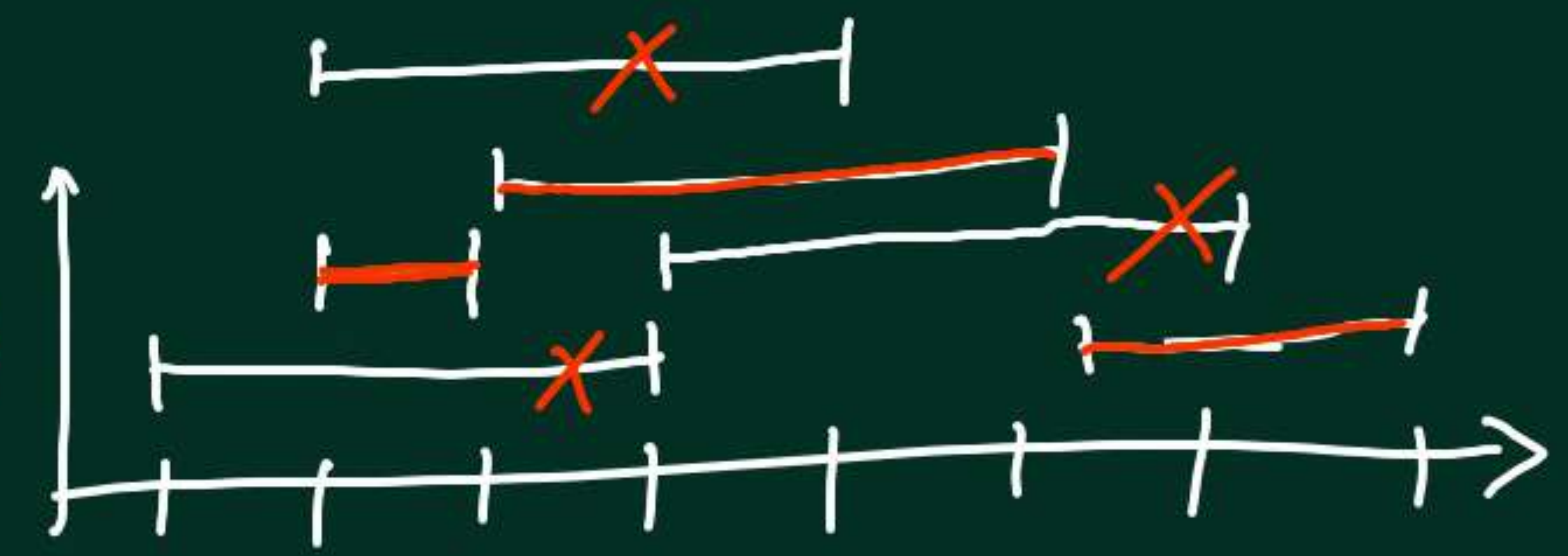
Interval Scheduling

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



Greedy approach : min finish time

$\mathcal{I} \leftarrow$ input set

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

sort w.r.t. finish time

$\mathcal{I}' \leftarrow \mathcal{I} - \{$ 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

Induction step:

Opt($\mathcal{I}$

Greedy · optimal for n intervals.

$\mathcal{I}$ $\quad I_0 \leftarrow$ 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 $\underbrace{I_0, J_1, J_2, \ldots J_\ell}_{\ell+1}$

Obs: $J_1, J_2, \ldots J_\ell$ disjont from I_0 $\in \mathcal{I}'$

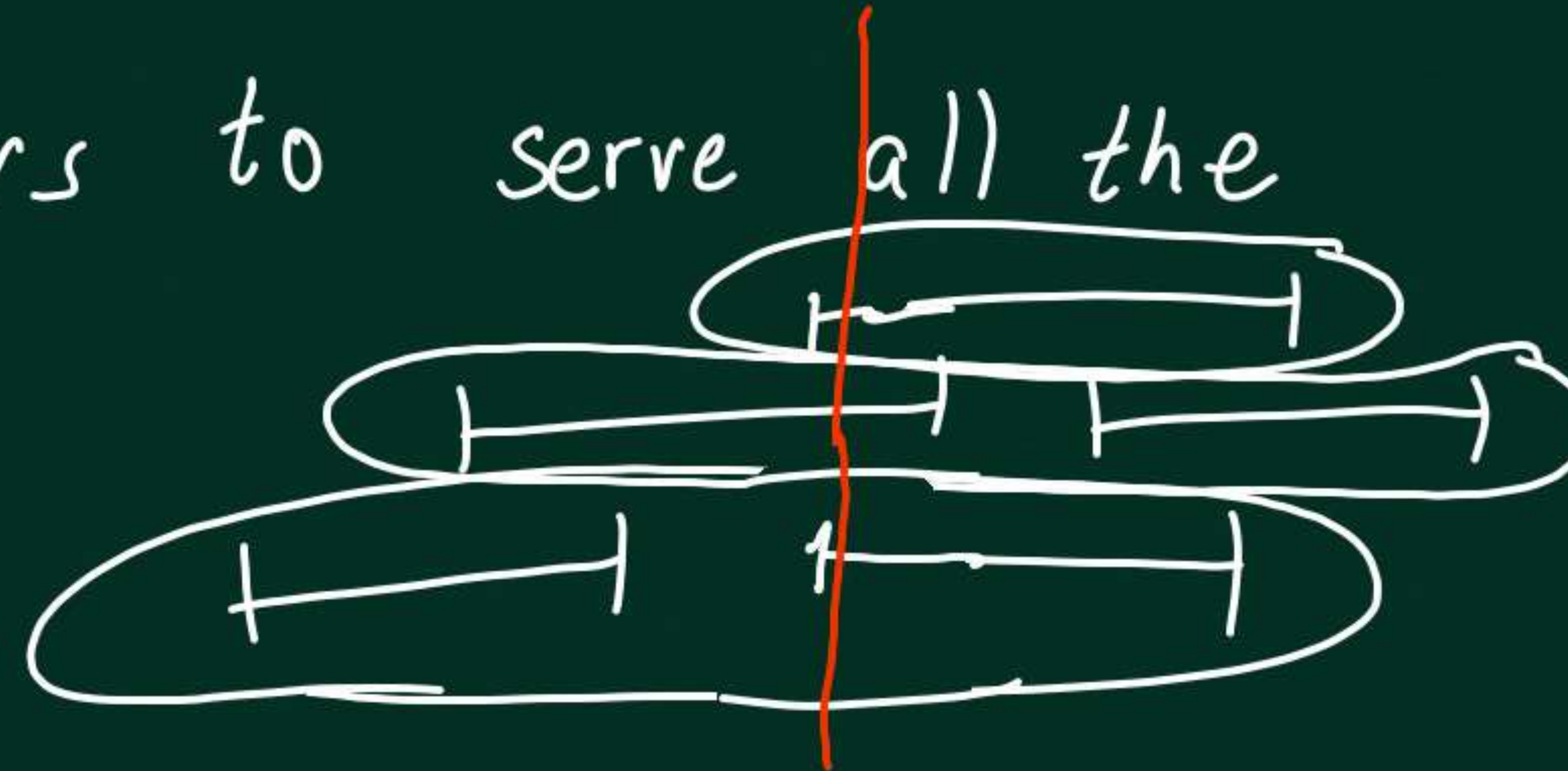
$J_1, J_2, \ldots J_\ell \leftarrow$ valid solution for $\mathcal{I}'$

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

1.

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
deadlin	6	4	2

Que: Maximize the number of assignments within deadline.

ess

Largest duration Interval scheduling

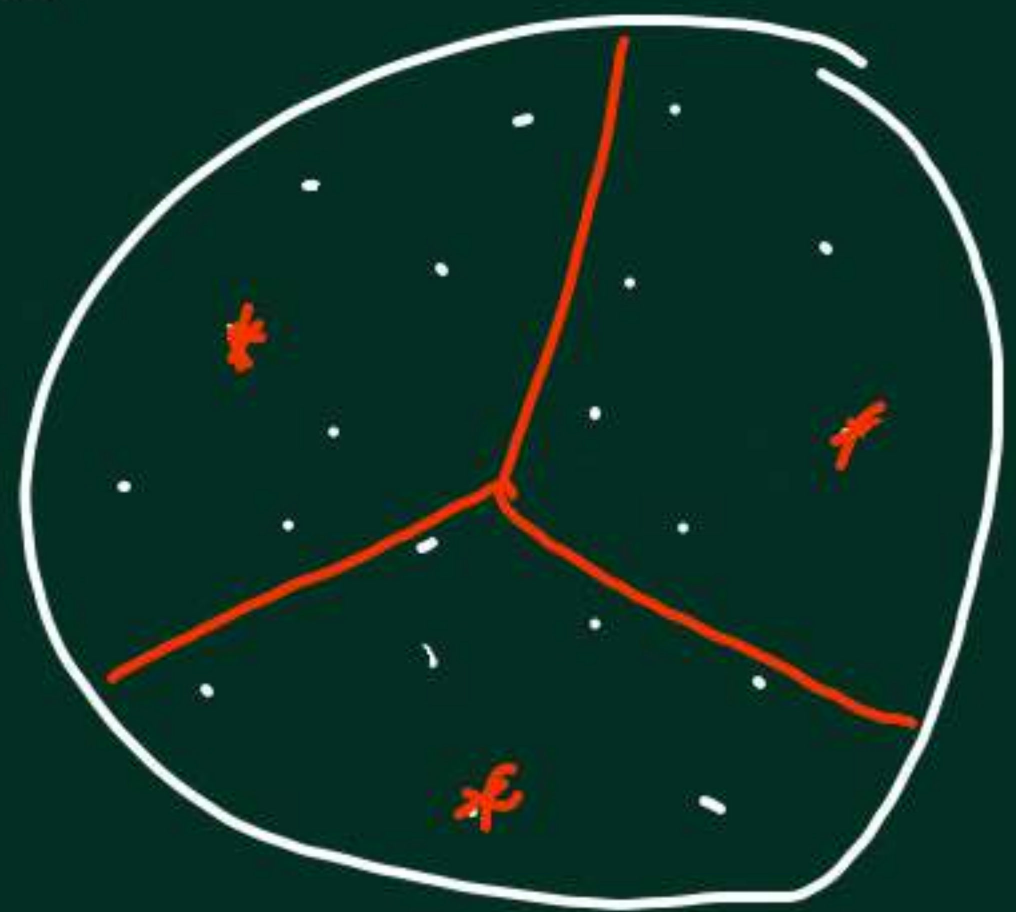
Given a set of intervals, find a set of disjoint intervals with maximum possible total length

