① Box Stacking:

Area A2

Area A1

$A1 > A2$
↳ stacking possible.

$l_1 \times b_1 > l_2 \times b_2$
→ sort on the basis of Area.

Max height from box,

$\begin{rcases} l \\ b \\ h \end{rcases}$ ✓ Rotation of boxes are allowed, (we can interchange l, b & h)

we can use more than one instance of same box.

possibility of stacking:
→ $l_1 > l_2$ and $b_1 > b_2$

$l \rightarrow$	3	5	4
$b \rightarrow$	2	4	2
$h \rightarrow$	4	3	6

$[l, b, h] \longrightarrow$
box₁

① max height
↳ *consider all possibility from single box.
* we will consider a config only single time.

No. of box, we will consider.

$\overline{[l \ b \ h]}$, $\overline{[l \ h \ b]}$, $\overline{[b \ h \ l]}$, $\overline{[b \ l \ h]}$
$\quad$ b1 $\qquad$ b2 $\qquad$ b3 $\qquad$ b4
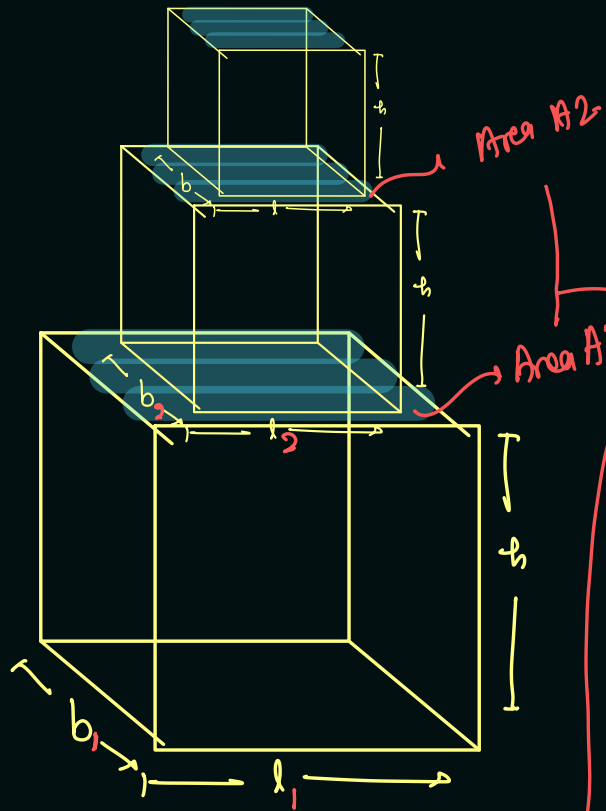
$\overline{[h \ l \ b]}$, $\overline{[h \ b \ l]}$
$\quad$ b5 $\qquad\quad$ b6

$l \to$ 3   5   4

$b \to$ 2   4   2

$h \to$ 4   3   6

$[l \ b \ h], \ [l \ h \ b], \ [b \ h \ l], \ [b \ l \ h], \ [h \ l \ b], \ [h \ b \ l]$

3 config $\to$ consid. l & consid. b

$l \to$	3	3	2	2	4	4	5	5	4	4	3	3	4	4	3	3	6	6
$b \to$	2	4	4	3	3	2	4	3	3	5	4	5	3	6	6	4	3	4
$h \to$	4	2	3	4	2	3	3	4	5	3	5	4	6	3	4	6	4	3
area ($l \times b$)	6	12	8	6	12	8	20	15	12	20	12	15	12	24	18	12	18	24

NOTE: To get the proper of l & b in decreasing, we consider a dependent factor on l & b, which is area $= l \times b$.

$\to$ Sort on the basis of area, [Deasing order] of base.

from all 6 possible order add if upcoming $l > b$

Sort in decreasing order of area (i.e. $l \times b$)

$l \to$	6	4	5	4	6	3	5	3	4	3	4	3	3	4	4	2	3	2
$b \to$	4	6	4	5	3	6	3	5	3	4	3	4	4	3	2	4	2	3
$h \to$	3	3	3	3	4	4	4	4	6	6	5	5	2	2	3	3	4	4

$l \rightarrow$	6	5	6	5	4	4	4	4	3
$b \rightarrow$	4	4	3	3	3	3	3	2	2
$h \rightarrow$	3	3	4	4	6	5	2	3	4

---

dp → | 3 | 3 | 4 | 7 | 9 | 13 | 5 | 10 | 13 |

max from all possible. ⟶ (13) Result

Apply logic of LIS on l and b Simultaneously, and add height.

