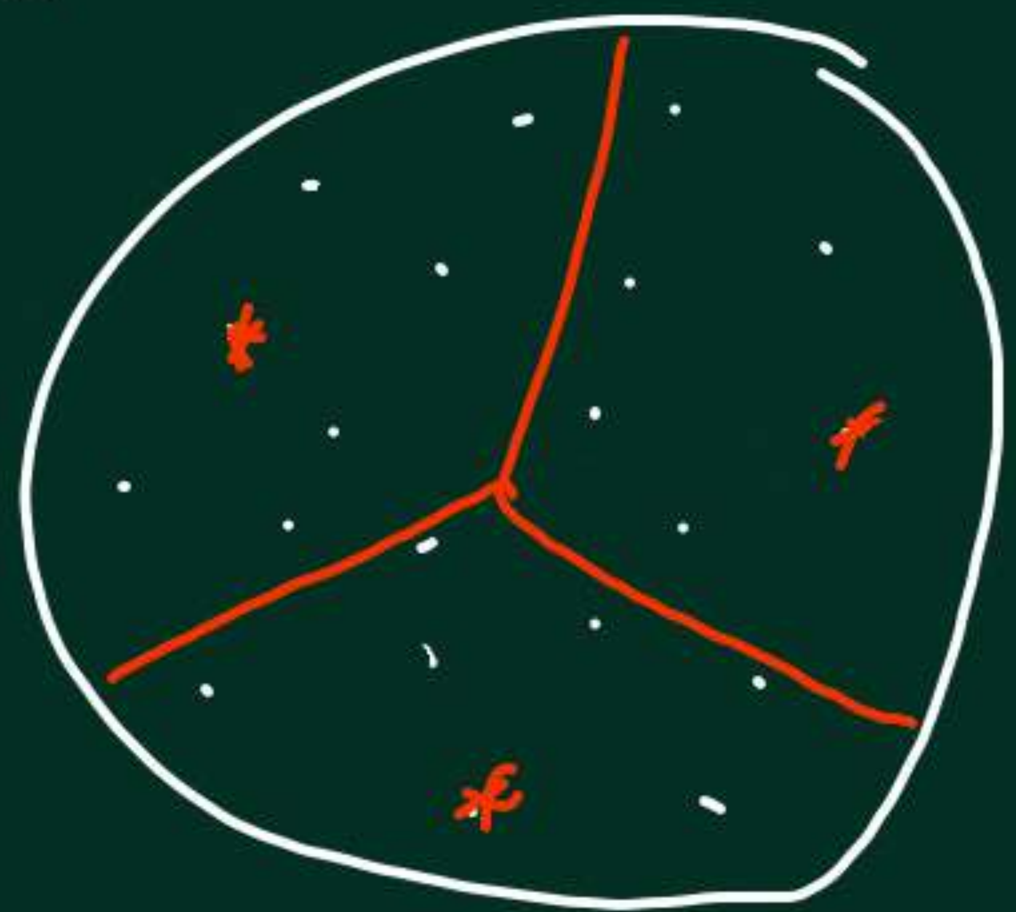
Given a set of intervals, find a set of disjoint intervals with maximum possible total length

Greedy approaches:

1. Max length first
2. minimize the intersecting length.
3. minimize the number of intersections.
4. earliest start time 5) earliest finish time

Dynamic Programming

- Categorize all possible solutions into various categories
- Find an optimal solution from each category using the same algorithm recursively on some other input instances.



- Compare these optimal solutions and take the best.
- Store the solutions for all the inputs you have already solved.

