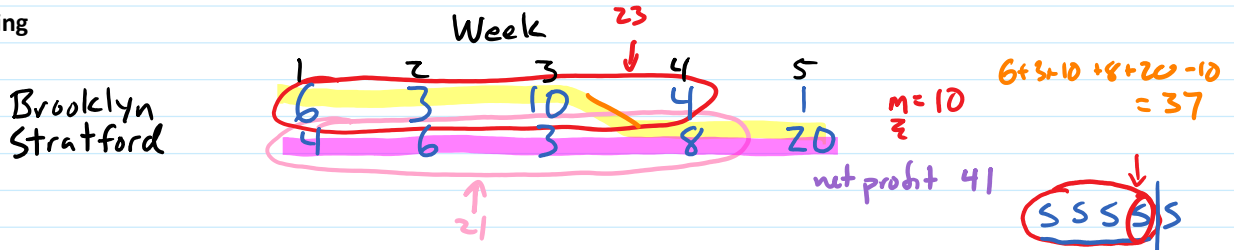
Vendor Scheduling



$OPT_B(i)$ = max profit for weeks $1 \dots i$ ending in B

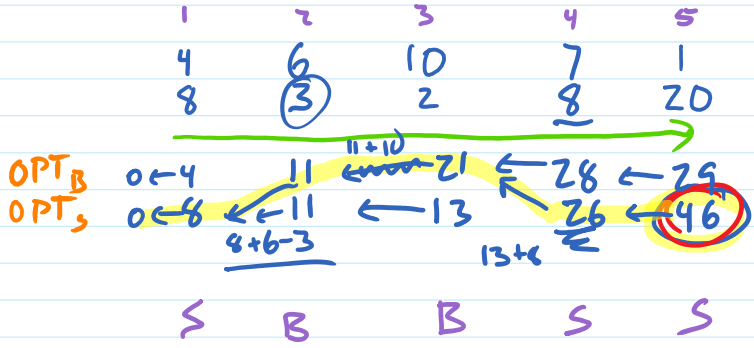
$OPT_S(i)$ = " " " S

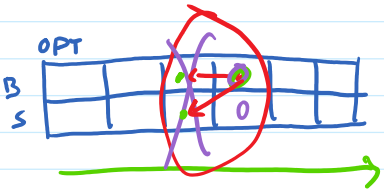
$$OPT_B(i) = \begin{cases} 0 & \text{if } i = 0 \\ \max\left(OPT_B(i-1),\ OPT_S(i-1) - m\right) + B_i & \end{cases}$$

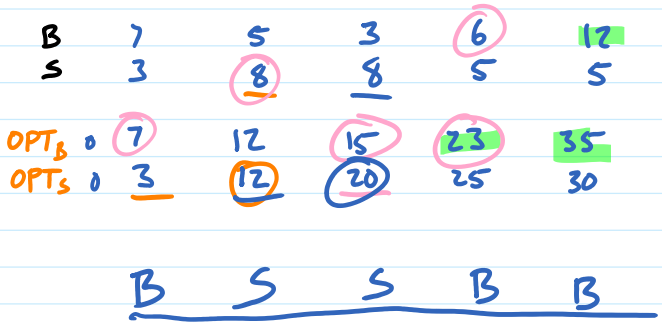
($OPT_B(i-1)$: B in week $i-1$; $OPT_S(i-1)$: S in week $i-1$; B_i: Σ)

$$OPT_S(i) = \begin{cases} 0 & \text{if } i = 0 \\ \max\left(OPT_S(i-1),\ OPT_B(i-1) - m\right) + S_i & \end{cases}$$

($OPT_S(i-1)$: S in week $i-1$; $OPT_B(i-1)$: B in week $i-1$; S_i: Σ)







$OPT_S(2) = OPT_B(1) + S_2 - M$

table has $2n$ entries, $O(1)$ per entry $\Rightarrow \Theta(n)$ total

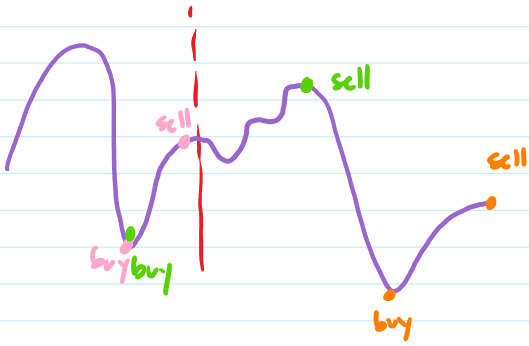
```
S = array 1...n of NIL
S[n] = B if OPT_B[n] > OPT_S[n] else S
for i = n to 2
    if S(i) = B
        if OPT_B(i) = OPT_B(i-1) + B_i
            S[i-1] = B
        else
            S[i-1] = S
    else
        .
        .
        .
```