Maximum length of Pair chain! →

config →   [a, b]   [c, d] ⟶ Combined length ②

chaining →   b < c

left   right

[1, 2] [7, 8] [4, 5]

$\boxed{left \quad \curvearrowright \quad right}$

[1, 2] [7, 8] → [4, 5] ⟶ ②

[4, 5] ↷ [7, 8] ⟶ ③

left   riger

pair →   [a, b] , [c, d] , [e, f] , [g, h]   arr of pairs
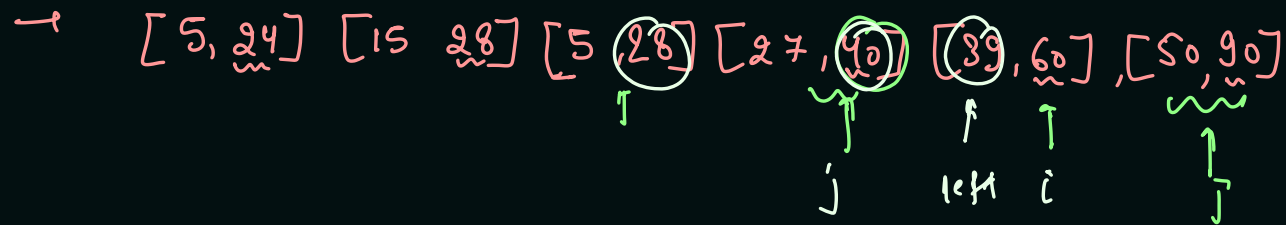
[27, 40], [15, 28], [50, 90]

[5, 29] [39, 60]

[5, 28]

To make the chain first right is smaller
then next left, so it is mandatory that
right should be small so, sort on
the basis of right.

use of

LIS

arr of pair → [5, 24] [15 28] [5, 28] [27, 40] [39, 60], [50, 90]

arr
of
pair → [5, 24] [15 28] [5, 28] [27, 40] [39, 60], [50, 90]

j   left i   j

use of
L1S

①	1	* 1	2	2	2

max of

* → if chain end at 'i' then max. possible length.

$$[5, 24] \rightarrow [27, 40] \rightarrow [50, 90]$$

count Palindromic substring :→

String → <u>a b c c b c</u>

<u>Brute force</u>

① Generate all palindromic substrings, then check if it is palindromic or not.

② optimisation. using DP. →

Recursion
↳ memoisation
       ↓
      tabulation ← 2D-DP

tabulation is optimise than memoisation

one D-DP

TSP 1 & 2

memoisation → Tabul

	a gap-0	b₁ gap-2	c₂ gap-3	c₃ gap-4	b₄ gap-5	c₅
a₀	a T	ab F	abc .F	abcc F	abccb F	abccbc F
b₁	●	b T	bc F	bcc F	bccb T	bccbc F
c₂	●	●	c T	cc * T	ccb F	ccbc F
c₃	●	●	●	c T	cb F	cbc T
b₄	●	●	●	●	(T) b	bc F
c₅	●	●	●	●	●	c T

Ending →

l-length-T   2-lgth-gap-1
gap-0

Starting

Ch₁ ——— ch₂

Middle

True

False

Base → case → Propagm

call → to DP

ch1 == ch₂ → string - True
                    False.

✱ → substring start from
2 index to 4 index
i.e. c c b
   ↳ True / False ;

String → a b c c b c

$\boxed{n=6}$

gap0   gap1   gap2   → Ending

a_0 → b_1 → c_2 ^{gap3} C_3 $_{gap4}$ b_4 $_{gap5}$ C_5

Starting a_0 $i=0$

Starting b_1

C_2

C_3

b_4

C_5

	a_0	b_1	c_2	C_3	b_4	C_5
a_0	T	F	F	F	F	F
b_1		T	F	F	T	F
C_2			T	T	F	F
C_3				T	F	T
b_4					T	F
C_5						T

In x^{th} gap →

→ x^{th} gap begin with
   $i=0$ & $j=x$

→ i & j increment . $i++$, $j++$.

→ $j < n$ → Running condition

Equal

⓪ — ②

middle part → True.

③ $\,$ ④—④ $\,$ ⑤
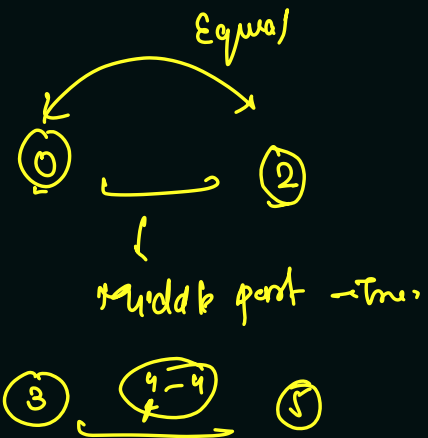
Longest palindromic Substring :

String → a b c c b c   → longest palindromic length??

→ Ending

	a_0	b_1	c_2	c_3	b_4	c_5
a_0 Starting	T *a*	F *ab*	.F *abc*	F *abcc*	F *abccb*	F *abccbc*
b_1	●	T *b*	F *bc*	F *bcc*	T *bccb* '14 length-4-1?	F *bccbc*
c_2	●	●	T *c*	T *cc*	F * *ccb*	F *ccbc*
c_3	●	●	●	T *c*	F *cb*	T *cbc*
b_4	●	●	●	●	T *b*	F *bc*
c_5	●	●	●	●	●	T *c*

if $dp[i][j]$ is true

└ that mean
String from i to j is palindrum.

└ length of that string is

└ $j - i + 1$