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

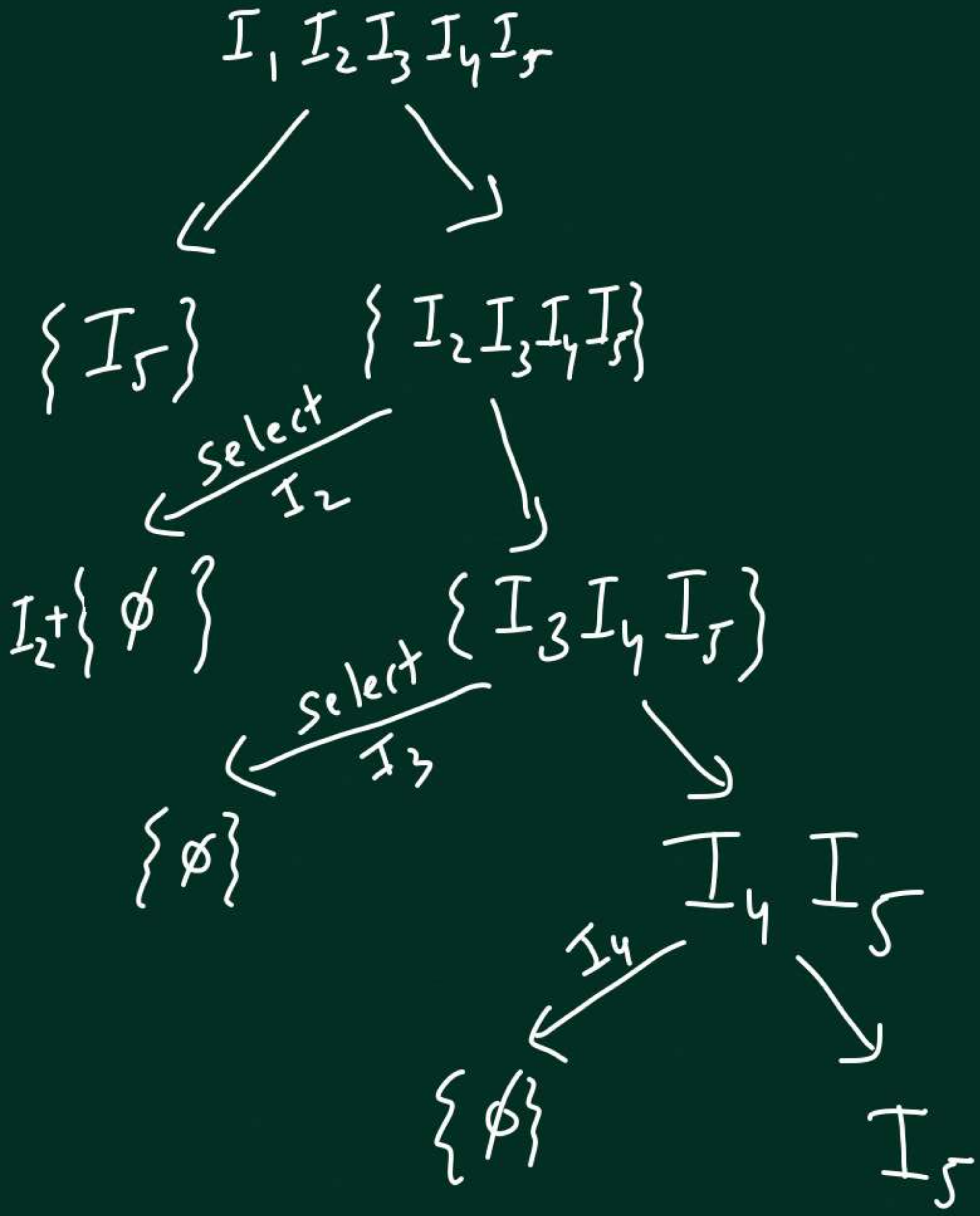
solutions which contain I_1

$ALG(I_5) + I_1$

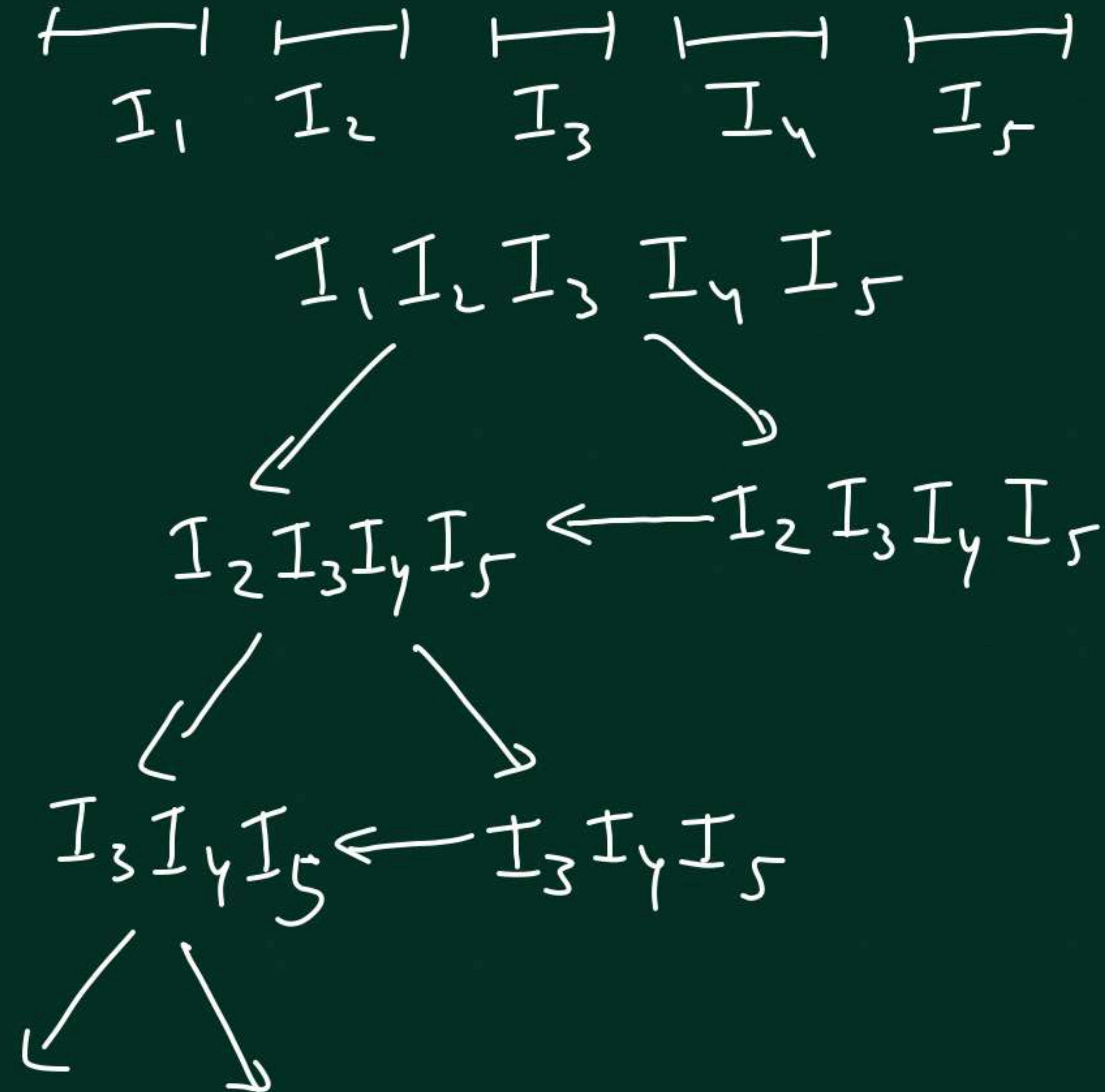
solution which don't contain I_1

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

Best



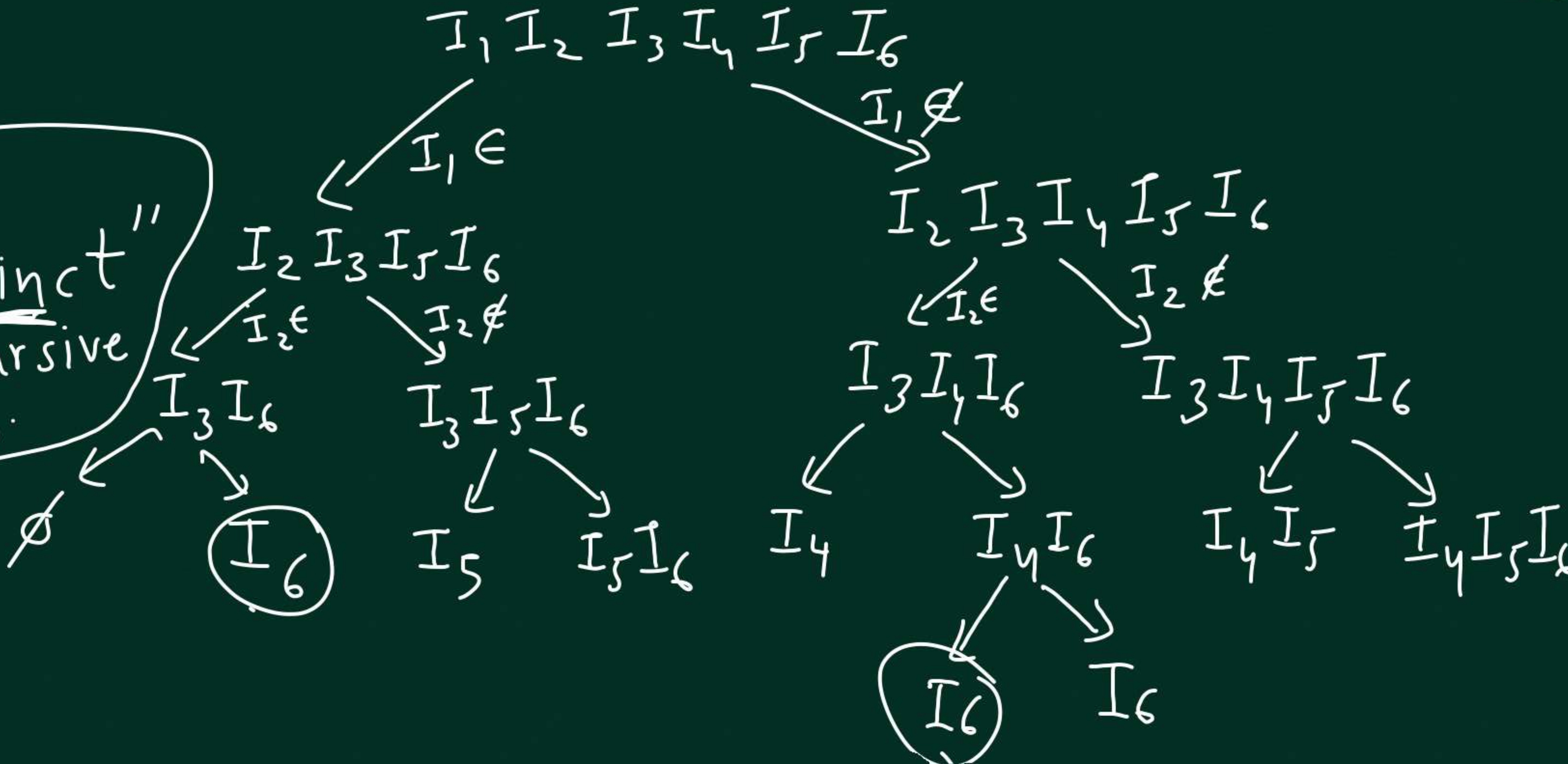
$I_1 I_2 I_3 I_4 I_5$
$\{I_5\}$
$\{I_2 I_3 I_4 I_5\}$
select I_2
$I_2 + \{\phi\}$
$\{I_3 I_4 I_5\}$
