

Permutations

→ Given a string, we have to print all the different permutations of the string.

→ "abc" →

a b c
a c b
b a c
b c a
c a b
c b a

} 3! → 6 permutations

"abaa" →

2

$$\frac{4!}{3!} \rightarrow \underline{\underline{4}}$$

abaa

aaba

aaab

baaa

4 permutations

aabacc →

6!

3! 2!

abc \rightarrow abc
acb

bac
bca

cab
cba

\rightarrow every ^{unique} character of the string becomes the first character of some permutation

Then calculating permutations of the remaining chars

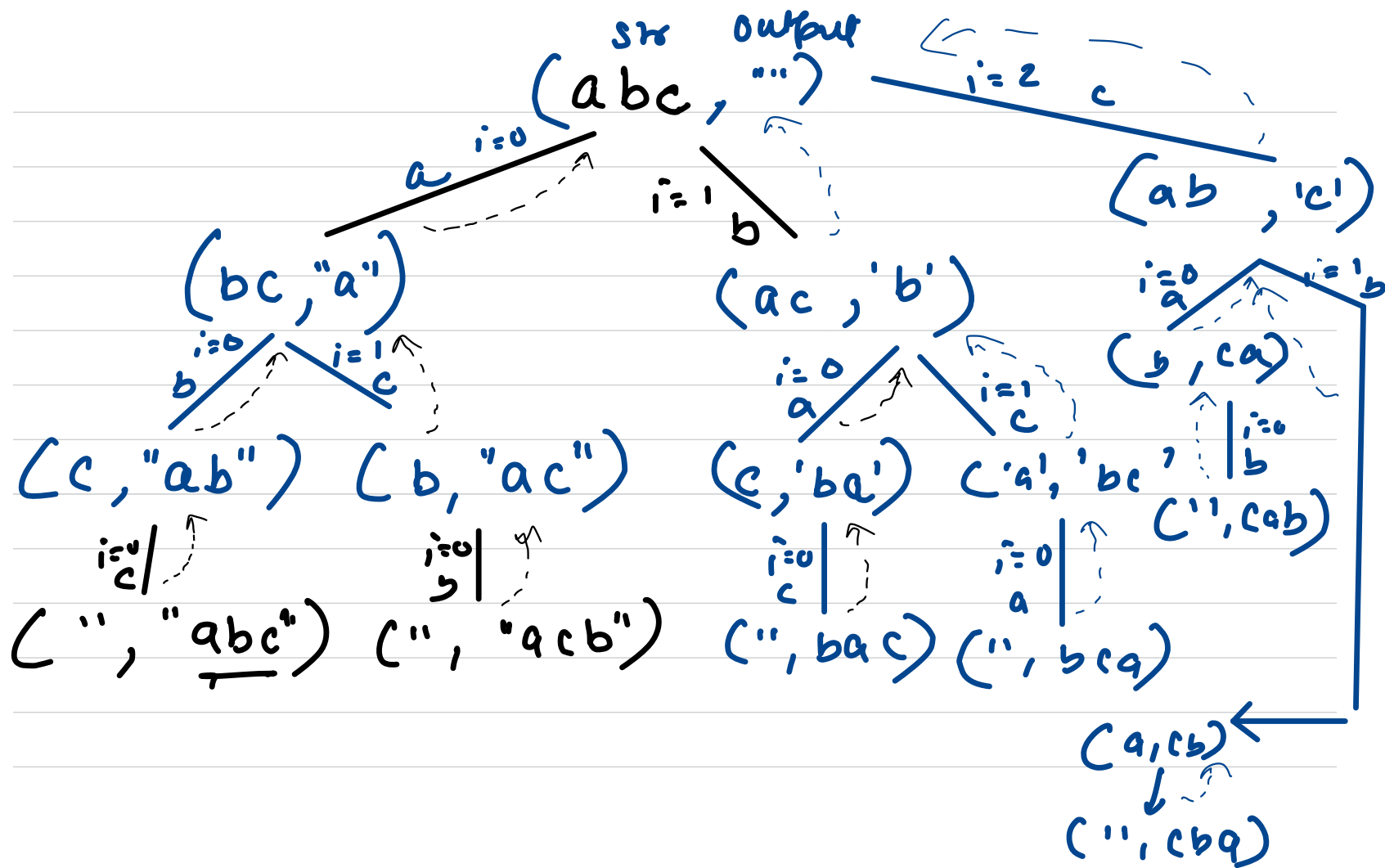


$f(\text{str}, \text{output}) =$

function prints all
permutation of str

and every perm is
stored in output

```
for (i=0; i < str.length; i++) {  
    ch = str[i];  
    ros = str.substr(0, i) + str.substr(i+1);  
    f(ros, output + ch)  
}
```



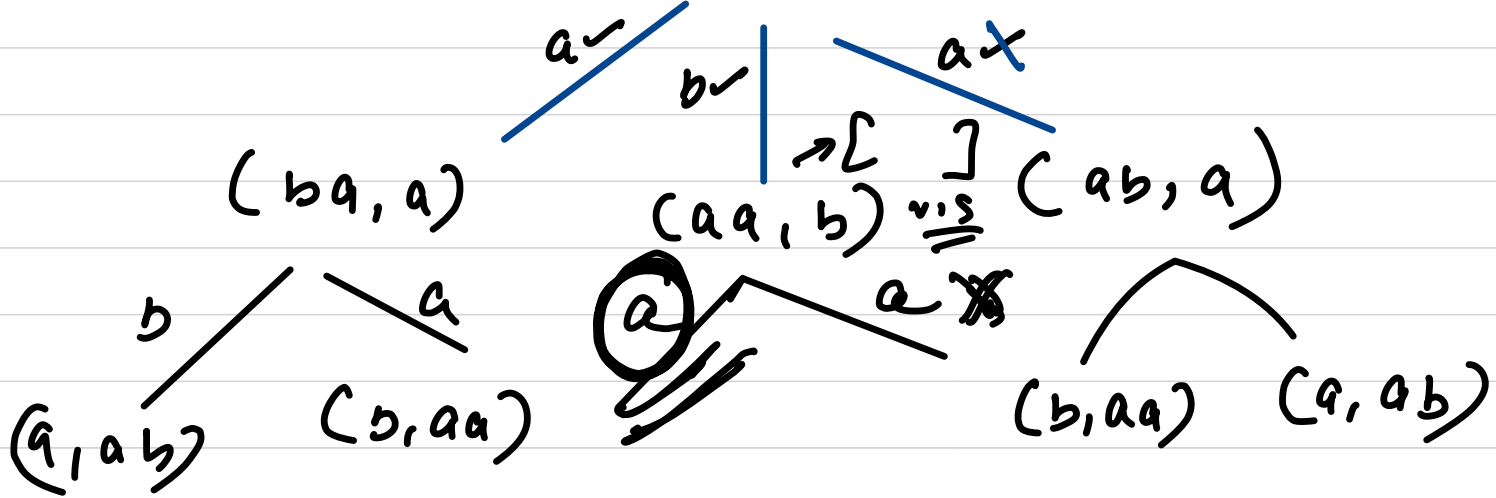
$s = a b c d \rightarrow \underline{\underline{acd}}$