Longest duration interval scheduling

possible solutions ← subsets of disjoint intervals

I_1 × $(0,5)$ I_2 × $(4,8)$ I_3 × $(3,7)$ I_4 × $(2,6)$ I_5 ✓ $(5,9)$

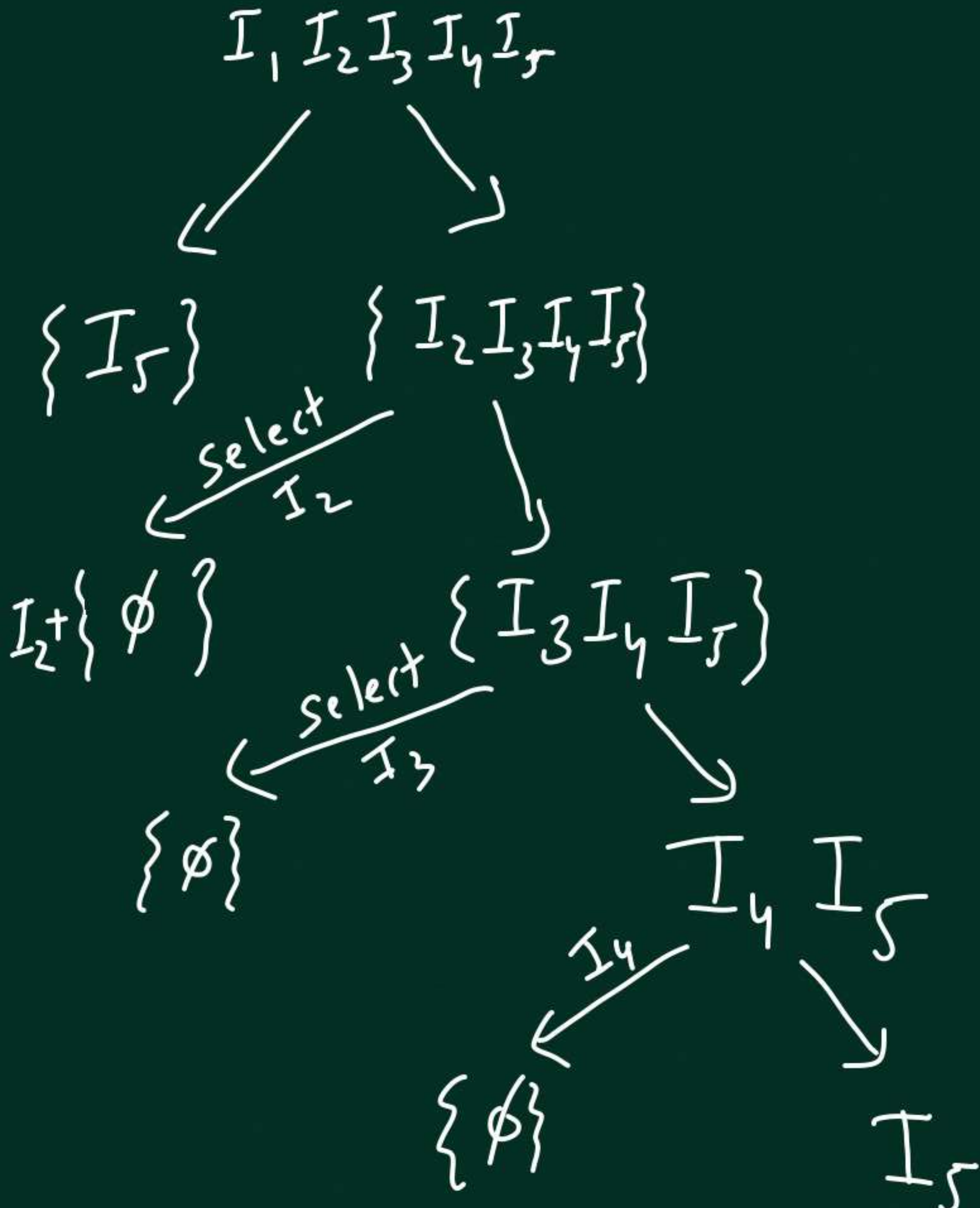
solutions which contain I_1

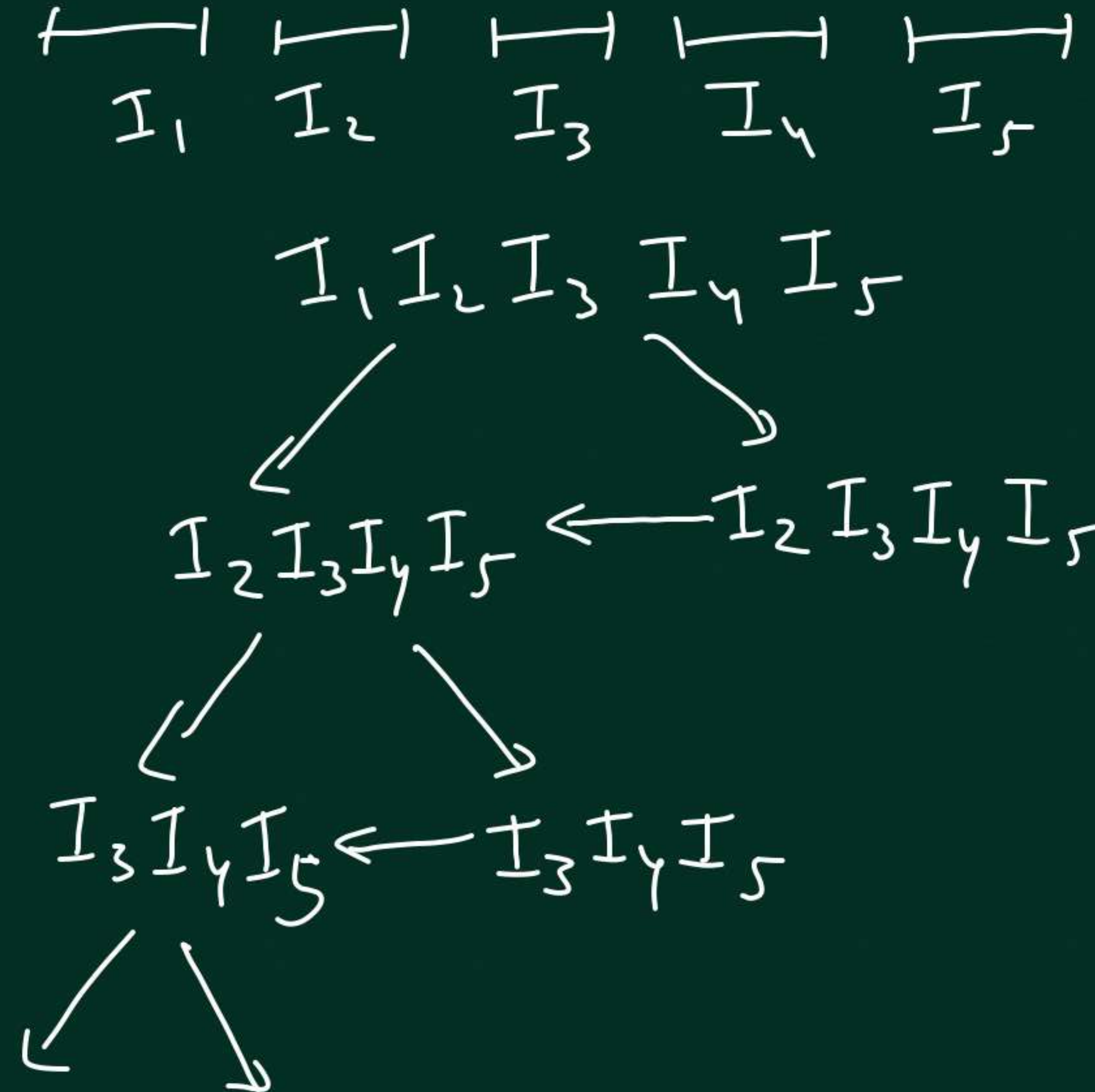
$ALG(I_5) + I_1$

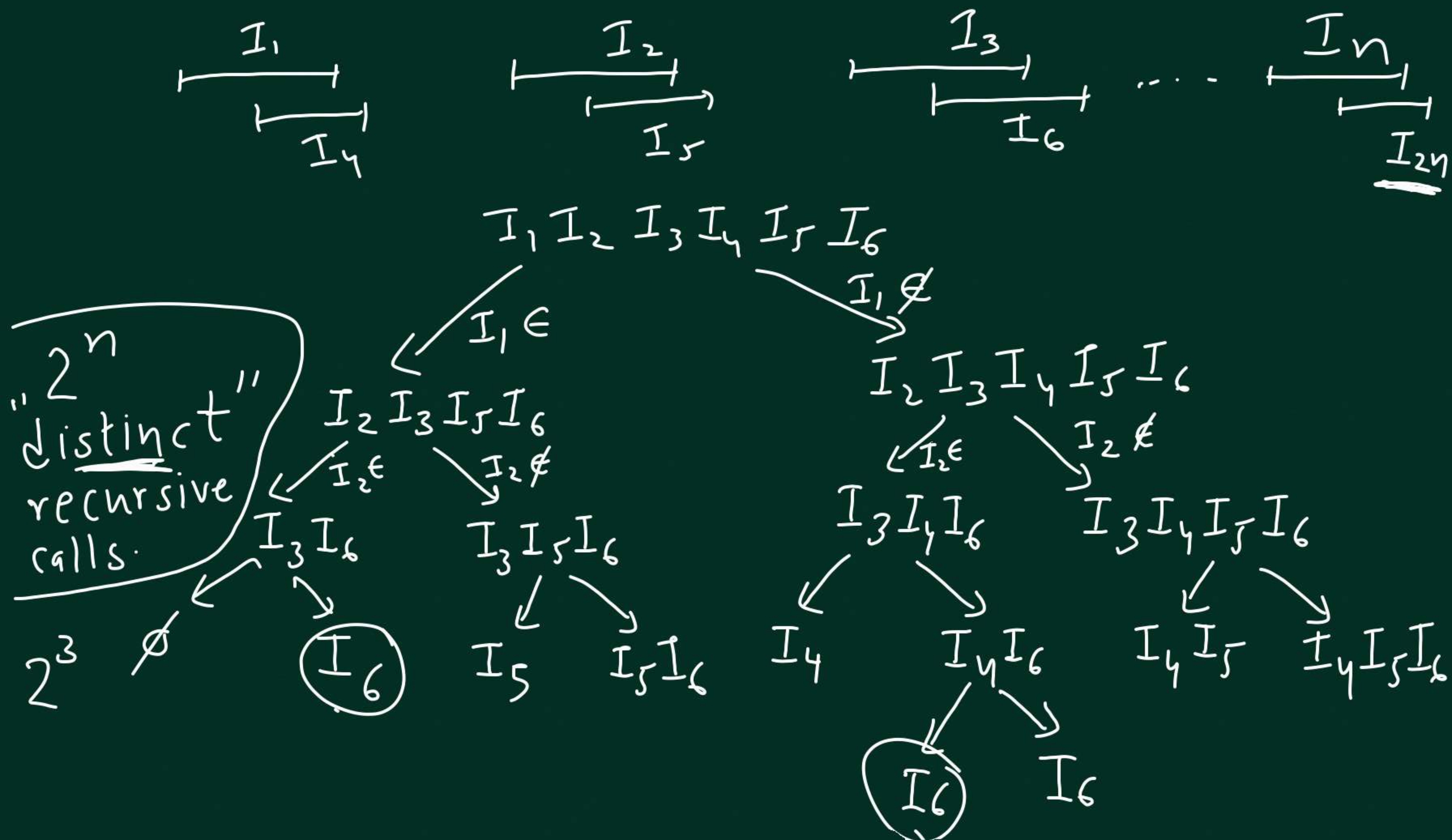
solution which don't contain I_1

$ALG(I_2, I_3, I_4, I_5)$

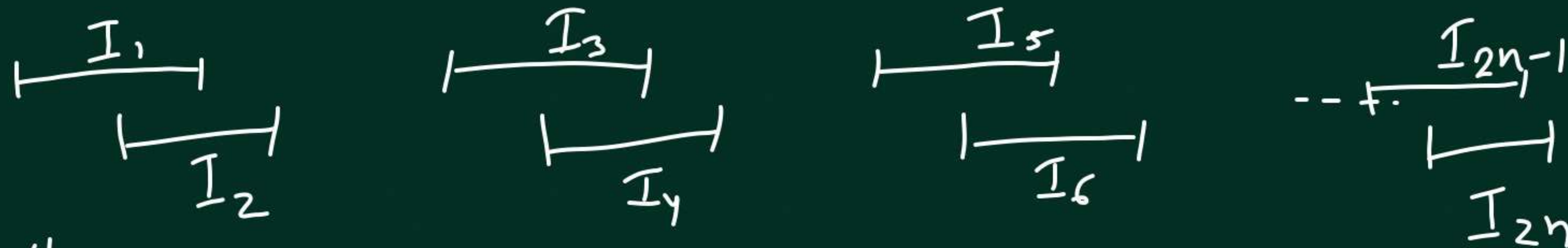
Best


I1 I2 I3 I4 I5
{ I5 }
{ I2 I3 I4 I5 }
select I2
I2 + { ∅ }
{ I3 I4 I5 }
select I3
{ ∅ }
I4 I5
I4
{ ∅ }
I5

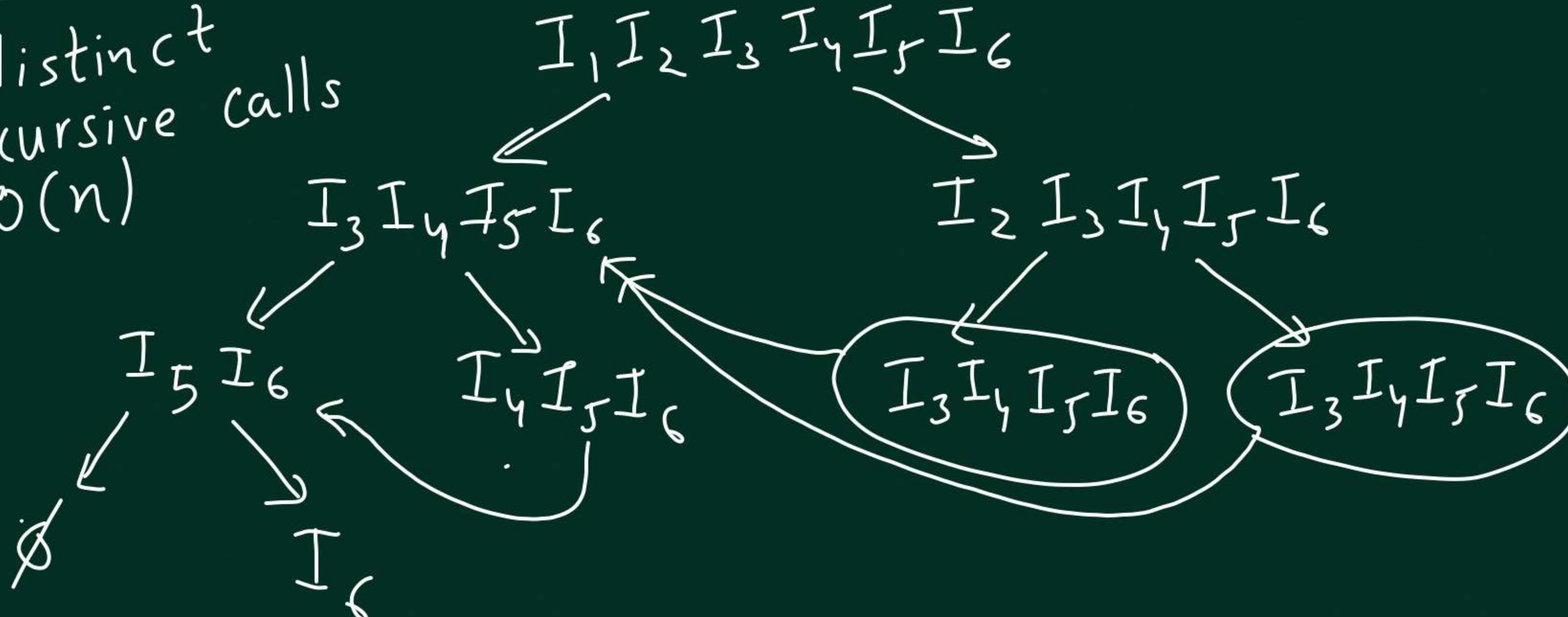

I1
I2
I3
I4
I5
I1 I2 I3 I4 I5
I2 I3 I4 I5
I2 I3 I4 I5
I3 I4 I5
I3 I4 I5



I_1 I_2 I_3 I_n
I_4 I_5 I_6 I_{2n}
$I_1 I_2 I_3 I_4 I_5 I_6$
$I_1 \in$
$I_1 \notin$
$I_2 I_3 I_5 I_6$
$I_2 I_3 I_4 I_5 I_6$
$I_2 \in$
$I_2 \notin$
$I_3 I_6$
$I_3 I_5 I_6$
$I_3 I_4 I_6$
$I_3 I_4 I_5 I_6$
$\emptyset$
I_6
I_5
$I_5 I_6$
I_4
$I_4 I_6$
$I_4 I_5$
$I_4 I_5 I_6$
I_6
I_6
"2^n "distinct" recursive calls.
2^3