select I_3
$\{\phi\}$
$I_4 I_5$
I_4
$\{\phi\}$
I_5

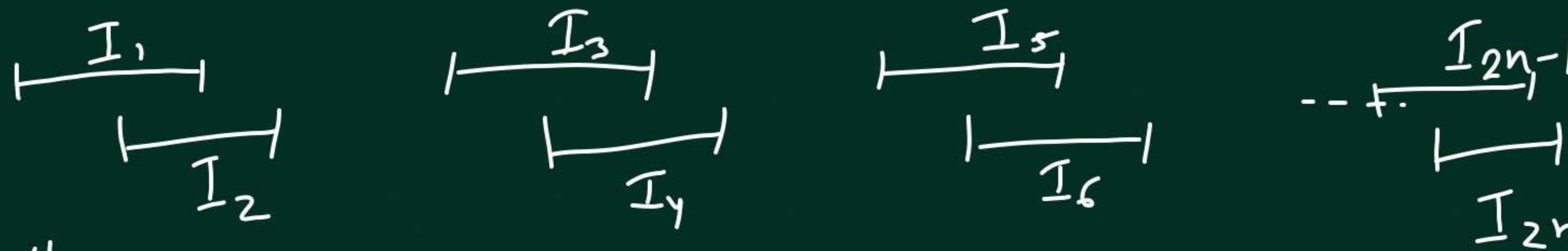


I_1 I_2 I_3 I_4 I_5
$I_1 I_2 I_3 I_4 I_5$
$I_2 I_3 I_4 I_5$
$I_2 I_3 I_4 I_5$
$I_3 I_4 I_5$
$I_3 I_4 I_5$

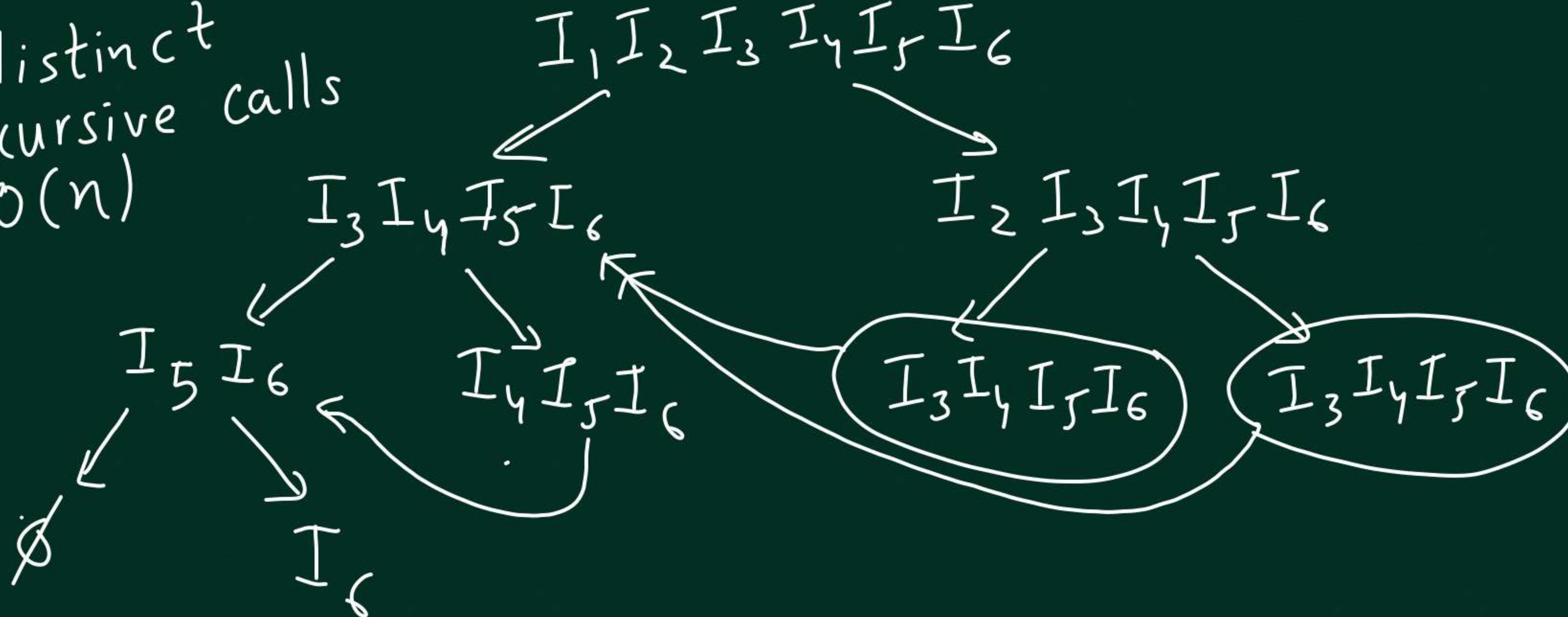
"2^n "distinct" recursive calls.

2^3

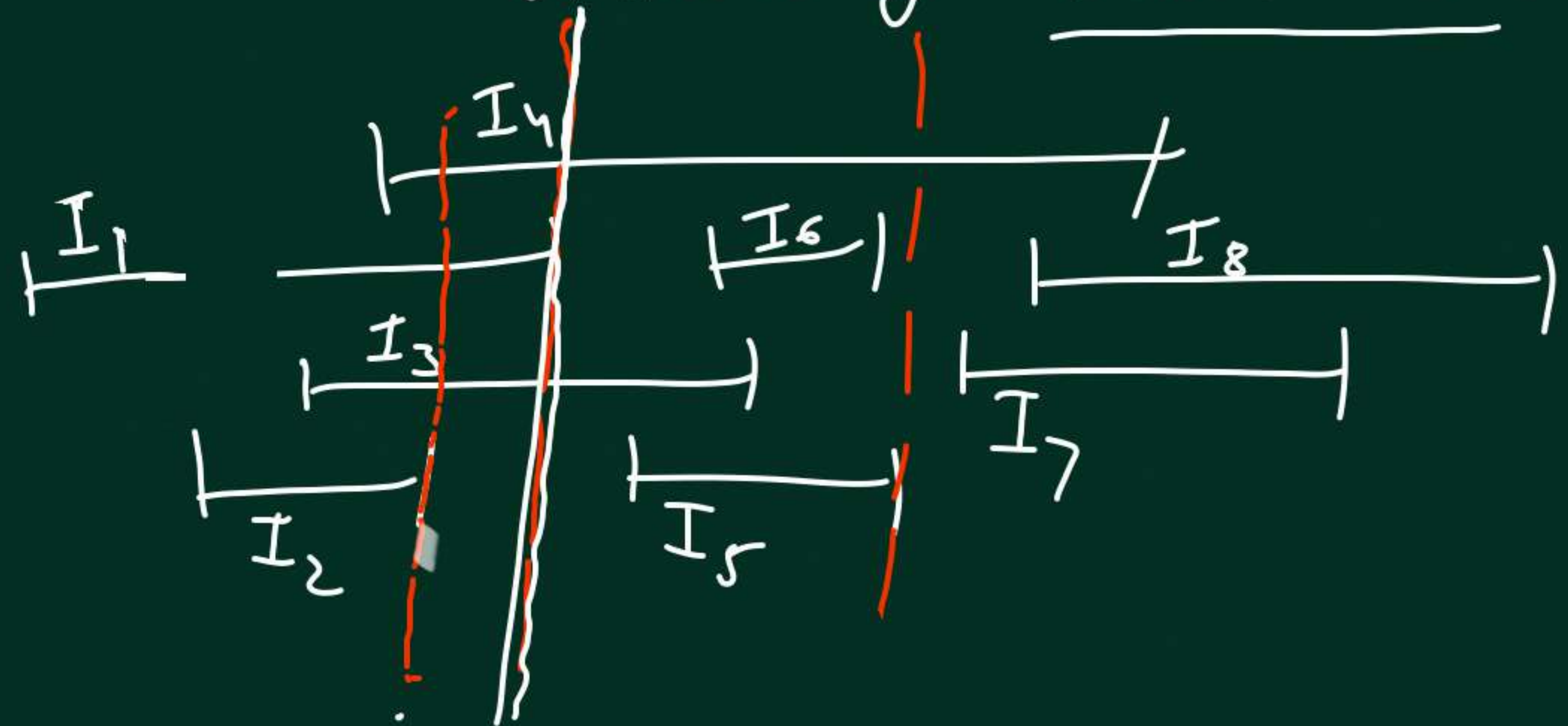




No. of "distinct" recursive calls $O(n)$



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)