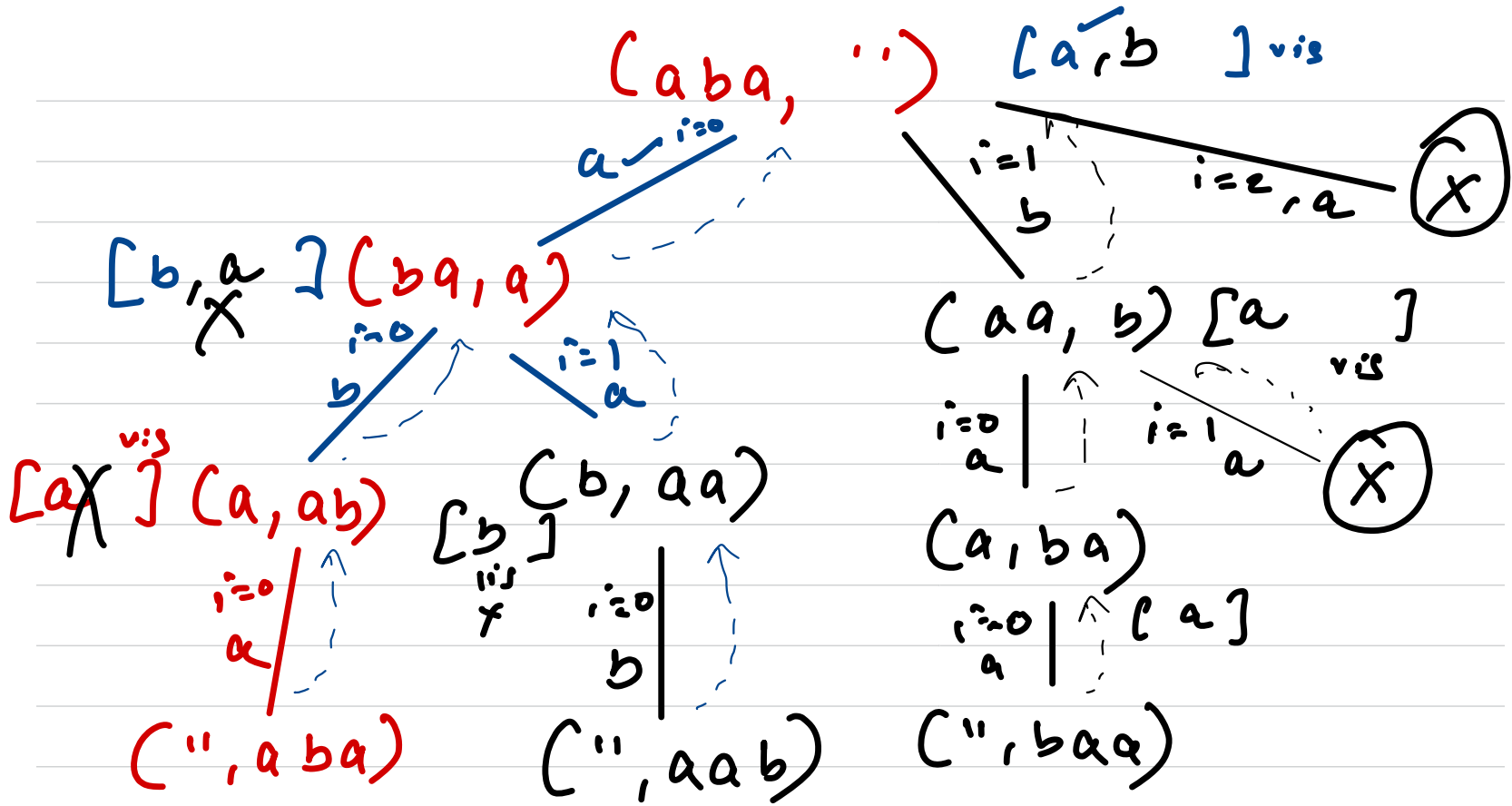
$s.\text{substr}(0,1) + s.\text{substr}(2)$
 $a \quad + \quad cd$
 $\quad \quad b$
 $\quad \quad acd$

$s.\text{substr}(0,2) + s.\text{substr}(3)$ abd
 $ab \quad + \quad d$
 $\quad \quad \downarrow$
 $\quad \quad \underline{\underline{abd}}$

$(\underline{a}b^2, \dots)$ [

vis





$$T(n) = n \times T(n-1) + n^2 c$$

↓

no. of ops
for permutations
of string of
length n.