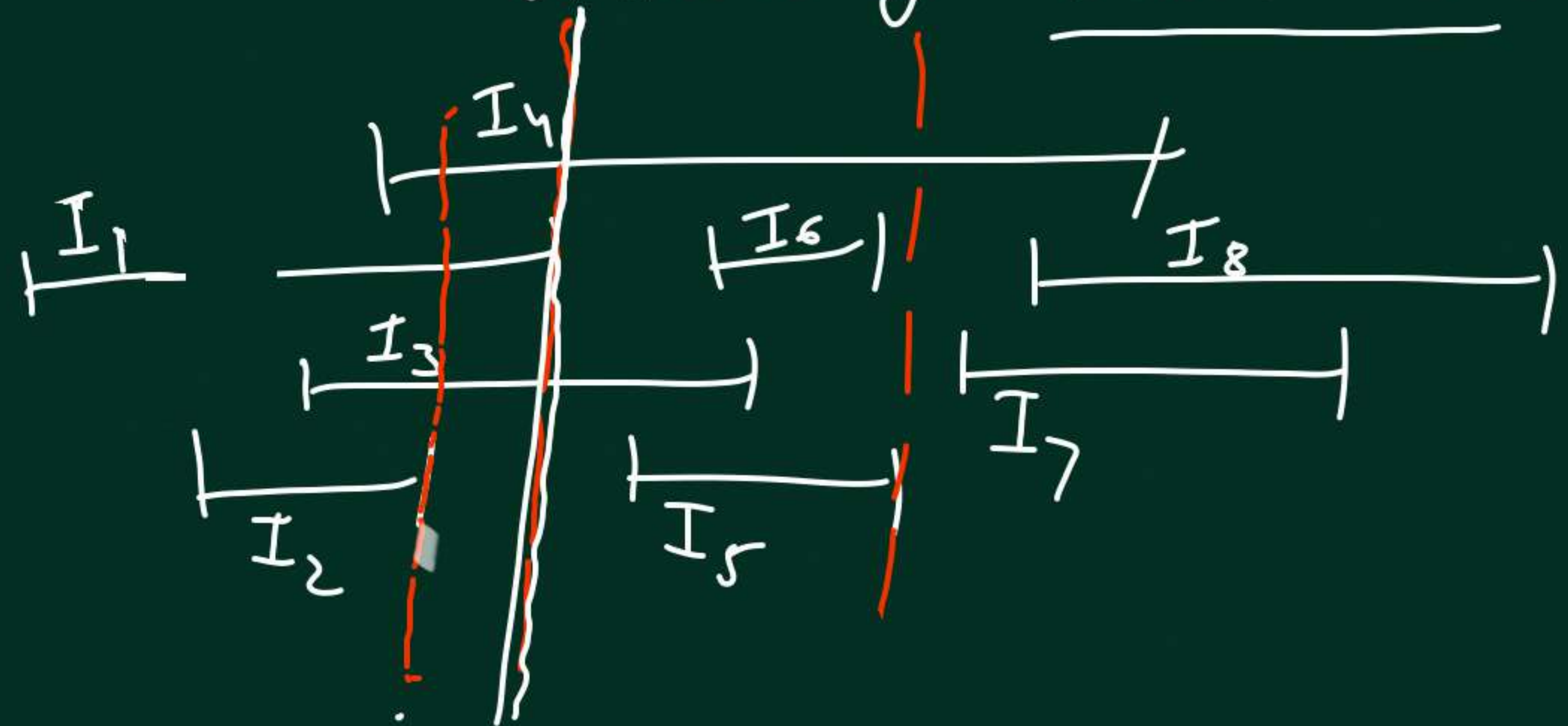


No. of "distinct" recursive calls $O(n)$



Order — increasing start time

increasing finish time (Bad examples)



At most n distinct recursive calls.

$I_1 \cdots I_n$

$I_j\, I_{j+1} \cdots I_n$ $\qquad$ $I_2 \cdots I_n$

$j \leftarrow$ first index s.t. $\text{start}(I_j) > \text{finish}(I_1)$

$\text{first}[k] \leftarrow$ first index j s.t. $\text{start}(I_j) > \text{finish}(I_k)$

$\text{Opt}[j]$

Optimal solution $(I_j, I_{j+1}, \dots, I_n)$

$\text{Opt}[1]$

For any k, $\{I_k \cdots I_n$

$$\text{Opt}[k] = \text{Max}\begin{cases} \text{Opt}[k+1] \\ |I_k| + \text{Opt}[\text{first}[k]] \end{cases}$$

Interval Scheduling.

① Max no. of intervals (Greedy)

② Longest total duration (DP)

$O(n \log n)$

③ Max total weight of selected Intervals (DP)

④ Count the number of disjoint subsets of intervals (DP)

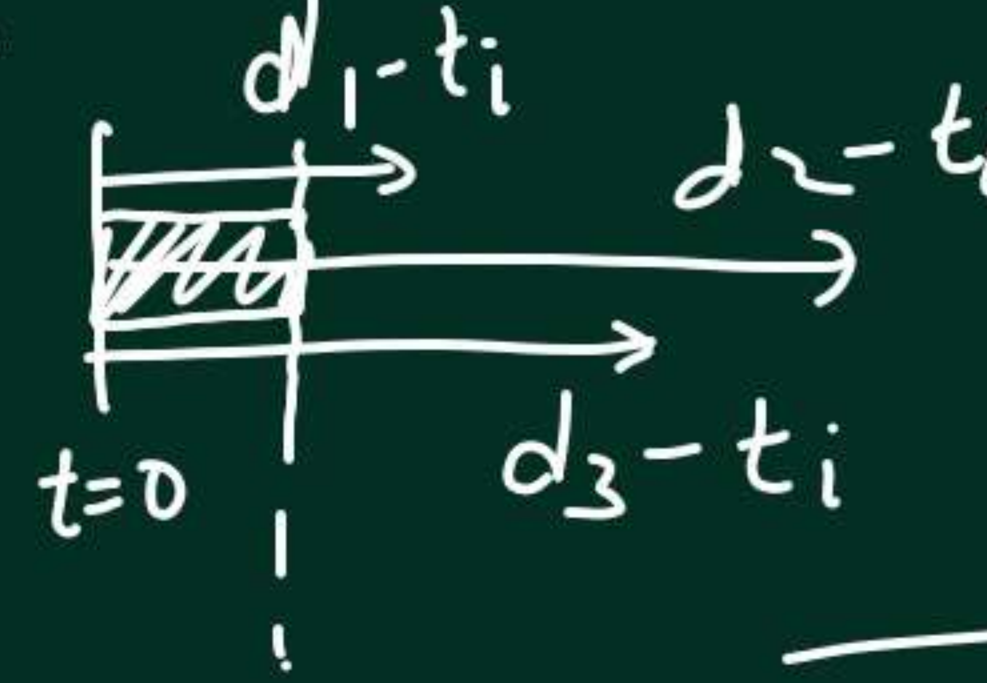
Maximum Lateness

Minimize max lateness

n assignments

deadlines $d_1, \dots, d_n$

time $t_1, t_2, \dots, t_n$



$$L_{max} = \max_i L_i$$

Lateness of Assign i

$$L_i = \max\{0, f_i - d_i\}$$

↑ finish A_i

DP

possible solutions

$n!$

n

Suppose A_i is scheduled at the first position.

$$\mathrm{Opt}\left(\{A_1, A_2, \dots, A_n\}\right)$$

$$\min_i \left\{ \max\left\{ \max\{t_i - d_i, 0\},\ \mathrm{Opt}\left(\{\overbrace{\underset{\underbrace{d_1 - t_i}_{D}}{A_1}, \underset{\underbrace{d_2 - t_i}_{D}}{A_2} \dots A_{i-1}, A_{i+1} \dots \underset{\underbrace{d_n - t_i}_{D}}{A_n}}\}\right) \right\} \right\}$$