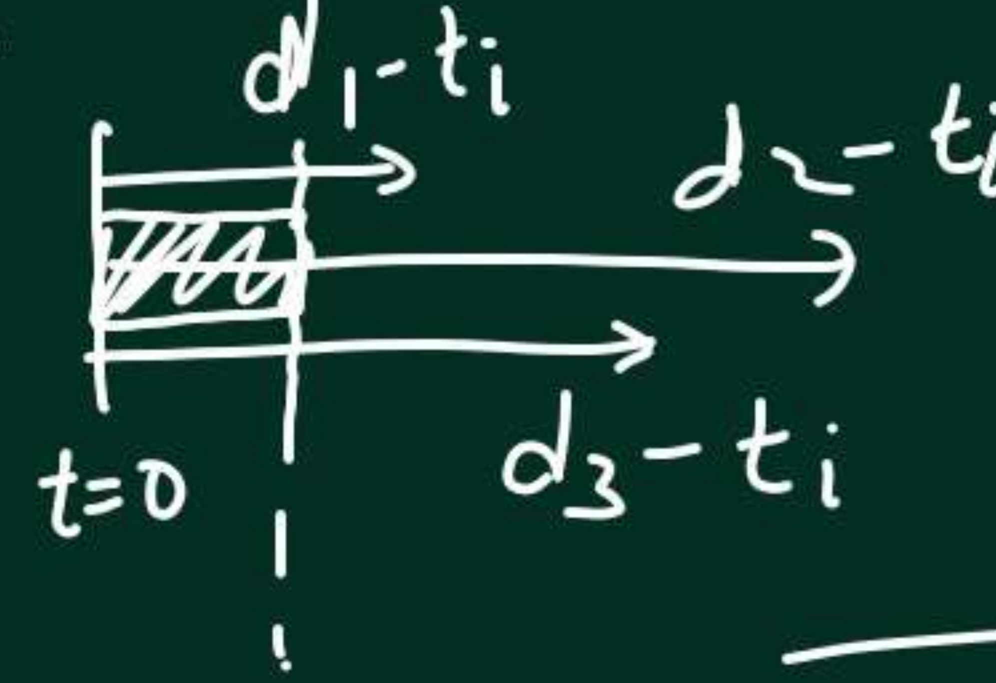
Maximum Lateness

, Minimize max lateness

n assignments

deadlines $d_1, \ldots, d_n$

time $t_1, t_2, \ldots, t_n$



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

Lateness of Assign i $\quad L_i = \max\{0, f_i - d_i\}$

$\uparrow$

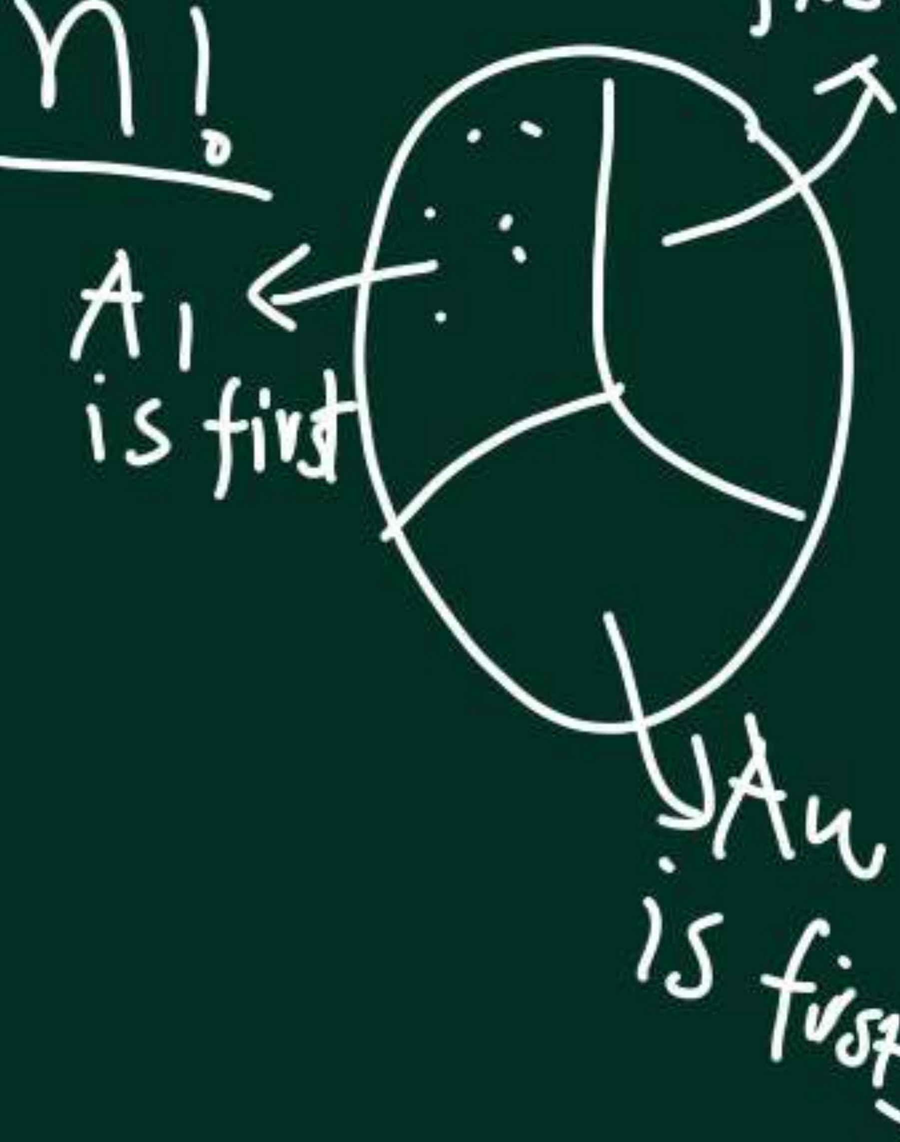
finish A_i

DP

possible solutions

$n!$

n



Suppose A_i is scheduled at the first position.

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

$$\min_i \left\{ \mathrm{Max}\left\{ \mathrm{Max}\{t_i - d_i, 0\}, \mathrm{Opt}\left(\{A_1, A_2 \ldots A_{i-1}, A_{i+1} \ldots A_n\}\right) \right\} \right\}$$

$$\underbrace{d_1 - t_i}_{D} \quad \underbrace{d_2 - t_i}_{D} \quad \ldots \quad \underbrace{d_n - t_i}_{D}$$