→ $O(n!)$

Backtracky

(cba, i=0)

ω A

a d=0

d=1 c d=2

(a**cb**, i=1)

(cba, 1)



j=1 d=2

d=1 a d=2 b

(abc, 2)

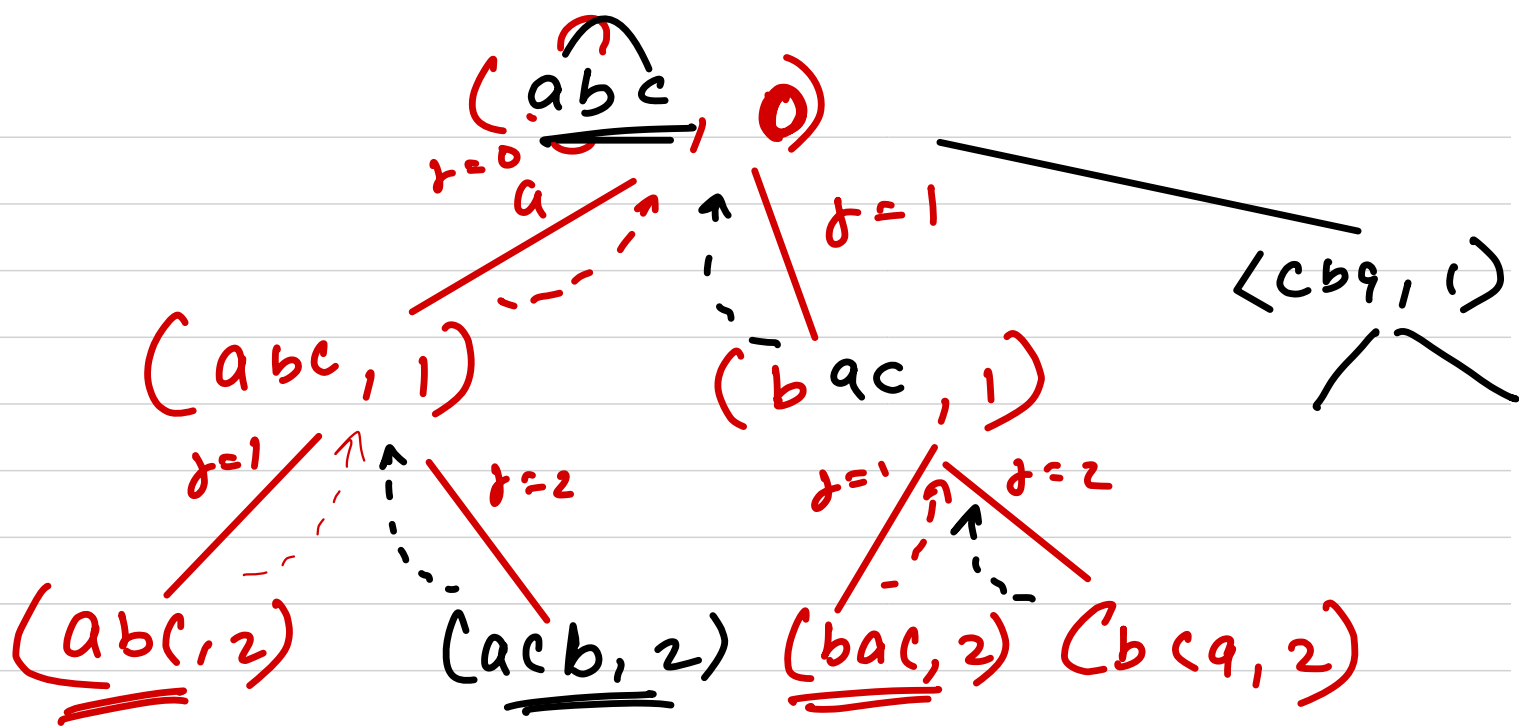
(a**cb**, 2)

(cab, 2)

(cba, 2)

$f(str, i)$ = for($j=i; j < str.length; j++$)
 swap (str, j, i)
 $f(str, i+1)$
 swap (str, j, i)
 ↓
return all permutation
of str starting from index i

ans = $f(str, 0)$



Median of 2 Sorted Arrays

→ asc

We will be given 2 sorted arrays, our task is to find the median of these sorted arrays, where the no. of elements in the first array can be different than the other.