know closing stock price over last n days

find best days to buy and then sell

one buy, one sell
buy before sell

→ best plan for 1st $\frac{n}{2}$ days
→ best plan for last $\frac{n}{2}$ days
→ best plan for across 1st / last
buy sell



opt_profit

if n=2 then return (min over 2 days, max over 2 days, buy/sell if 1st < 2nd else None)

split into smaller subproblems
solve each subproblem
combine results

else

min_L, max_L, opt_L = opt_profit(1st half)

min_R, max_R, opt_R = opt_profit(last half)

return (min(min_L, min_R), max(max_L, max_R), max(opt_L, opt_R, $max_R - min_L$))

divide-and-conquer

Mergesort

```
MERGE-SORT ( A )
    if len(A) < 2
        return a copy of A
    else
        L ← 1st half of A      ] divide into subproblems
        R ← rest of A          ]
        L ← MERGESORT(L)       ] solve subproblems
        R ← MERGESORT(R)       ]
        return MERGE(L,R)      combine results
```

key comparisons

$T(n)$ = worst-case steps to sort n items

$$T(n) = T(\tfrac{n}{2}) + T(\tfrac{n}{2}) + n-1$$

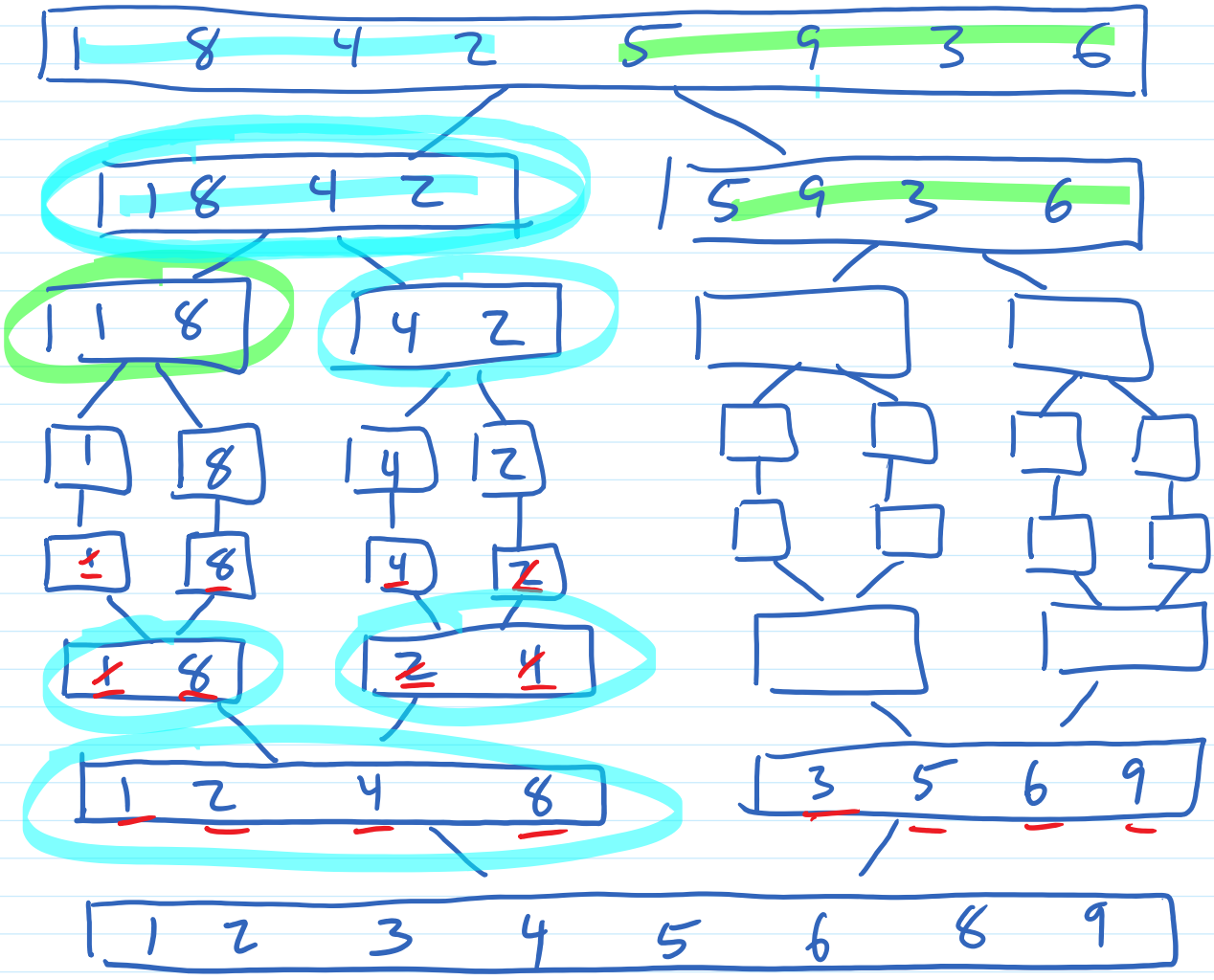
$$= 2 \cdot T(\tfrac{n}{2}) + n-1$$

2 = # subproblems; $\frac{n}{2}$ = size of subproblems; $n-1$ = work to divide/combine