No. of "distinct" recursive call

Greedy

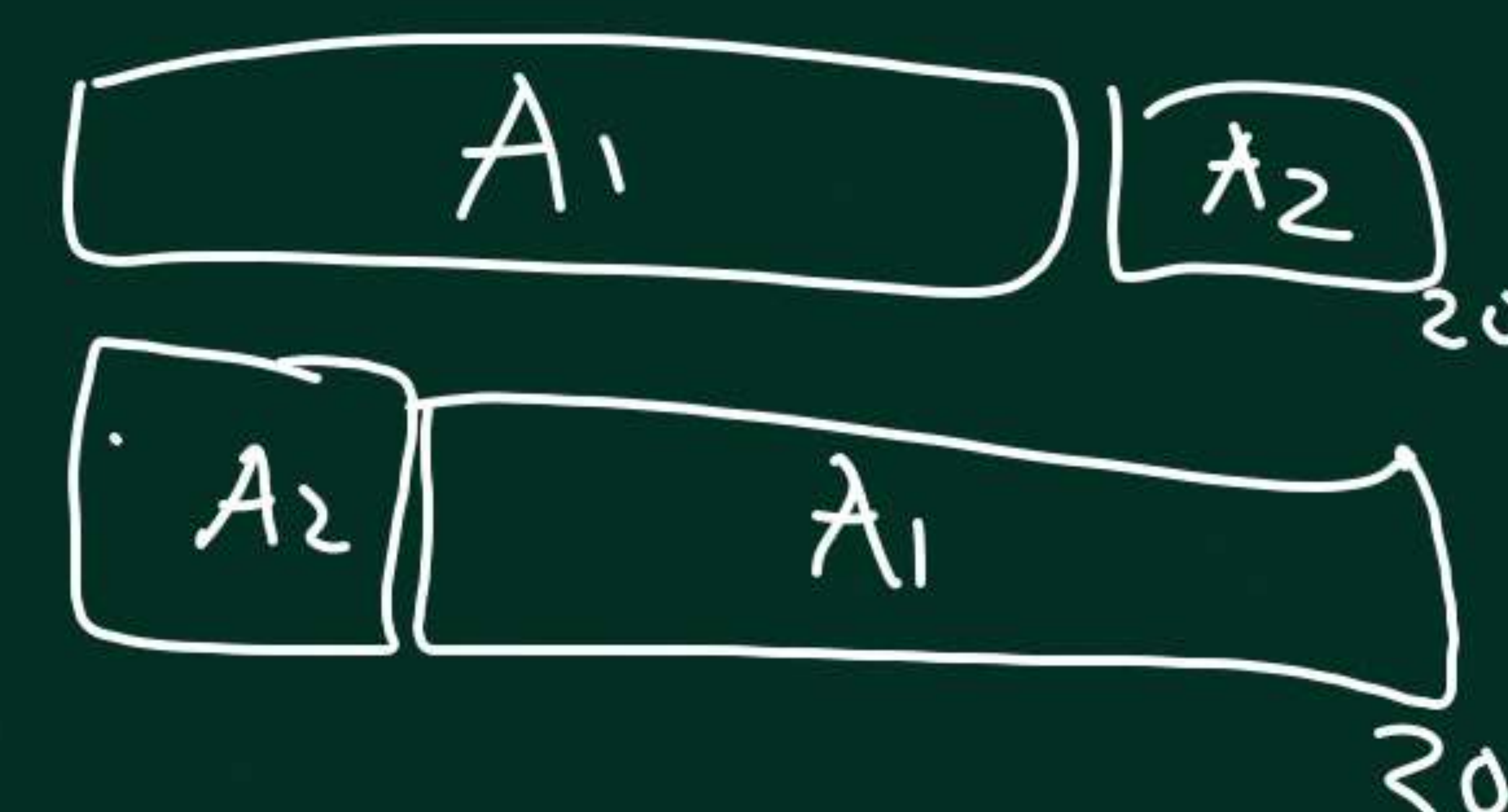
Schedule that assignment first which

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

$L = 15$

$L = 1$ ✓



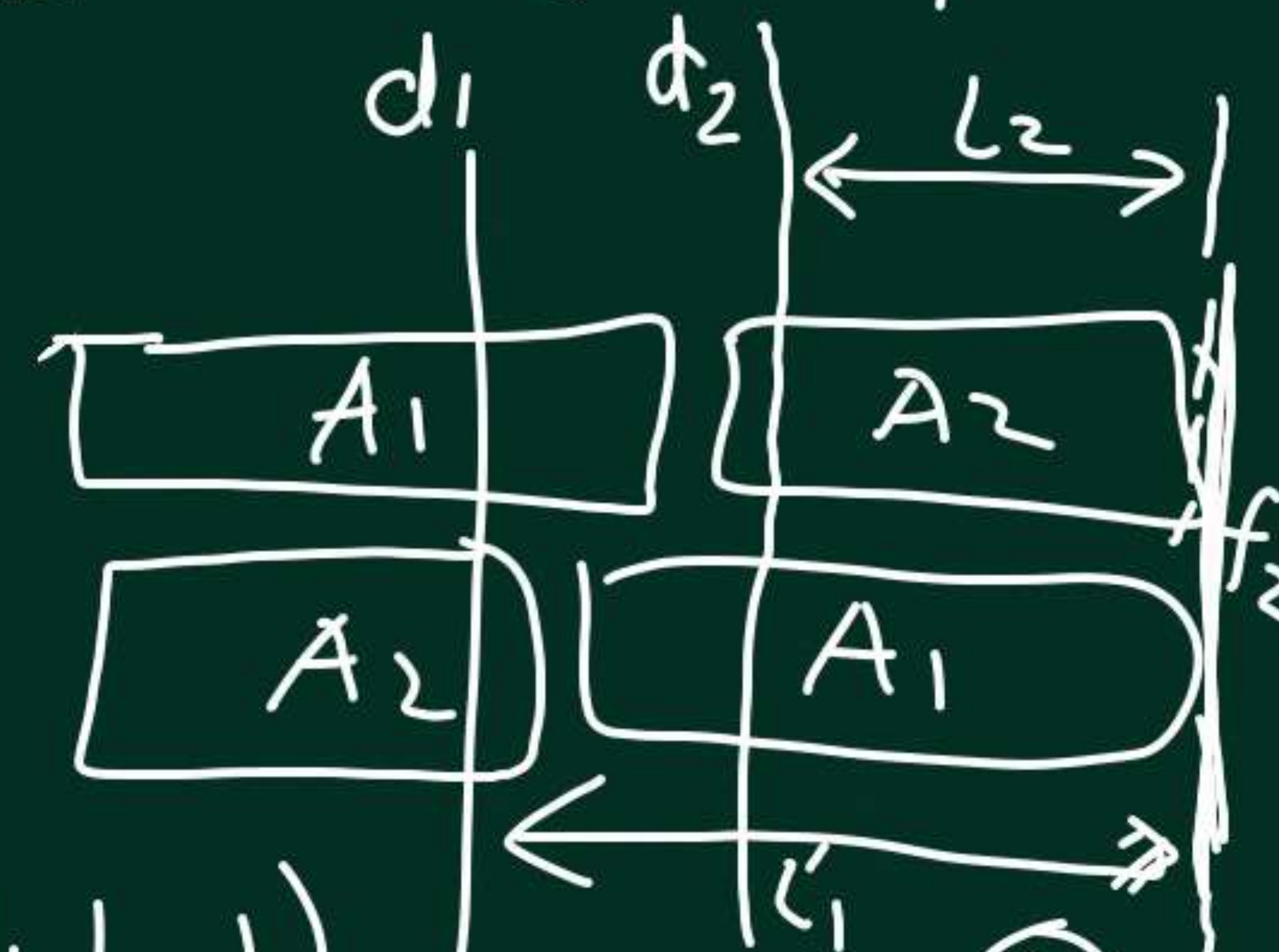
Schedule in the increasing order of deadlines.

Why is this optimal?

$$d_1 \le d_2 \le \dots \le d_n$$

Two Assignments

$d_1 \le d_2$



L_1 L_2

L_1' L_2'

Lateness(shedule 1) $\le$ Lateness(schedule 2).

1. $L_1 \le L_1'$
2. $L_2 \le L_1'$

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

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

t_1

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

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

lateness improves.

Iterated Matrix Multiplication

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

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

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

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

$M_1 \quad M_2 \quad M_3$

$\underbrace{2\times 3 \quad 3\times 4}\quad\underbrace{\quad}\quad 4\times 5$

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

$24 + 40$

$= 64$

Optimal order?

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

$120 + 200 \qquad 60 + 150$

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

$30 + 60 = 90$

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

$P_1 < P_2$

$P_0P_1P_2 + P_0P_2P_3 = P_0P_2(P_1+P_3)$

$P_1P_2P_3 + P_0P_1P_3 = P_1P_3(P_0+P_2)$

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

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

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

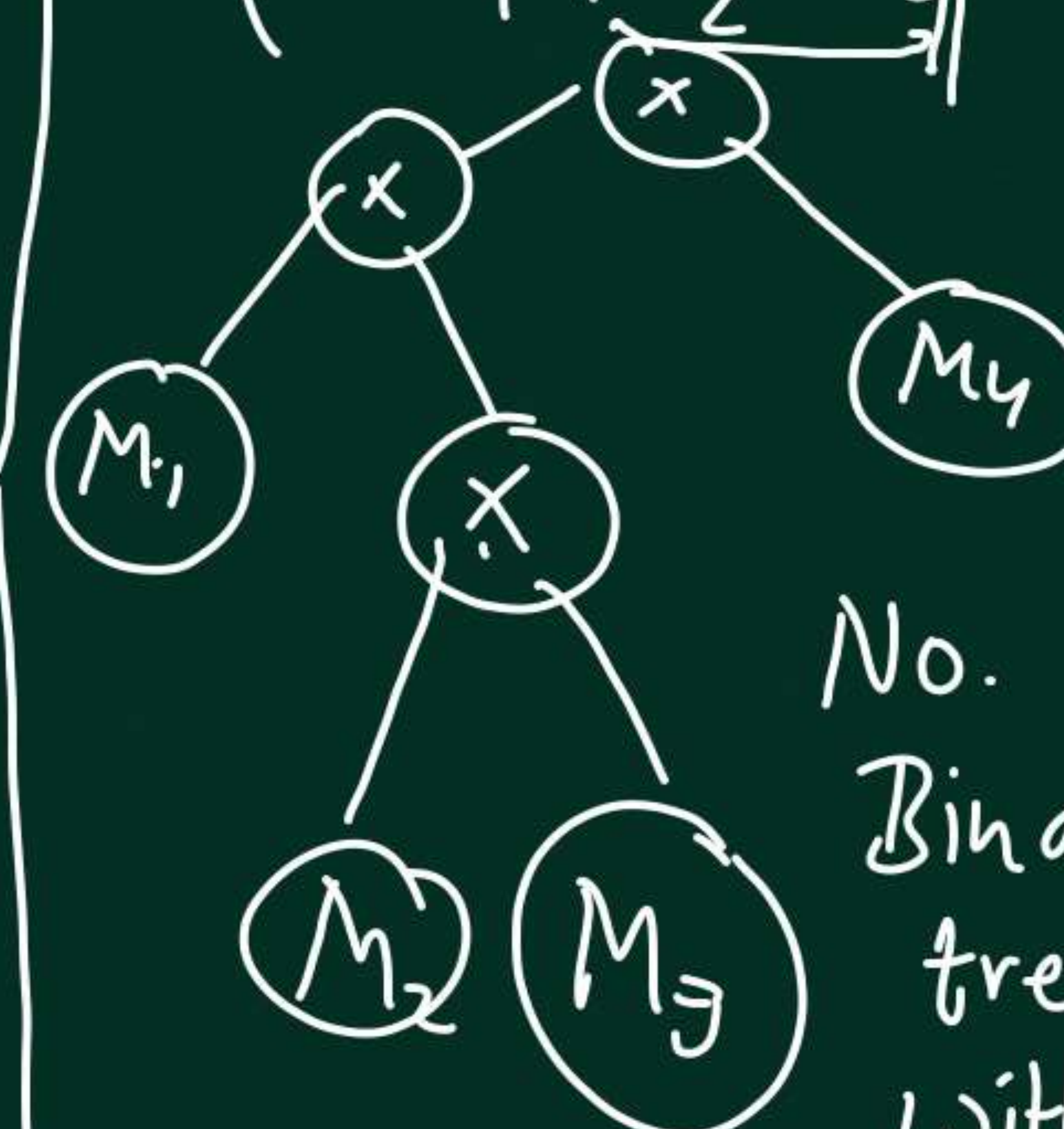
Output Best order

DP 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.