$[7, 12, 14, 15]_n$

$[1, 2, 3, 4, 9, 11]_m$

→ 1, 2, 3, 4, 7, 9, 11, 12, 14, 15

→ 8 ans

$(n, m) \leq 10^6$

10, 1, 6, 3, 5
→
1, 3, 5, 6, 10
↑
median

10, 1, 2, 3, 5, 6
→ 1 2 3 5 6 10
↓
middle

Brute force

→ 1) place second array after
first array

2) Sort this combined data

3) Return the mid.

[]_n

[]_m

[]_n + []_m → []_{n+m}

$O((n+m) \log(n+m))$

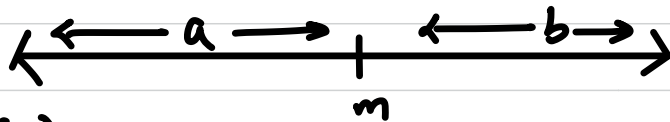
Merge two sorted arrays ==

↳ $O(n+m)$

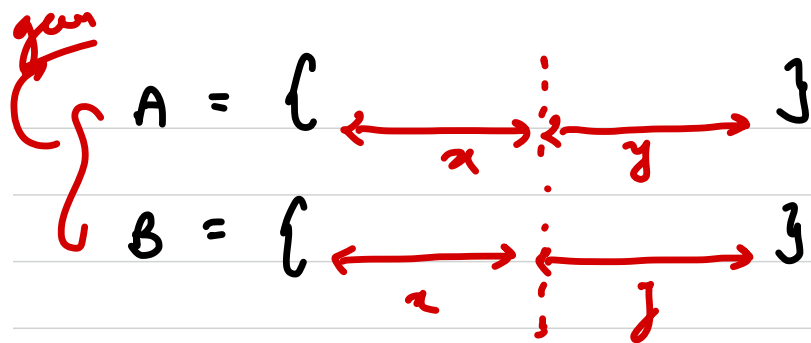
↳ Return the middle.

How can we optimize

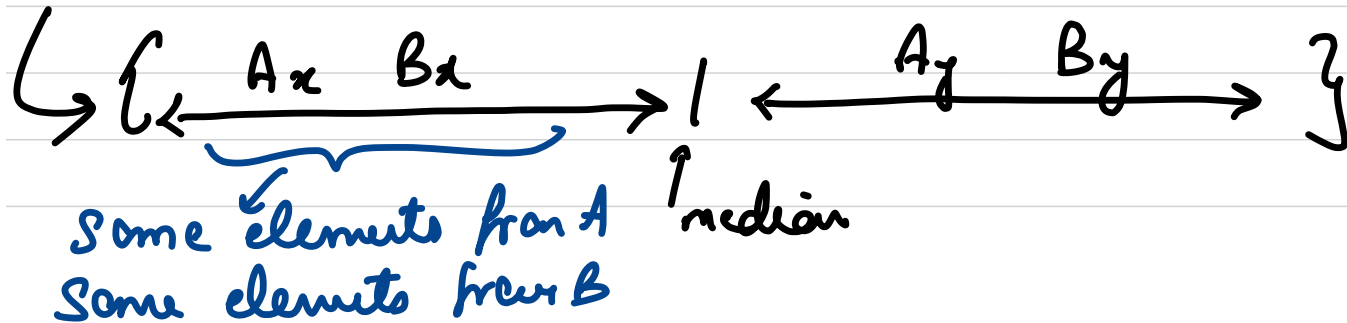
→ Observation → In our final sorted array, median is the middle most element. So the no. of elements to the left of mid is going to be equal to no. of elements to the right of mid.