No. of "distinct" recursive call

Greedy

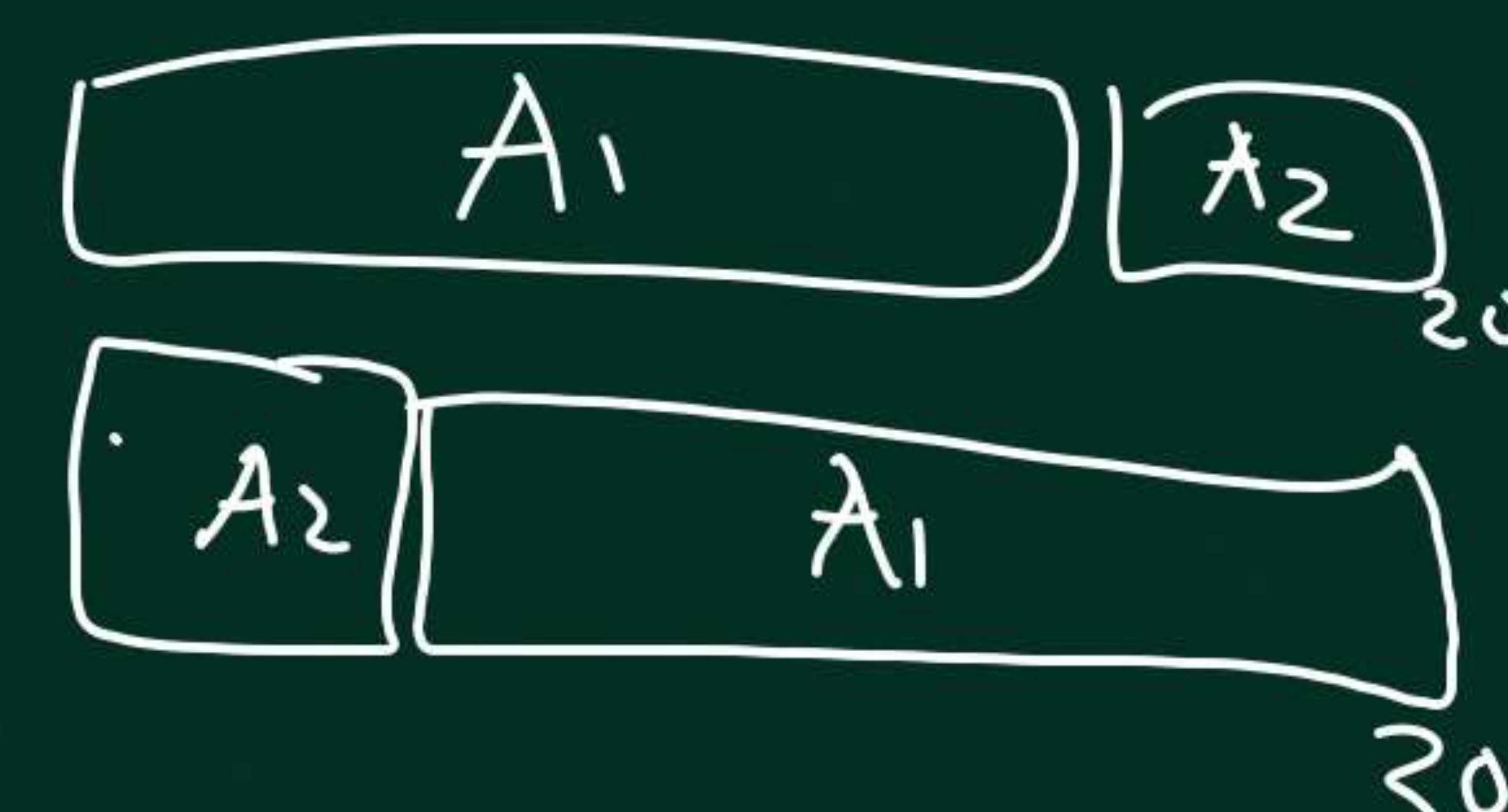
Schedule that assignment first which

1. minimizes d_i
2. ~~minimize t_i~~
3. ~~minimize $d_i - t_i$~~

	A_1	A_2
t	18	2
d	19	5
$d-t$	1	3

$L = 15$

$L = 1$ ✓



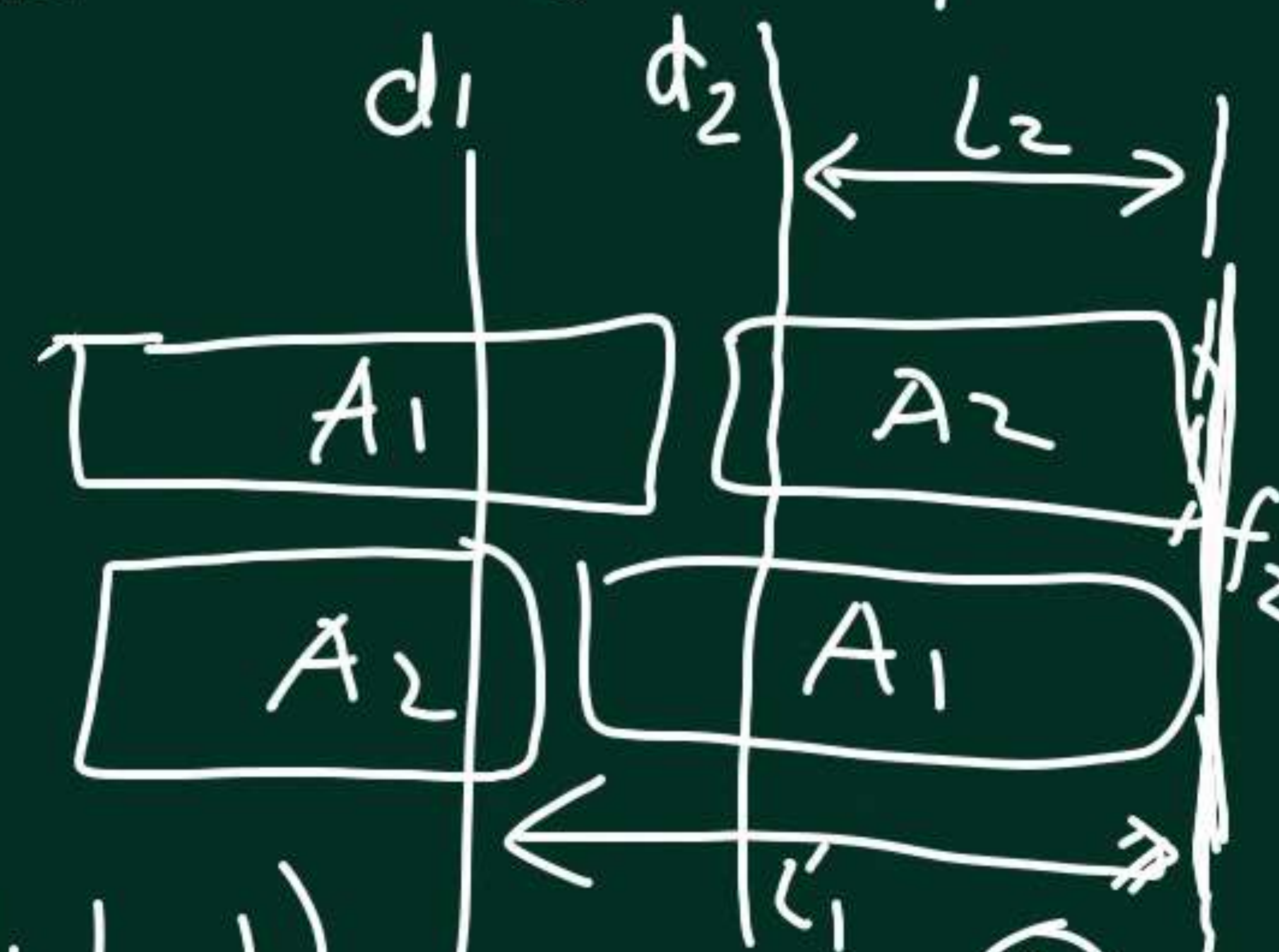
Schedule in the increasing order of deadlines.

Why is this optimal?

$$d_1 \leq d_2 \leq \cdots \leq d_n$$

Two Assignments

$d_1 \leq d_2$ $\rightarrow$



L_1 $\quad L_2$

L_1' $\quad L_2'$

Lateness(shedule 1)

$\leq$ Lateness(schedule 2).

① $L_1 \leq L_1'$

② $L_2 \leq L_1'$

$A_5 \; A_2 \; A_4 \; A_8 \; \boxed{A_1} \; A_{10} \; \ldots$

$\boxed{A_1} \; A_5 \; ---- \; A_8 \; A_{10} \; \ldots$

t_1

$A_5 \; A_2 \; A_4 \; A_8 \; A_1 \; A_{10} \; ---$

$A_5 \; A_2 \; A_4 \; A_1 \; A_8 \; A_{10}$

lateness improves.

Iterated Matrix Multiplication

$$[2\times 3][3\times 4][4\times 5] = 2\times 5$$

$$M_1 M_2 M_3 = (M_1 M_2) M_3 = M_1 (M_2 M_3)$$

$$\underset{p\times q}{A}\ \underset{q\times r}{B} = \underset{p\times r}{C}$$

$$\left.\begin{array}{l} pr \times q \text{ multi} \\ pr\,(q-1) \text{ additions} \end{array}\right\} O(pqr)$$

$$M_1 \quad M_2 \quad M_3$$

$$2\times 3 \quad 3\times 4 \quad 4\times 5$$

$$(\overbrace{M_1 M_2}^{2\times 4}) \cdot M_3$$

$$24 + 40 = 64$$

Optimal order?

$$10\times 3 \quad 3\times 4 \quad 4\times 5$$

$$120 + 200 \qquad 60 + 150$$

$$\overset{2\times 3}{M_1} (\overbrace{\overset{3\times 4}{M_2} \overset{4\times 5}{M_3}}^{3\times 5})$$

$$30 + 60 = 90$$

$$P_0 \quad P_1 \quad P_2 \quad P_3$$

$$P_1 < P_2$$

$$\underline{P_0 P_1 P_2} + P_0 P_2 P_3 = P_0 P_2 (P_1 + P_3)$$

$$\underline{P_1 P_2 P_3} + P_0 P_1 P_3 = \underline{P_1 P_3 (P_0 + P_2)}$$

$$M_1 \quad M_2 \quad M_3 \quad \cdots \quad M_n = M$$

$$P_0 \times P_1 \quad P_1 \times P_2 \quad P_2 \times P_3 \quad \cdots \quad P_{n-1} \times P_n \qquad P_0 \times P_n$$