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

$$\times$$

$M_1 M_2$ $\qquad$ $M_3 M_4 M_5$

Classifying possible orders

① first multiplication

② last multiplication

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

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

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

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

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

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

$P_0, P_1, P_2, \ldots$

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

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

Every subsequence is possible

$P_0 P_2 P_3$

$P_0 P_3$

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

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

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

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

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

$P_0 P_1 P_3$

$P_0\, P_1\, P_2\, P_3\, P_4\, P_5\, P_n$

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

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

$$= \min_i \left\{ P_K P_i P_K + \begin{matrix} \mathrm{Opt}(P_K - - P_i) \\ \mathrm{Opt}(P_i - - P_\ell) \end{matrix} \right\}$$

$O(n^3)$ time.

Implementation

Subset Sum problem.

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

Is there a <u>subset</u> whose sum is zero?

Subset Sum problem. (Dynamic Programming)

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

-9

Is there a subset whose sum is zero?

Trivial ← 2^n

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

→ will not have a_n

Is there a subset of $a_1 \ldots a_n$ with sum T

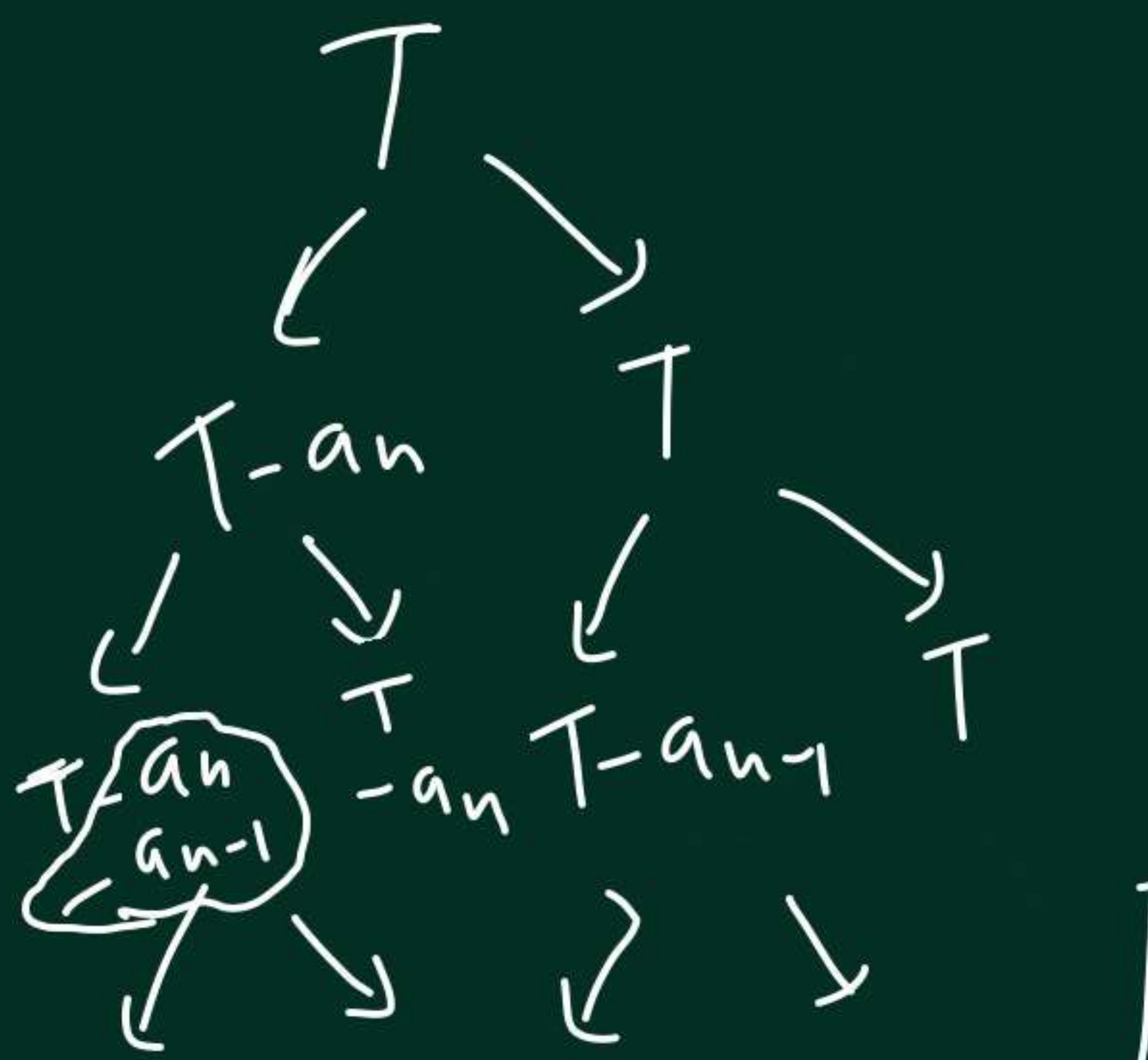
Is there a subset of $a_1 \ldots a_{n-1}$ with sum $T - a_n$

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

No. of distinct recursive calls?

prefix $\leftarrow$ $(a_1, \ldots, a_j,$ Target $t)$

n



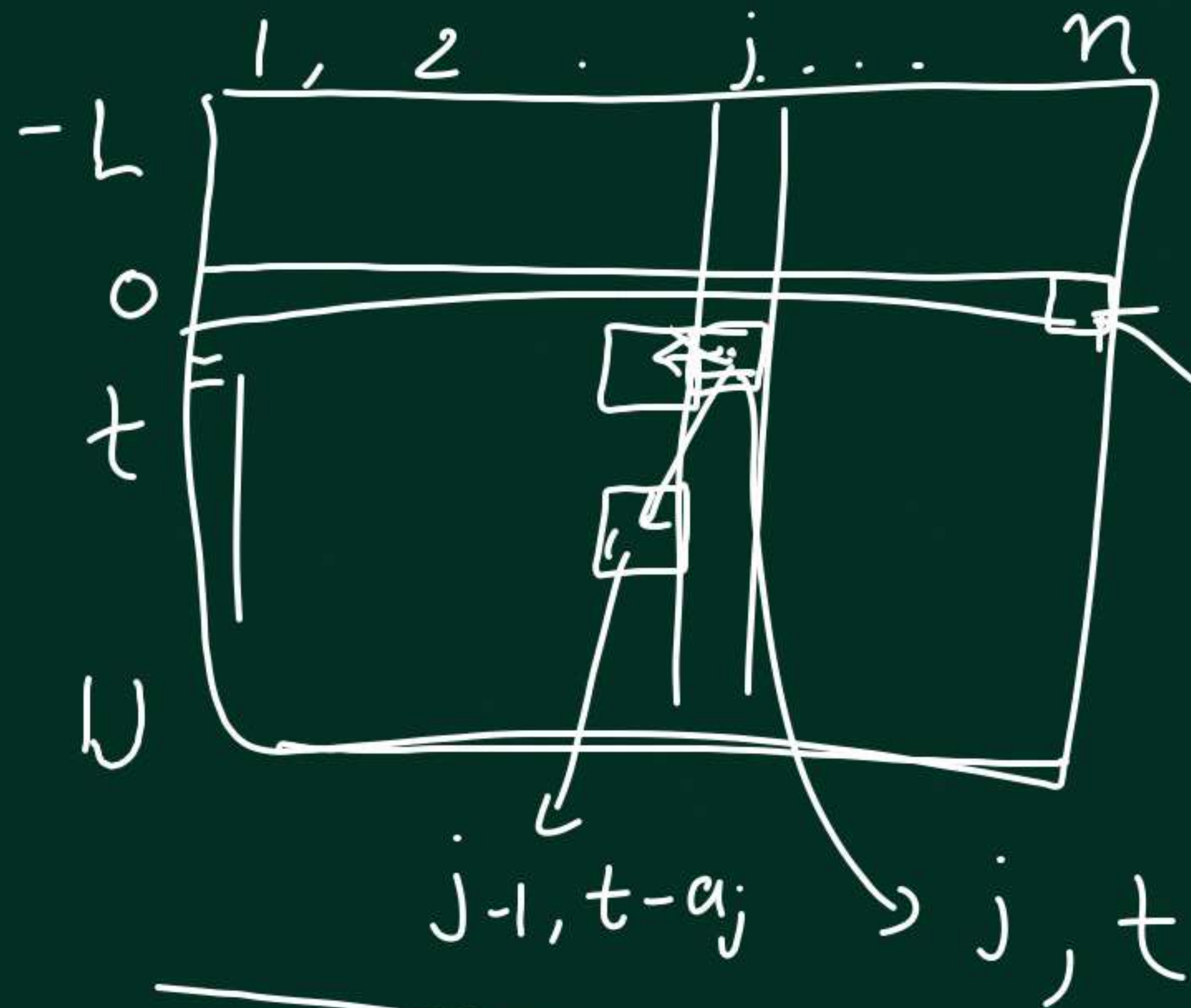
$$\sum_{i:\, a_i<0} \quad \le T \le \sum_{i:\, a_i>0} a_i$$

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

Subset-sum(j, t) ⟵ true if there is subset of $a_1 \cdots a_j$ with sum t

⟵ false othewise

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



Implementation

n numbers
↓
input size
= total no. of bits

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

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.

$O(n \cdot W)$

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

Value $p_1, p_2, \ldots, p_n$

weights $w_1, w_2, \ldots, w_n$

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

Fractional Knapsack.

