(2) a) False. → counterexample Q1.

OR ↳

$b_1 : g_1 g_2$	$g_1 : b_2 b_1$
$b_2 : g_2 g_1$	$g_2 : b_1 b_2$

b) True.

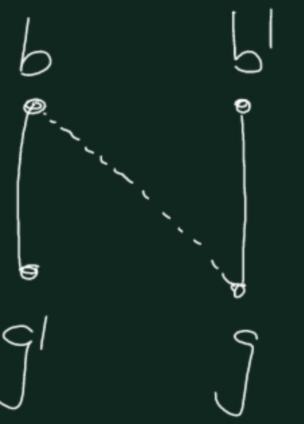
If b is first on g's list,
and g is first on b's list.
Then in all STABLE MATCHINGS
b is matched with g.

There can be many stable matchings in this case and the stable matching given by the Gale-Shapley algorithm is only one of them.

→ Don't use GSA in the proof.

Valid explanation: (without GSA)

If M is a matching where b and g are not matched, then they form a rogue couple, so M is not stable.



Significant inversions

$A[1:n]$ → We will sort A along the way.

① Divide A into 2 smaller arrays

$q = \lfloor \frac{n+1}{2} \rfloor$ $A[1:q]$ and $A[q+1, n]$

② Recursively solve for these.

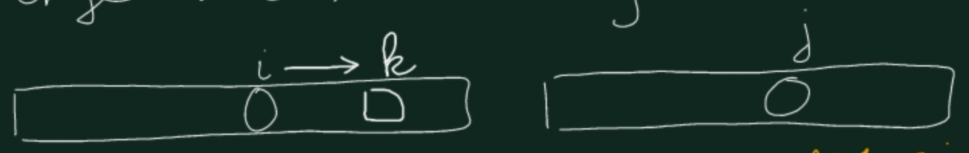
If they have 1 element → return 0.

(3.) Now we have sorted arrays $A[1,q]$ and $A[q+1,n]$ and we know the number of significant inversions in these

sinv 1 in $A[1,q]$

Sinv 2 in $A[q+1,n]$

Merge the 2 subarrays



$A[i] \leq A[j]$ ← no significant inv we have to copy $A[i]$ — $+0$

3 cases:

- $A[j] < A[i] \leq 2A[j]$ * — $+(q-k+1)$
- $2A[j] < A[i]$ ← we have a significant inv we copy $A[j]$ — $+(q-i+1)$

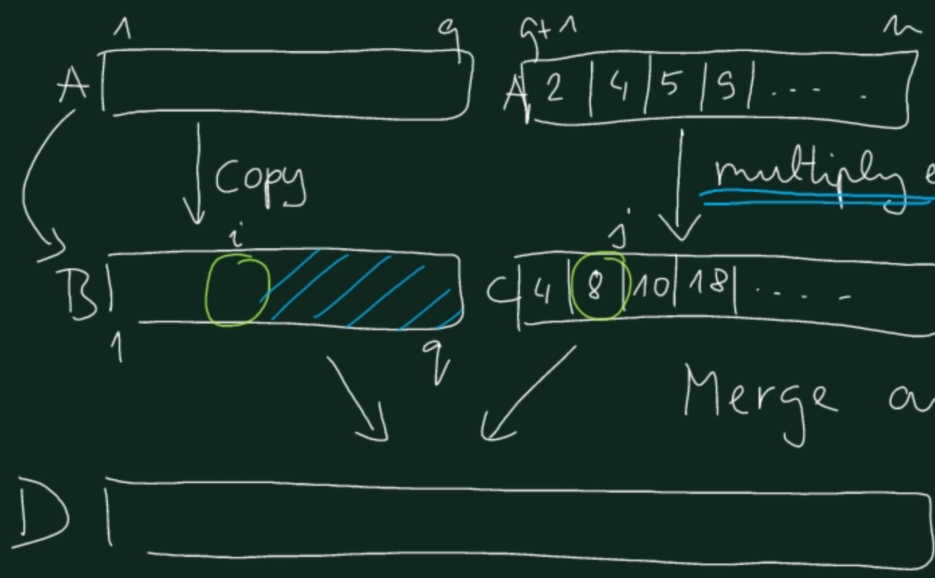
* use an index k to find the first $A[k]$ in $A[i..q]$ for which $A[k]$ and $A[j]$ are in significant inversion $A[j] < A[i] \leq 2A[j] < A[k]$

sinv3

$O(n)$ time?