Input $P_0, P_1, P_2, \ldots, P_n$

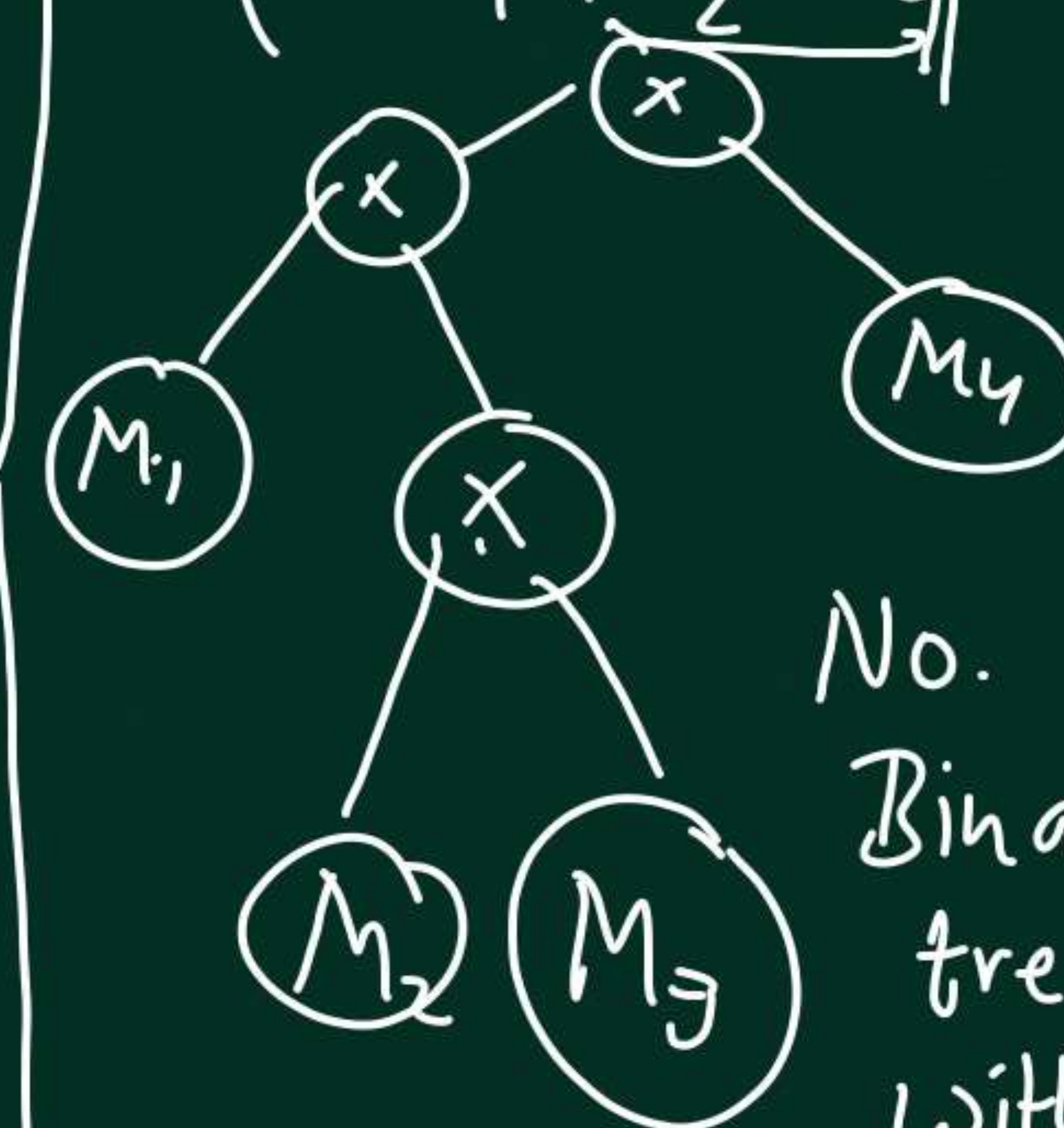
Output Best order

<u>DP</u> categorizing solutions

$$M_1 (M_2 M_3) M_4$$

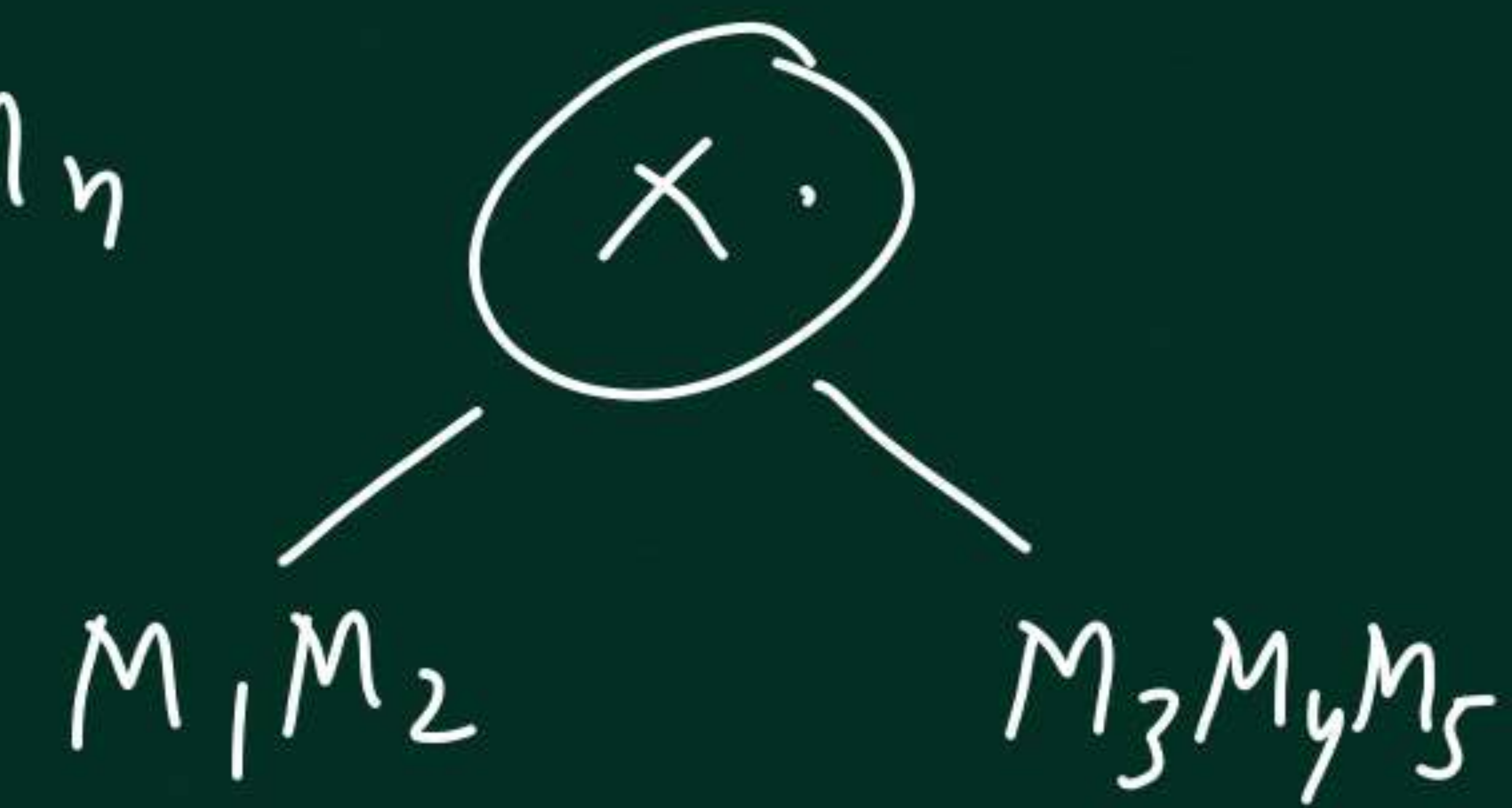
$$M_1 (M_2 M_3 M_4)$$

$$(M_1 (M_2 M_3)) M_4$$



No. of Binary trees with n leaves.

$$M_1\ M_2(M_3\ M_4)M_5 \cdots M_n$$



Classifying possible orders

① first multiplication

② last multiplication

$$M_1\ M_2\ M_{i-1}(M_i\ M_{i+1}) \cdots M_n \longrightarrow M_1\ M_2 \cdots (M_i M_{i+1}) \cdots M$$

$$P_0\ P_1\ P_2\ P_{i-2}\ P_{i-1}\ P_i\ P_{i+1} \qquad P_0\ P_1\ P_2\ P_{i-1}\ P_{i+1} \cdots P$$

$$Opt(P_0, P_1, \cdots P_n) = \min_i \left(P_{i-1} \cdot P_i \cdot P_{i+1} + Opt(P_0, P_1, \cdots P_{i-1} P_{i+1} P_n \right.$$

$$\mathrm{Opt}(P_0, P_1, \ldots, P_n)$$

$$= \min_{i} \left\{ P_{i-1} P_i P_{i+1} + \mathrm{Opt}(P_0, P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n) \right\}$$

No. of distinct recursive calls? ← $\exp(n)$

$$(n-1) \times (n-2) \times (n-3)$$

$$P_0, P_1, P_2, \cancel{P_3}, \cancel{P_4}, \ldots, P_n$$

Every subsequence is possible

$$P_0, P_1, P_2, \ldots$$

$$\downarrow$$

$$P_0 P_2 P_3$$

$$\downarrow$$

$$P_0 P_3$$

$$(M_1 \; M_2 \; \ldots \; M_i) \times (M_{i+1} \; \ldots \; M_n)$$

$$P_0 \; P_1 \qquad P_{i-1} \; P_i \; P_i \; P_{i+1} \; \ldots \; P_n$$

$$Opt(P_0, P_1, P_2 \ldots P_n)$$

No of Distinct recursive calls $\leq \binom{n+1}{2}$

$$P_3 = \min_i \left[P_0 P_i P_n + Opt(P_0, P_1, \ldots, P_i) + Opt(P_i, P_{i+1}, \ldots P_n) \right]$$

$P_0 P_1 P_3$

$P_0 \; P_1 \; P_2 \; P_3 \; P_4 \; P_5 \; \ldots \; P_n$

Every recursive call $\longrightarrow P_k P_{k+1} \ldots P_l$

$$\mathrm{Opt}(P_k \cdots P_\ell)$$

$$= \min_i \left\{ P_k P_i P_k + \begin{array}{l} \mathrm{Opt}(P_k \cdots P_i) \\ \mathrm{Opt}(P_i \cdots P_\ell) \end{array} \right\}$$

$\underline{O(n^3)}$ time.

Implementation

Subset Sum problem.

$\{6, 2, -5, 4, -7, 15, -20, -10, 8, 9\}$

Is there a subset whose sum is zero?

Subset Sum problem. (Dynamic Programming)

$$\{\underbrace{6, 2, -5, 13, 4, -7, 15, -20, -10, 8}_{-9}, 9\}$$

Is there a subset whose sum is zero?

Trivial $\leftarrow 2^n$

Subset $\longrightarrow$ will have a_n (Is there a subset of $a_1 \ldots a_{n-1}$ with sum $-a_n$)

Subset $\longrightarrow$ will not have a_n

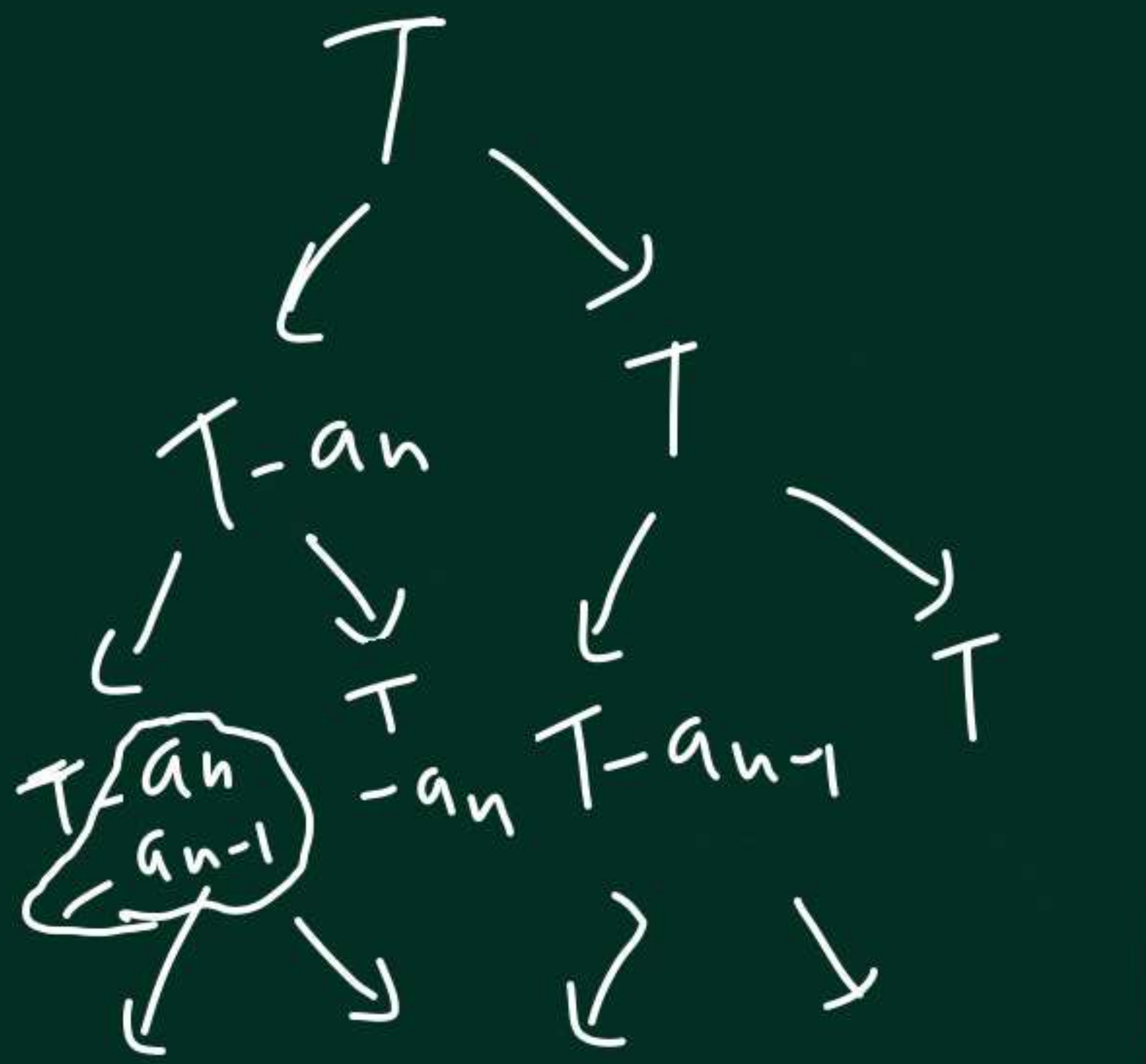
Is there a subset of $\underline{a_1 \dots a_n}$ with sum T

Is there a subset of $\underbrace{a_1 \dots}\, a_{n-1}$ with sum $T - a_n$

Is there a subset of $a_1 \dots a_{n-1}$ with sum T.