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

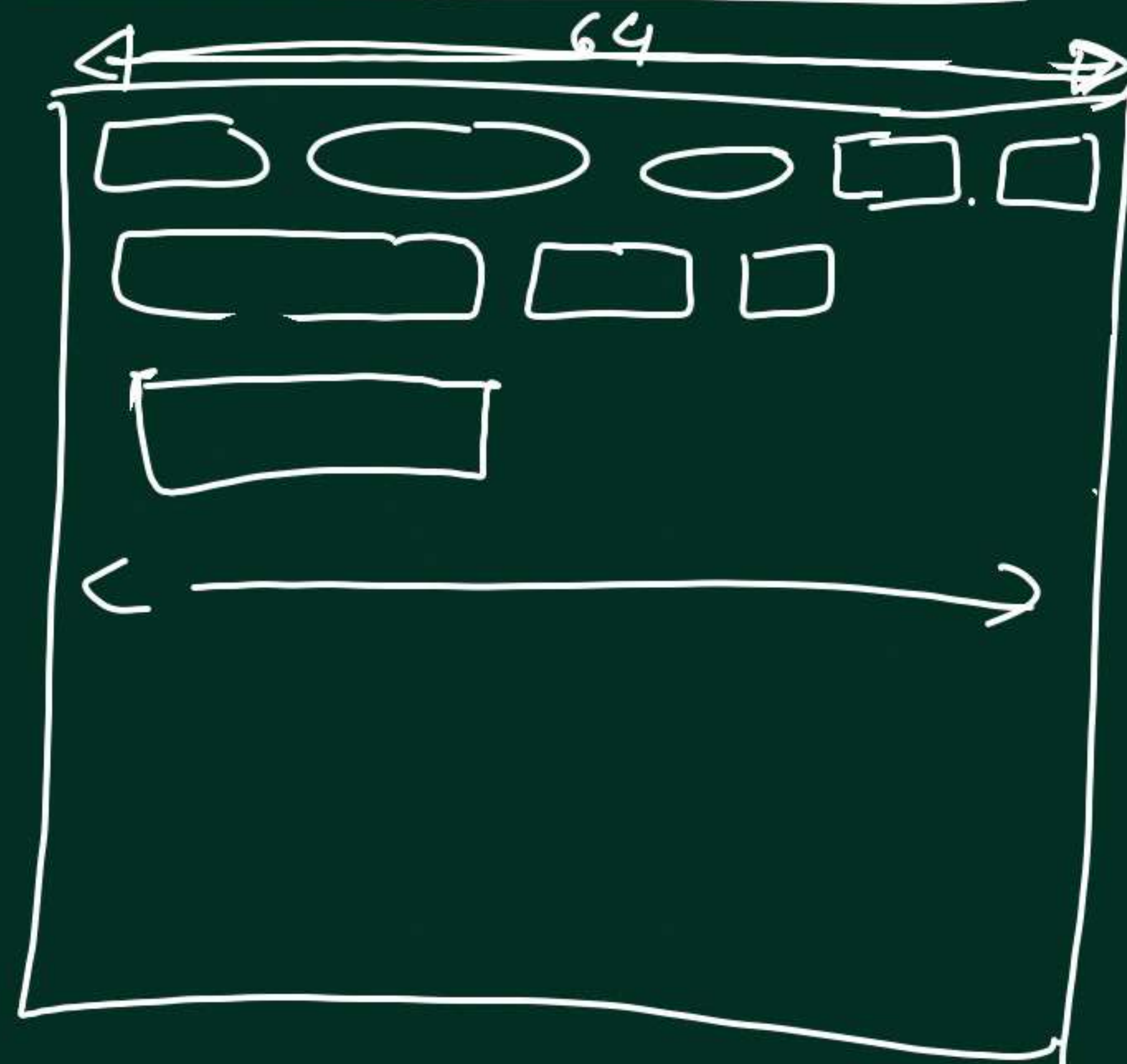
$$\max \sum_i p_i x_i$$

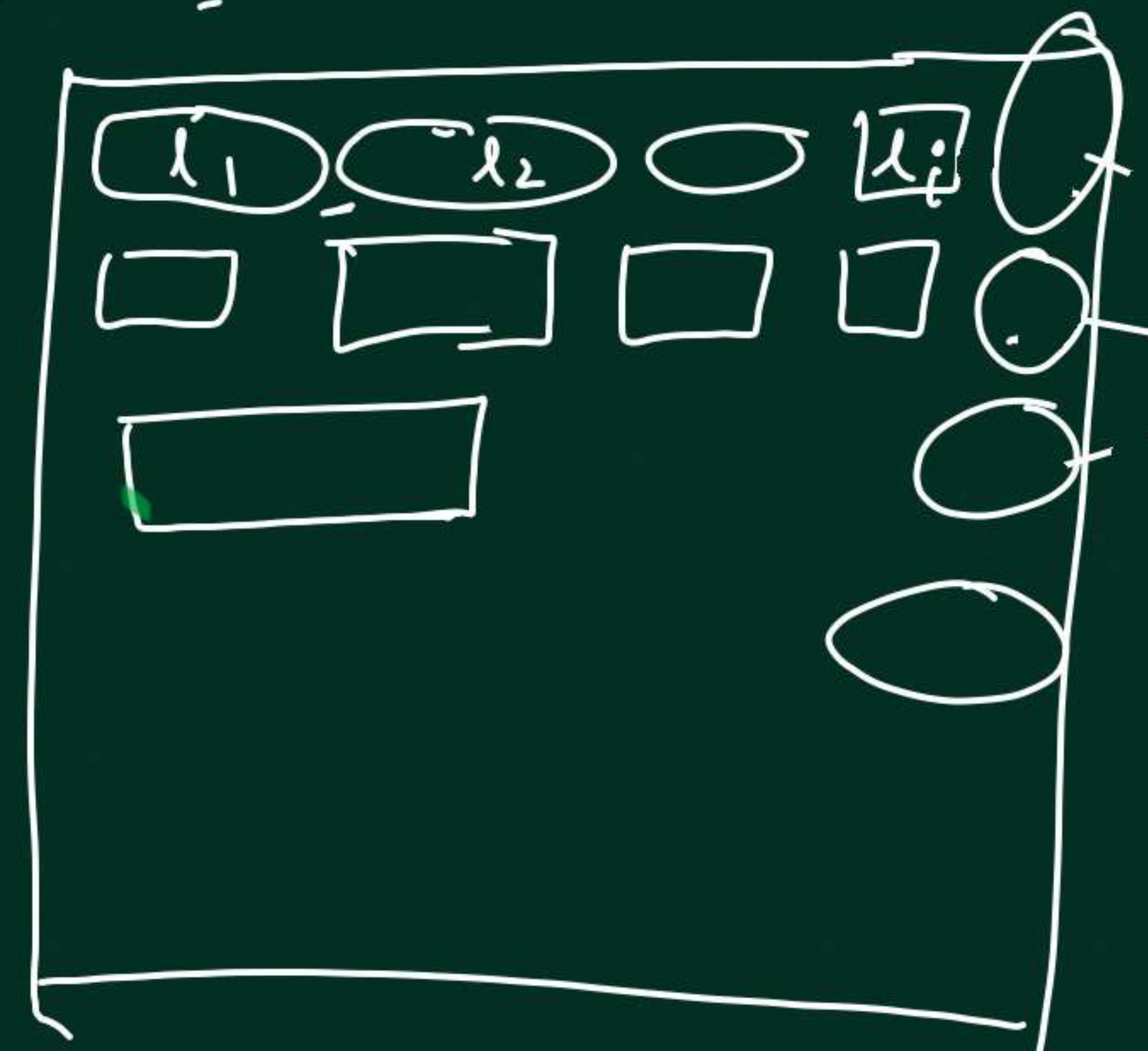
Subject to $\sum_i w_i x_i \leq W$

Greedy algorithm works.

Balanced Margins.

$Slack_1 = L - l_1 - 1 - l_2 - 1 \ldots - l_i$





Input ← sequence of word $l_1, l_2 ---, l_n$

Limit per line L.