→ k is never decreasing yes it's $O(n)$

Another solution:



$B[i] > C[j]$ ← NORMAL INVERSION HERE $+(q-i+1)$

$A[i] > 2A[j]$ ← Significant inv. in the original array — // —

→ The number of normal inversions between B and C equals the number of significant inversions between A[1;q] and A[q+1,n]

→ Now we have the number of sign. inv. between the 2 arrays.

→ We can throw away B, C, D

→ Do a normal merging of A[1,q] and A[q+1,n] with no counting. → To have A[1;n] sorted.

Not good:

if $A[i] \leq 2A[j]$

then copy $A[i]$

i++

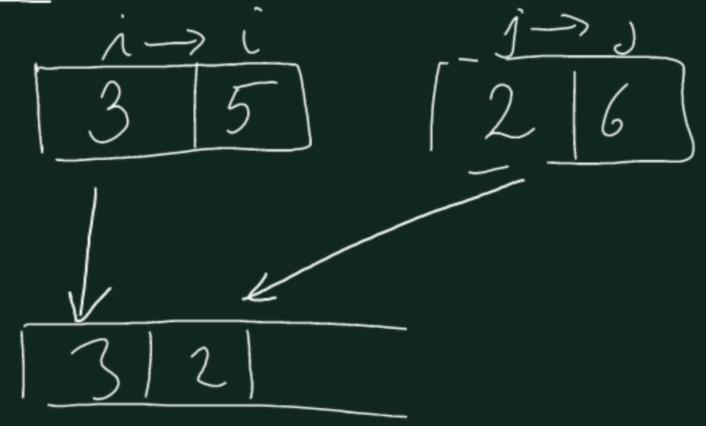
else

{copy $A[j]$

inv3 := inv3 + (q+1-i)

j++

counterexample:

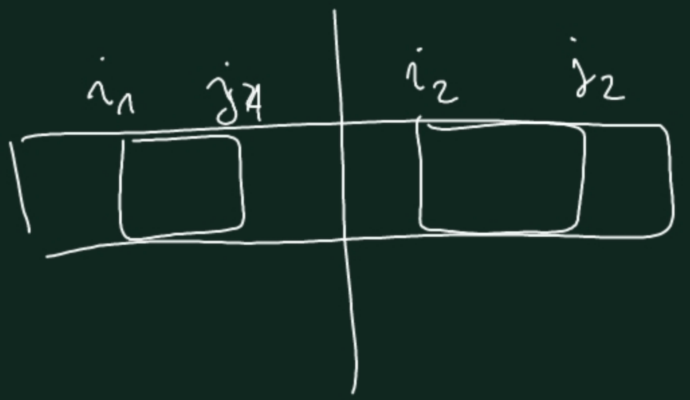


5.