No. of distinct recursive calls?

prefix $\underline{n}$ $\leftarrow$ $(\underbrace{a_1, \dots, a_j}, \quad$ Target $t)$

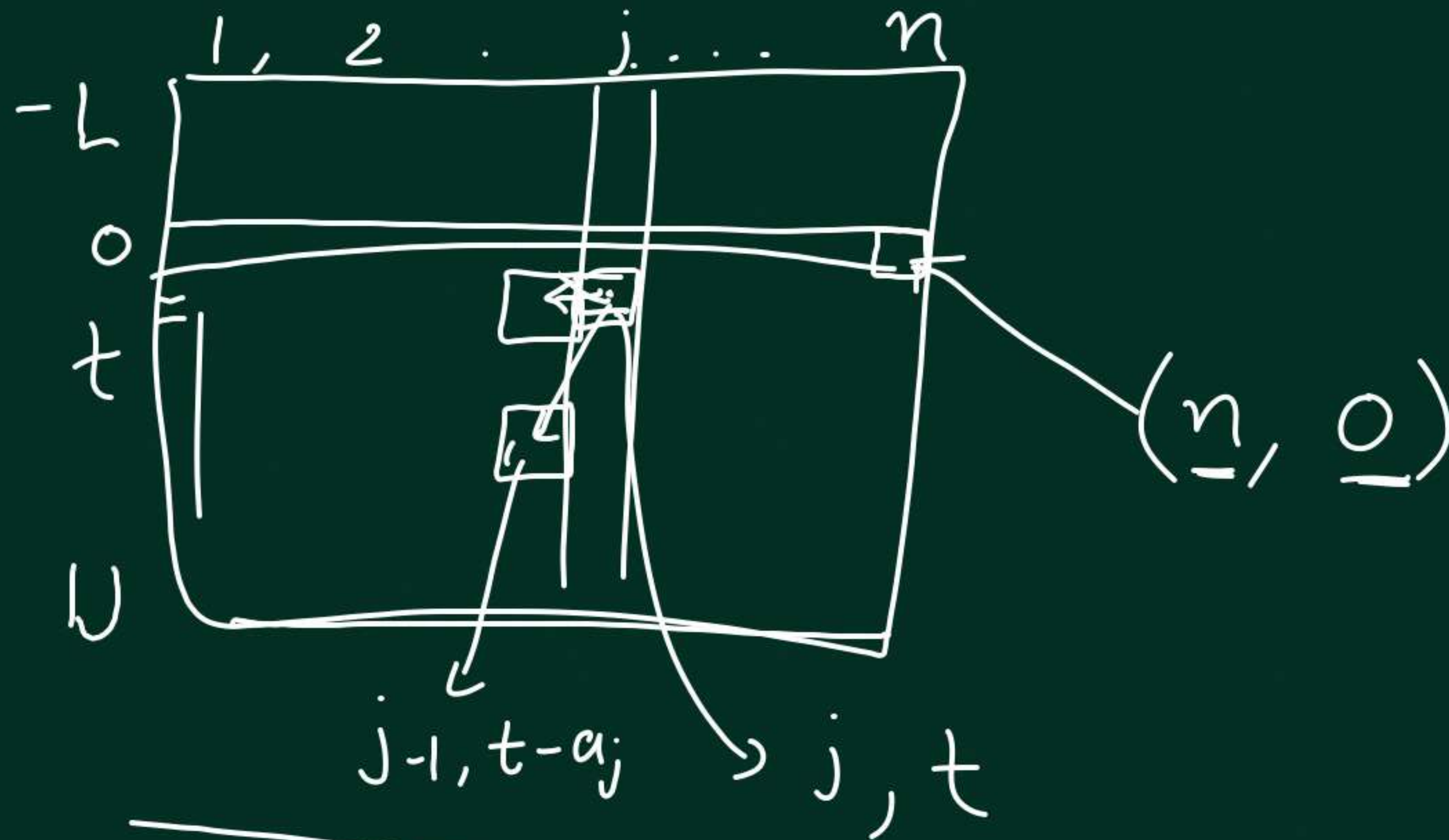


$$\underbrace{\sum_{i:\, a_i<0} \;\le\; T \;\le\; \sum_{i:\, a_i>0} a_i}$$

Size of this range $\sum_i |a_i| + 1$

$$\text{Subset-sum}(j,\ t) \leftarrow \begin{cases} \text{true} & \text{if there is subset of } a_1 \cdots a_j \text{ with sum } t \\ \text{false} & \text{othewise} \end{cases}$$

$$\text{subset-sum}[j, t] = \text{Subset-sum}[j-1, t] \text{ OR } \text{Subset}[j-1, t-a_j]$$



Implementation

n numbers

$\downarrow$

input size = total no. of bits

Complexity: $O(n \times \sum |a_i|) = O(\underbrace{n}\cdot n \cdot 2^{\ell})$

If $a_1 \cdots a_n \leftarrow \ell$ bit numbers, input size $n\ell$. pseudo-polynomial time.

Subset-sum $\leftarrow$ NP-hard.

unlikely to have a polynomial time algorithm.

Greedy doesn't work

Knapsack Problem

pseudopoly time.

$O(n \cdot W)$

Value $p_1, p_2, \ldots, p_n$

weights $w_1, w_2, \ldots, w_n$

$\text{poly}\left(n, \sum_i p_i\right)$

Pick the subset with max total value but total weight $\leq W$.

Fractional Knapsack.