$L \leftarrow$ linewidth

Input : $l_1, l_2, \dots\ l_n$

Line k 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

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

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

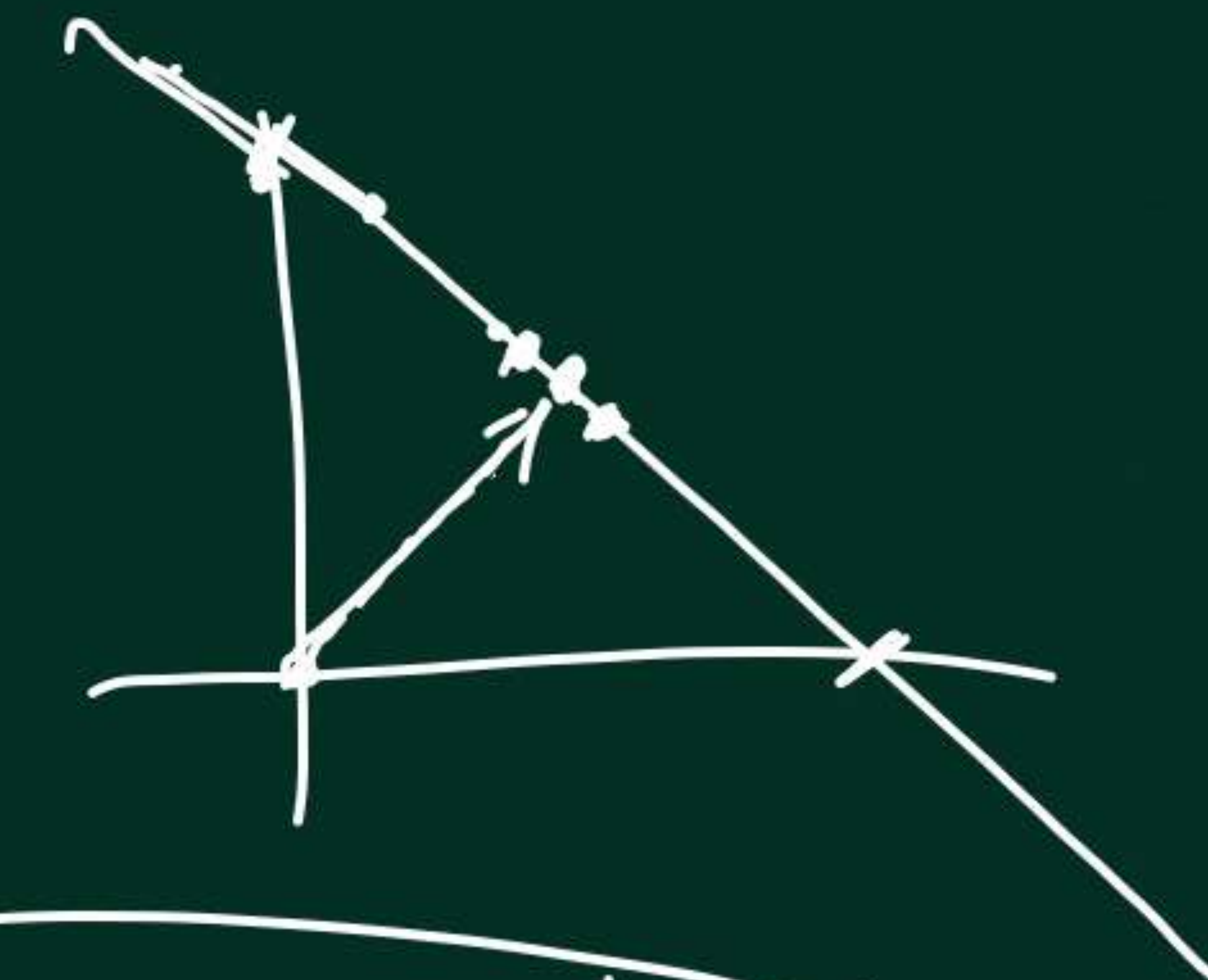
find $a, b, c \in \mathbb{Z}$

$a + b + c = 14$

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

$14 = 10 + 2 + 2$ $\quad 108$

$14 = 4 + 5 + 5$ $\quad 66$



No. of words in line 1 $\quad i_1$

— " — line 2 $\quad i_1+1, i_2$

line 3 $\quad i_2+1, i_3$

⋮

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.

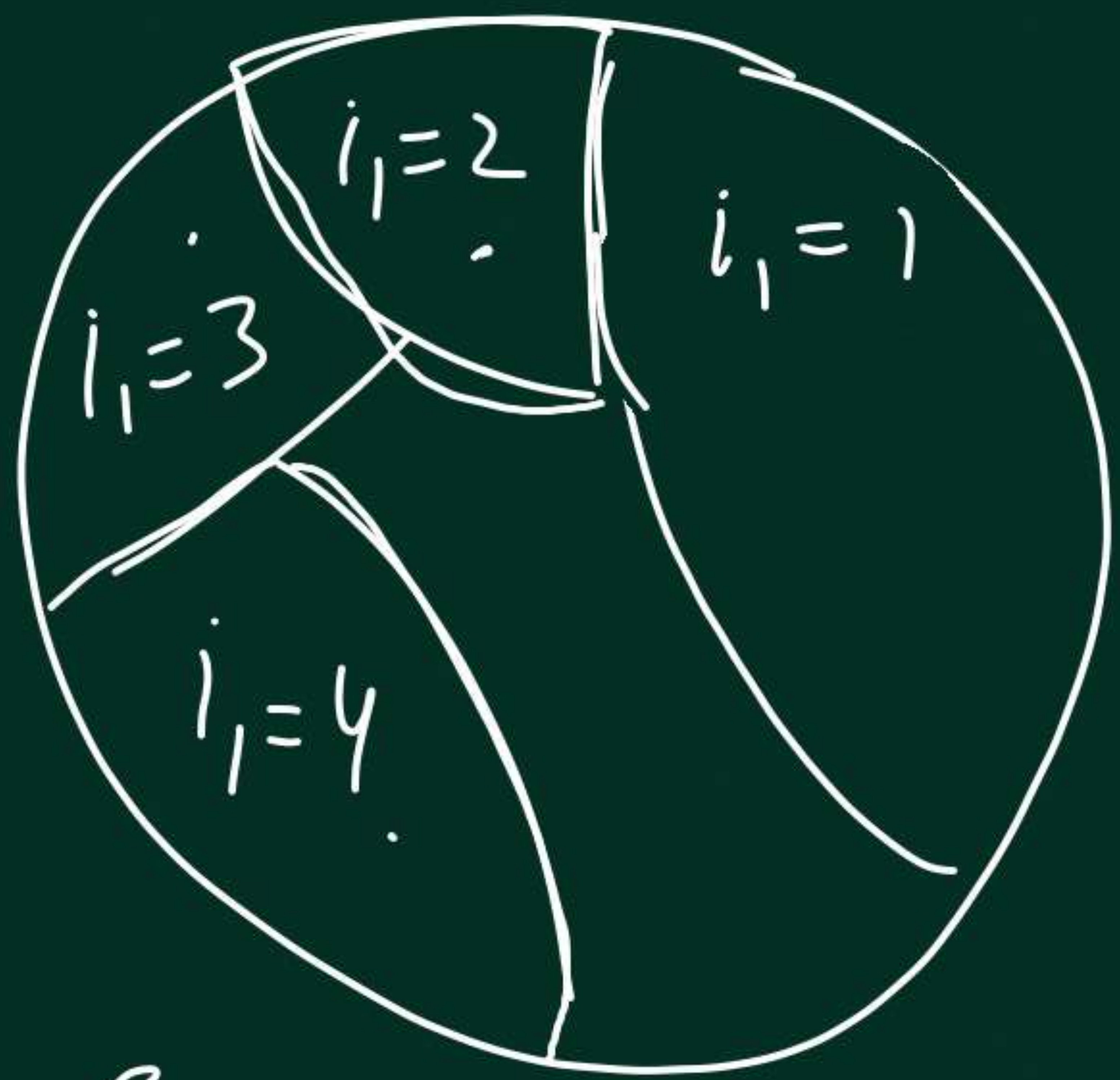
Dynamic Programming

Categorizing Possible solutions

$i_1 \leftarrow$ last word in firs line

$i_2 \leftarrow$ last — second

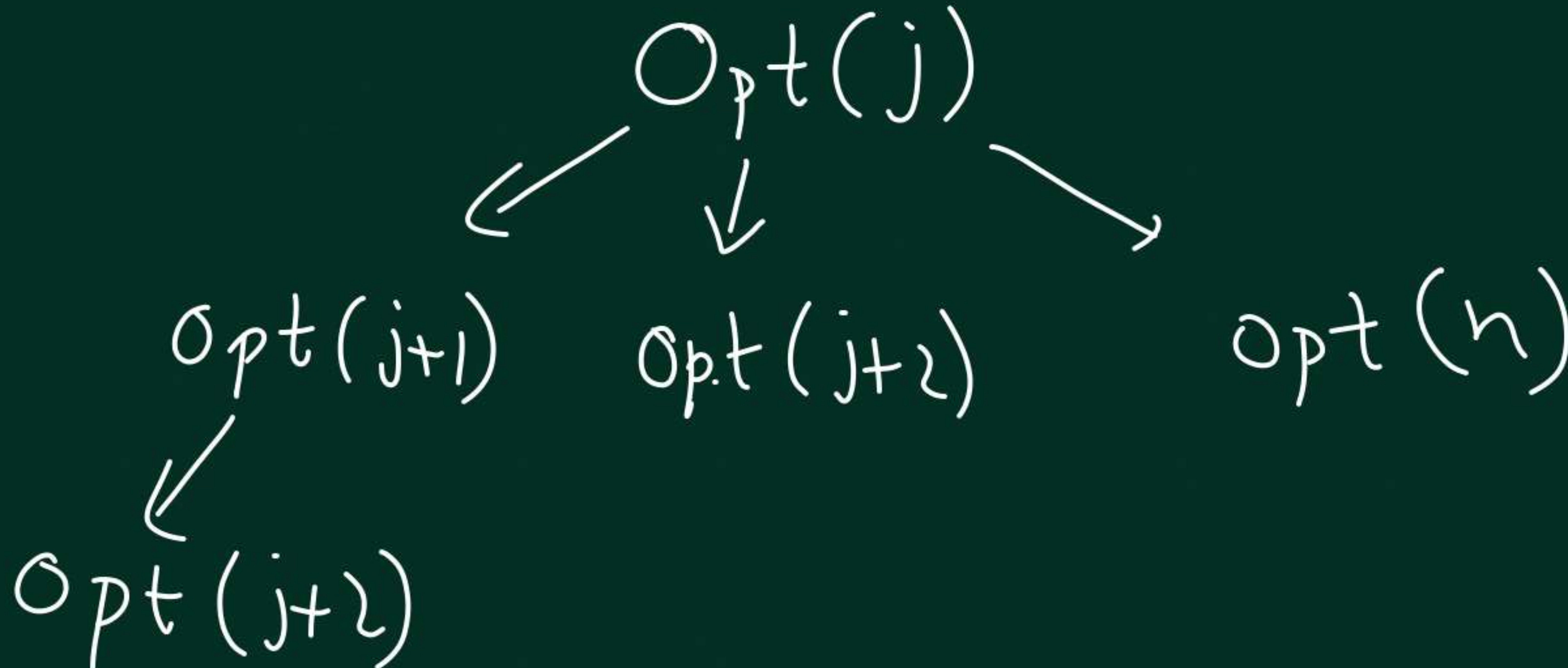
$i_3 \leftarrow$ last --- third.



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

No of "distinct" recursive calls.

$Opt(j) \leftarrow$ optimal value for the words $l_j, \ldots, l_n$



for (j=n to 1)

$$Opt(j) = \min \begin{cases} (L-\ell_j)^2 + Opt(j+1) \\ (L-\ell_j-1-\ell_{j+1})^2 + opt(j+2) \\ \vdots \\ \_\_\_\_ \quad Opt(n) \end{cases}$$

Running time

$O(n^2)$

Implementation

Compute all slack squares

Optimal solution

Before hand $(L-\ell_j-1-\ell_{j+1}-1 \ldots - \ell_k)^2$