$A[i] + \ldots + A[j]$ maximal

Beginning: usual

1. Divide $A[1,q]$ $A[q+1,n]$



2. Solve recursively

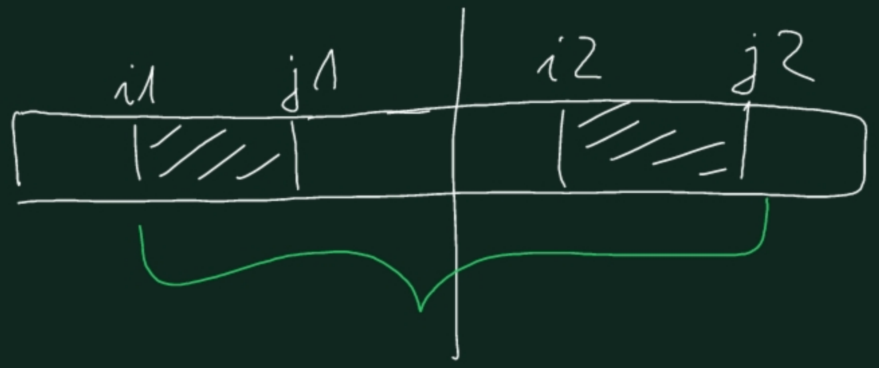
max1
i1
j1

max2
i2
j2

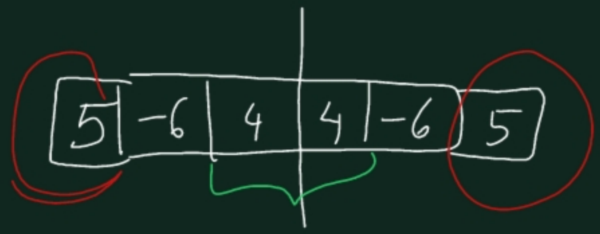
If the maximal sum in $A[1,n]$ is at $A[i,j]$ then there are 3 cases:

1. $1 \le i \le j \le q \implies i = i1,\ j = j1$
2. $q+1 \le i \le j \le n \implies i = i2,\ j = j2$
3. $1 \le i \le q < j \le n$?

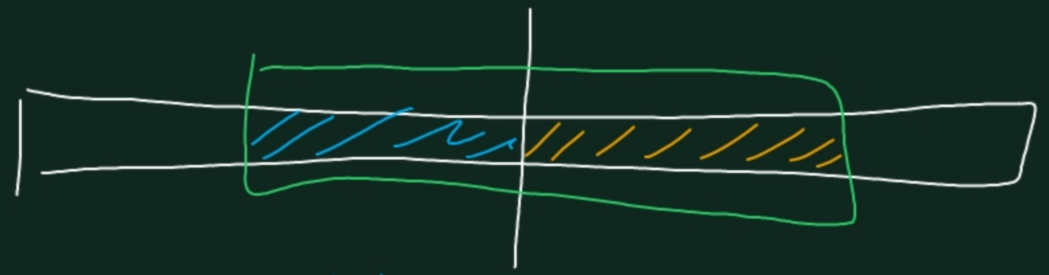
It is not good to take the sum from i1 to j2



counterexample:

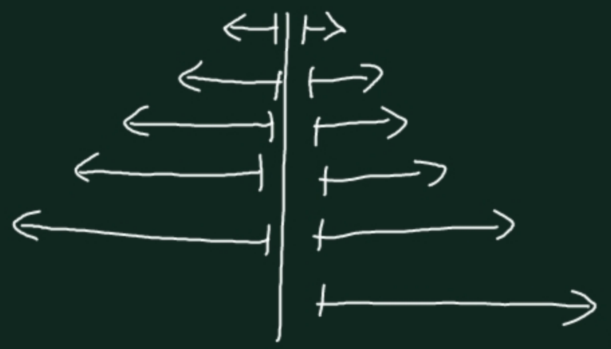


Idea:



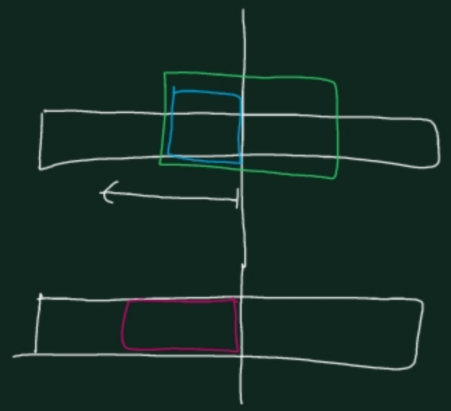
green = blue + orange

blue is the max of these



orange is the max of these

If the max sum is taken in the "middle" green
(so it contains elements from $A[1,q]$ and $A[q+1,n]$ too)
then the sum of the elements in $A[1,q]$
will be the max of:



$A[q]$
$A[q] + A[q-1]$
$A[q] + A[q-1] + A[q-2]$
$A[q] + A[q-1] + A[q-2] + A[q-3]$
$\vdots$
$A[q] + A[q-1] + \dots + A[1]$

} q sums to create & select max among q numbers

Time $O(n)$

Same in $A[q+1, n]$

We have to look for the maximal sum

$A[q+1]$

$A[q+1] + A[q+2]$

$\vdots$

$A[q+1] + A[q+2] + \dots + A[n]$

} can be calculated in $O(n)$ time

$\Rightarrow$ $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$