$$x_1, x_2, \ldots, x_n \in [0,1]$$

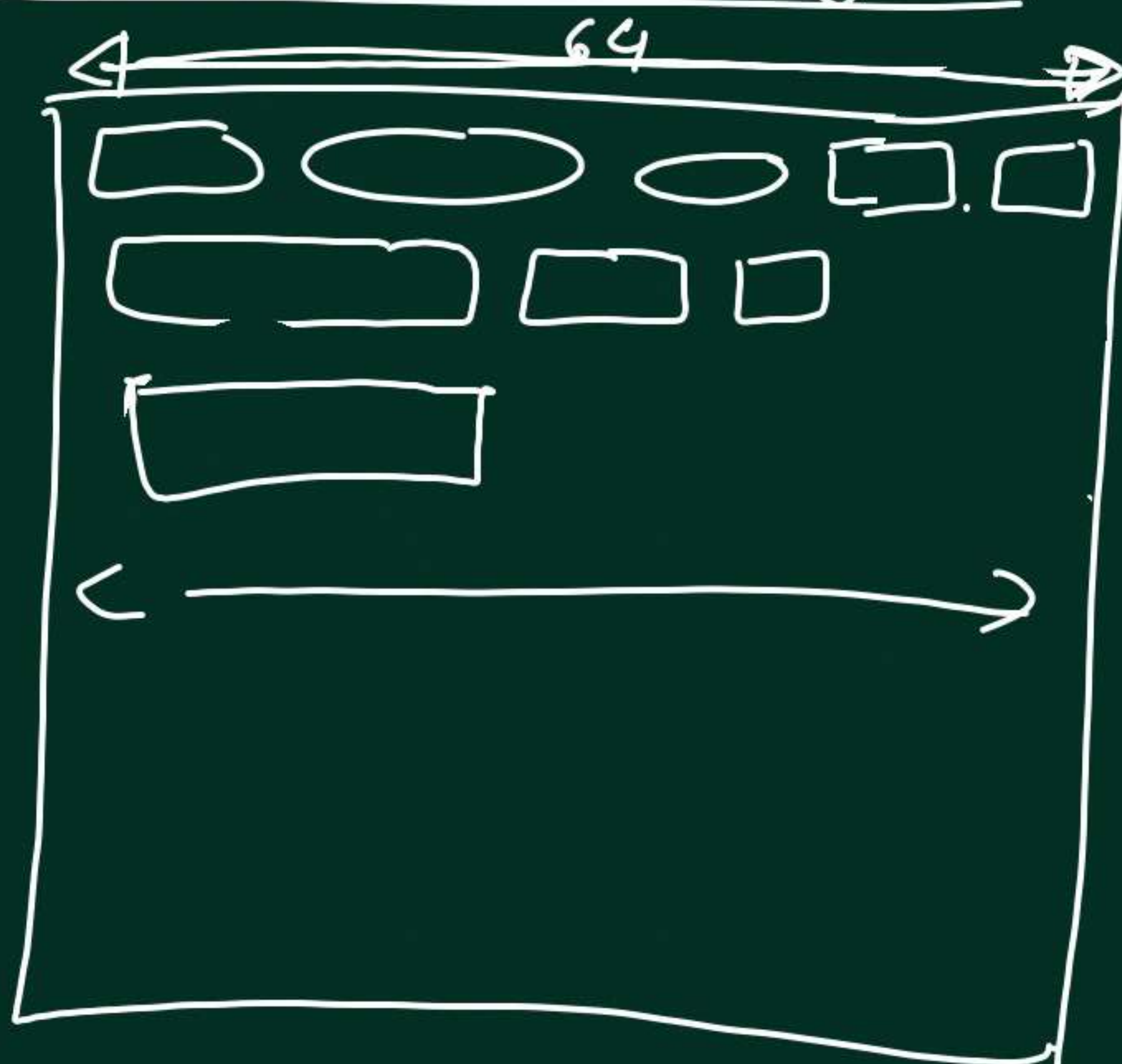
$$\max \sum_i p_i x_i$$

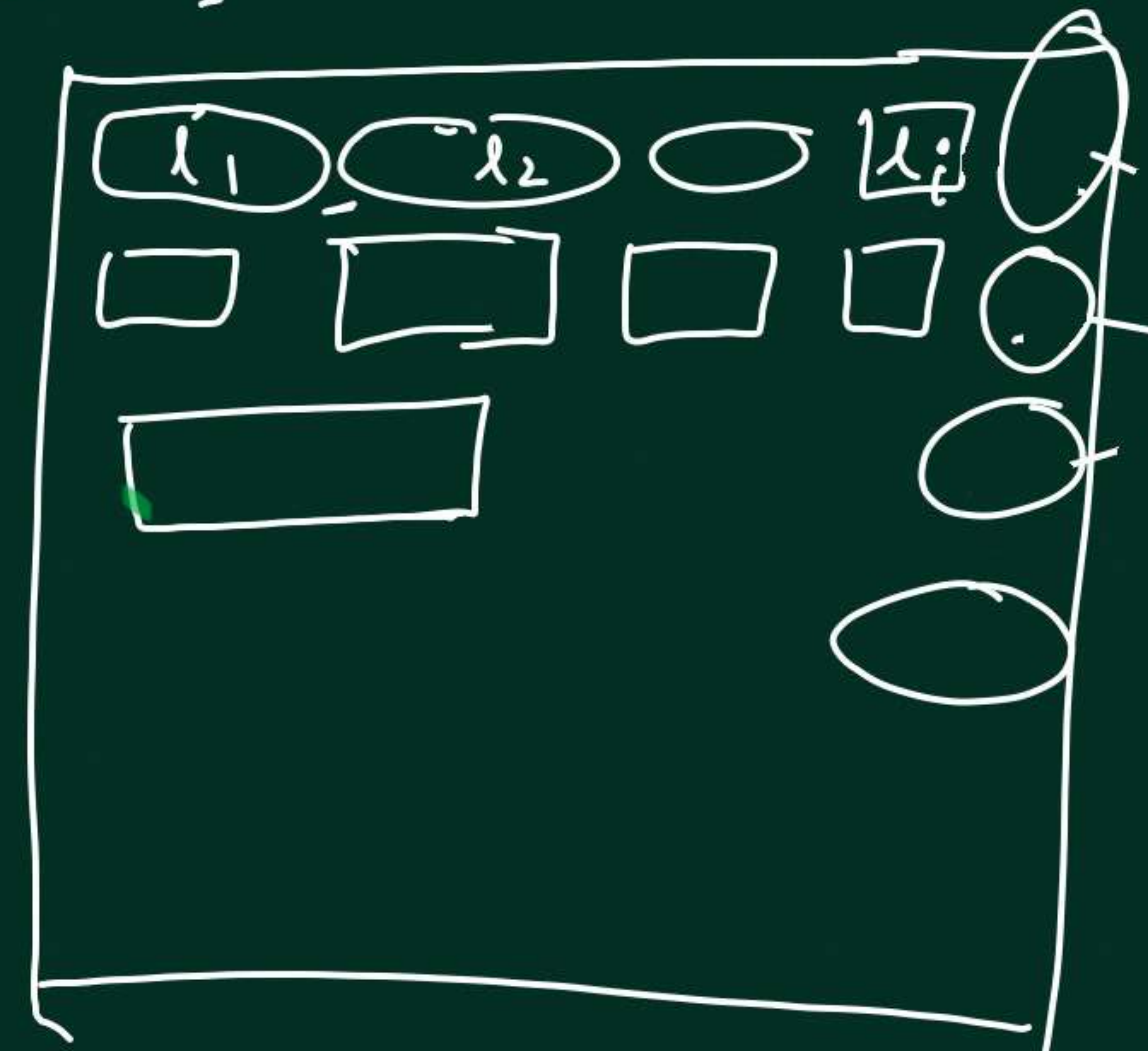
Subject to $\sum_i w_i x_i \leq W$

Greedy algorithm works.

Balanced Margins.

$\text{Slack}_1 = L - l_1 - 1 - l_2 - 1 \cdots - l_i$





Input $\leftarrow$ sequence of word $l_1, l_2, \ldots, l_n$ | Limit per line L.

$L \leftarrow$ linewidth

Input : $l_1, l_2, \ldots l_n$

Line k $\quad i^{th}$ to j^{th} word

$$slack_k = L - (l_i + 1 + l_{i+1} + 1 \; \cdots + l_{j-1} + 1 + l_j)$$

$\rightarrow$ roughly balanced

$$\sum_k slack_k = \text{const.}$$

$$\text{Minimize} \sum_{k=1}^{q} (slack_k)^2$$

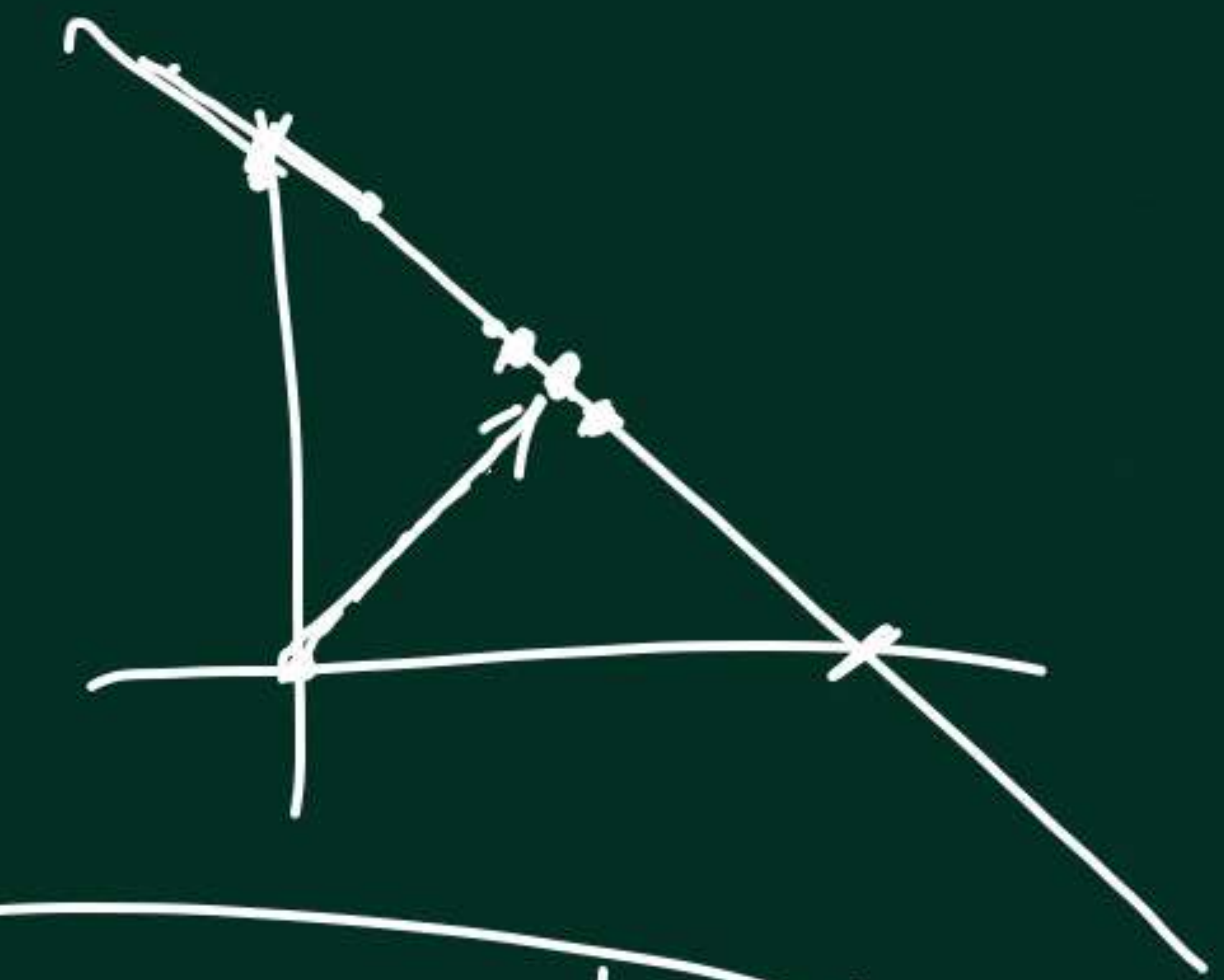
find $a, b, c \in \mathbb{Z}$

$$a + b + c = 14$$

minimize $\underline{a^2 + b^2 + c^2}$

$14 = 10 + 2 + 2$ 108

$14 = 4 + 5 + 5$ 66



No. of words in line 1 i_1

— 〃 — line 2 i_1+1, i_2

line 3 i_2+1, i_3

⋮

i_q

} exponentially many possible solutions.

Greedy Ideas.

→ As many as words as possible

→ Average slack
Greedily try to be
close to average slack.

does this minimize sum of $(\text{slack})^2$?

→ Check two lines at a time,
be greedy.