$$\underline{\underline{\text{len}(a) = \text{len}(b)}}$$

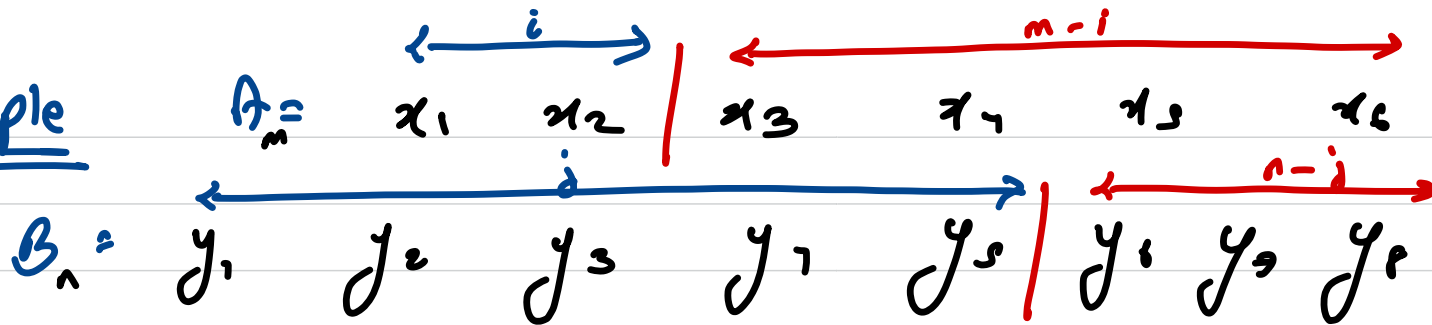


find array \rightarrow merge (A, B)
 C



after merging in sorted fashion, before median,
we will be having some elements from A &
some elements from B will be on the left
of median. & Similarly for right side.

Example



$$C = \underline{y_1 y_2 x_1 y_3 y_4 x_2 y_5} \text{ } m \text{ } \underline{x_3 x_4 y_6 y_7 x_5 y_8 x_6}$$

$$\left. \begin{aligned} i + j &= (m-i) + (n-j) \\ \text{len}(left) &= \text{len}(right) \end{aligned} \right\} \underline{\underline{\text{properly}}}$$

→ Length of array C will be $m+n$

and median will lie on the position

$$\left(\frac{m+n+1}{2} \right) \rightarrow \underline{\underline{\text{median}}}$$

middle
point

Now if we know how many elements from array A are going to contribute to the left of median, then we can easily calc the no. of elements that will participate on the left side of median from array B.

So we know length of final array $\rightarrow m+n$

So we also know how many elements on left half of the array.

If you say no. of elements in A which will be part of left half is x then

no of elements in B which will be part of left half is $\left\lfloor \frac{m+n+1}{2} - x \right\rfloor$

$A = a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6$

$B = b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7 \quad b_8 \quad b_9$

$$a_2 \leq a_5$$

and

$$a_2 \leq b_6$$

$$b_5 \leq b_6$$

$$b_5 \leq a_3$$

$\langle a_2, b_5 \rangle$

rightmost of
left side

$\langle a_3, b_6 \rangle$

leftmost of
right side.

These relations will only work when the
Split is correct.

during the partition we can check for the inequality
if they work then we have the answer
otherwise not.

if $a_2 \neq b_6 \rightarrow$ now because a_2 is more than b_6 , we need to reduce its value. So we go left.

if $b_5 \neq a_3 \rightarrow$ now a_3 is less than b_5 so we need to increase it, that means move pointer to right.

Using this we will be able to make
correct partition on array A & then based
on that partition we can get partition on
array B.

$[7, | 8, 12, 14, 15]_5$

$[1, 2, 3, 4, | 9, 11]_6$

$6 + 5 = 11$
 $(5) \text{ } M \text{ } (5)$

$14 \leq 2 \quad ? \quad \rightarrow \underline{\underline{\text{No}}}$

$8 \leq 4 \quad ? \quad \underline{\underline{\text{No}}}$

$7 \leq 9 \quad ? \quad \underline{\underline{\text{Yes}}}$

$4 \leq 8 \quad \rightarrow \underline{\underline{\text{Yes}}}$