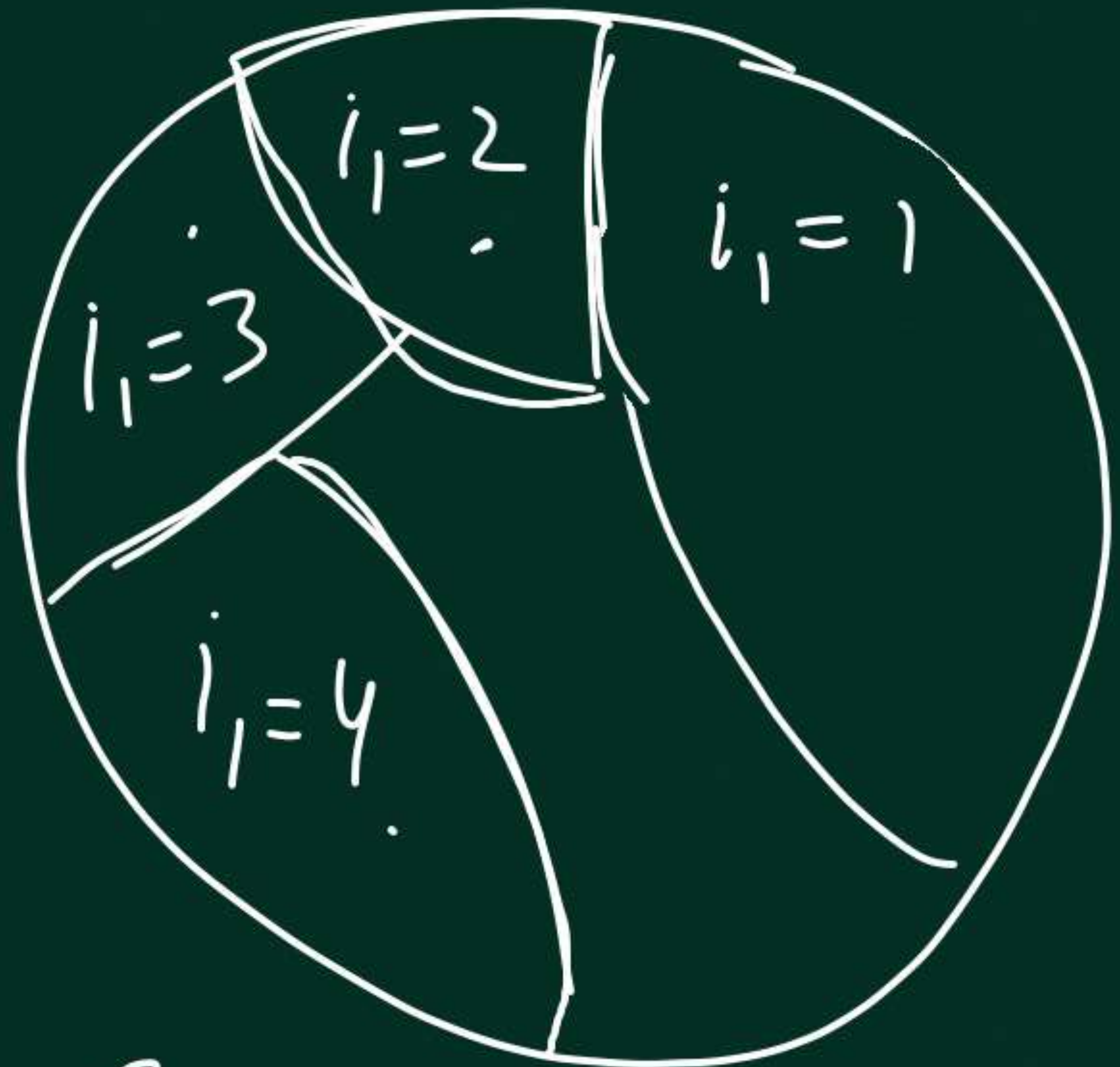
Dynamic Programming

Categorizing Possible solutions

$i_1 \leftarrow$ last word in firs tine

$i_2 \leftarrow$ last — Second

$i_3 \leftarrow$ last --- third.



$$OPT[1,n] = \text{Min}\begin{cases} (L-\ell_1)^2 + OPT[2,n] \\ (L-\ell_1-1-\ell_2)^2 + OPT[3,n] \\ (L-\ell_1-1-\ell_2-1\ldots\ell_h)^2 + OPT[h+1,n] \end{cases}$$

No of "distinct" recursive calls.

$\mathrm{Opt}(j) \leftarrow$ optimal value for the words $\ell_j, \dots, \ell_n$

$$
\begin{tikzcd}
 & \mathrm{Opt}(j) \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathrm{Opt}(j+1) \arrow[d] & \mathrm{Opt}(j+2) & \mathrm{opt}(n) \\
\mathrm{Opt}(j+2) & &
\end{tikzcd}
$$

for (j=n to 1)

$$\underline{Opt}(j) = \min \begin{cases} (L-l_j)^2 + Opt(j+1) \\ (L-\underline{l_j-1-l_{j+1}})^2 + opt(j+2) \\ \vdots \\ \underline{\qquad\qquad} \quad Opt(n) \end{cases}$$

Running time

$O(n^2)$

Implementation

Compute all slack squares

Optimal solution

Before hand $(L - l_j - 1 - l_{j+1} - 1 \cdots - l_k)^2$

Edit Distance / Sequence Alignment

Distance between strings.

Spell checker — Most similar, closest

Traning — Training

One way

4

S N O W	T R E E
N O W N	R A C E

Hamming Distance. = no. of positions where you have diff characters

Levenshtein Distance

= minimum no. of insertions, deletions or substitutions needed to go from one string to other

$$\text{SITTING} \xrightarrow{\text{sub}} \text{KITTING} \xrightarrow[\text{I}\to\text{E}]{\text{sub}} \text{KITTENG}$$

$$\text{KITTENG} \xrightarrow{\text{Del G}} \text{KITTEN}$$

SITTING
KITTEN

Distance 3

Deletion / Insertion $\leftarrow$ cost 2

Substitution

VOWEL – VOWEL $\leftarrow$ cost 1

Cons – Cons $\leftarrow$ cost 1

VOWEL – Cons $\leftarrow$ cost 3

MEAN

1

NEAN

2

NAN

1

NAM

2

NAME

MEAN $\xrightarrow{Del}$ MEA $\xrightarrow{Del}$ ME $\xrightarrow{Inse}$ AME $\xrightarrow{Ins}$ NAME

Cost 8

NAME

1

NEAN $\xrightarrow{1}$ NAAN $\xrightarrow{3}$ NAMN

$\xrightarrow{3}$ NAME

Cost 8

Sequence Alignment.

UGCTGACU

GAATGCA

Penalties ← Gap Penalty δ

← MisMatch Penalty

$\delta \quad \alpha_{CA} \quad \delta \quad \delta\,\delta$

U G C T G _ A C U

_ G A T G C A _ _

$\alpha_{CA} + 4\delta$

$x_1\ x_2 \ldots x_n$

$y_1\ y_2 \ldots y_m$

$$OPT(n,m) = \min \begin{cases} \delta + OPT(n-1,m) \\ \delta + OPT(n,m-1) \\ \alpha_{y_m x_n} + OPT(n-1,m-1) \end{cases}$$

DP

① $\underbrace{\qquad\qquad}\ x_n$

$\underbrace{\qquad\qquad}\ -$

② $\underbrace{\qquad\qquad}\ -$

$\underbrace{\qquad\qquad}\ y_m$

③ $\underbrace{\qquad\qquad}\ x_n$

$\underbrace{\qquad\qquad}\ y_m$

$OPT(n,m) \leftarrow$ optimal cost of Alignment between $x_1 \ldots x_n$ and $y_1 \ldots y_m$

$OPT(i,j) \leftarrow$ opt cost for alignment between $x_1 \dots x_i$, $y_1 \dots y_j$

$OPT(i,0) = i \cdot \delta$

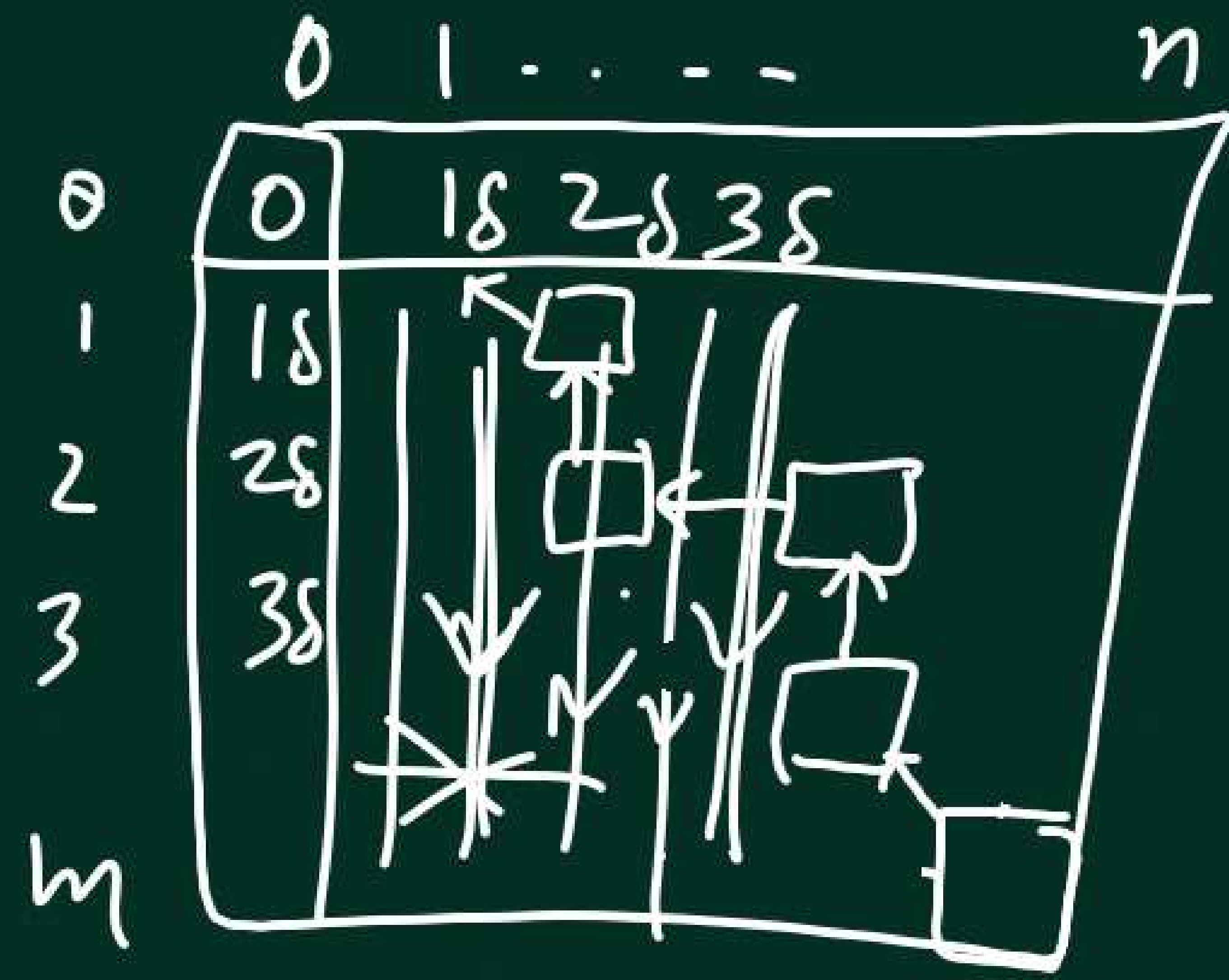
$OPT(0,j) = j \cdot \delta$

$OPT(n,m)$

for ($i=1$ to n)

for ($j=1$ to m)

$$OPT(i,j) = \min \begin{cases} \delta + OPT(i-1, j) \\ \delta + OPT(i, j-1) \\ \alpha_{x_i y_j} + OPT(i-1, j-1) \end{cases}$$



Optimal solution?

Opt cost $\leftarrow$ space $O(\min(m,n))$

Space Complexity $= O(mn)$

Space $O(m+n)$?

Time $O(mn)$.

Think about it