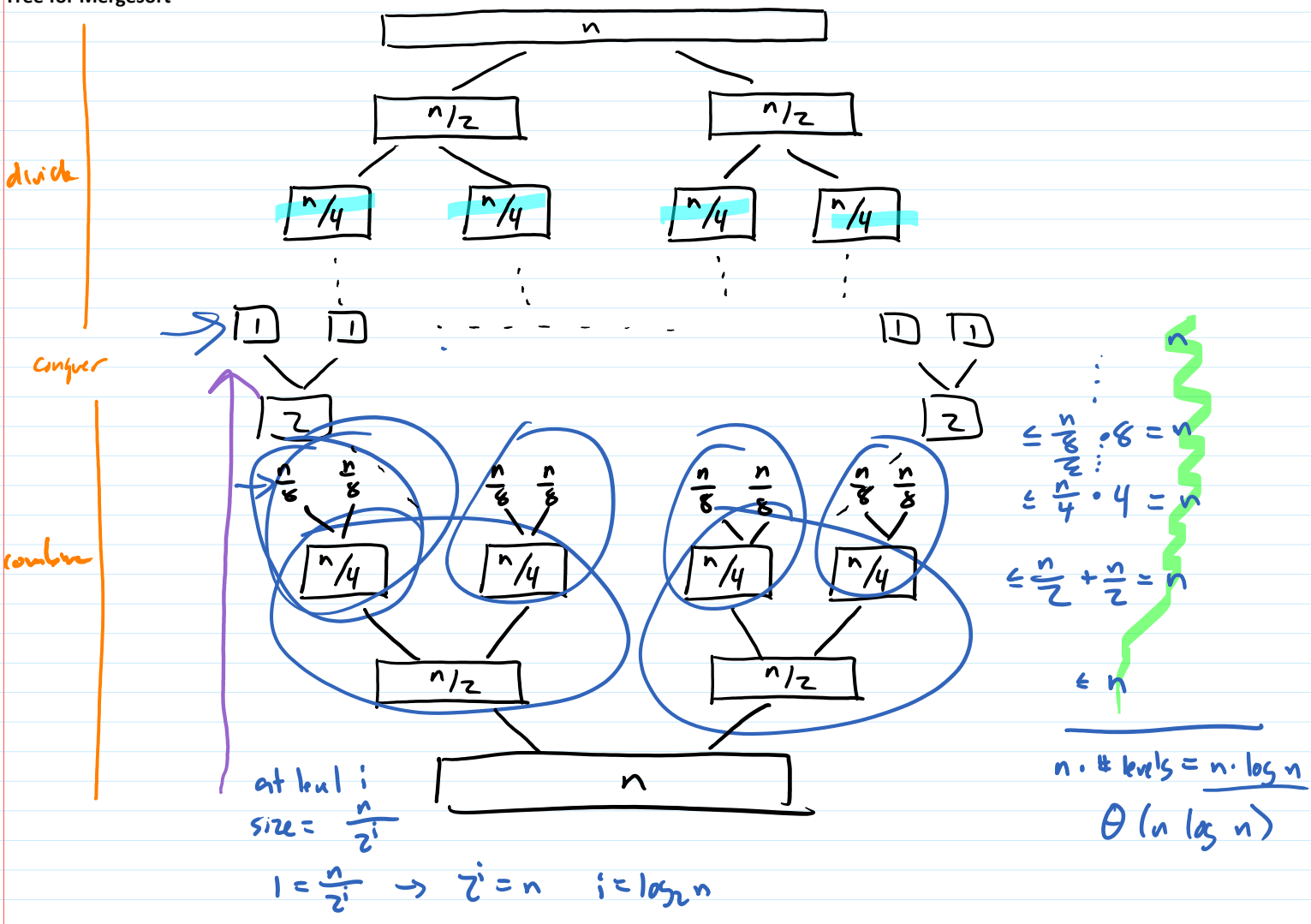
```
MERGE(L,R)
    A ← empty list
    i ← 0                      start at beginning
    j ← 0
    while i < len(L) and j < len(R)
        if L[i] ≤ R[j]         is current on left smaller?
            append L[i] to A   add that to merged list
            i ← i+1            move to next in left list
        else
            append R[j] to A   ]
            j ← j+1            ]
    append L[i ... len(L)-1] to A   ] add what remains on
    append R[j ... len(R)-1] to A   ] the nonempty list
    return A
```

```
QUICKSORT*(A)
 if len(A) ≥ 2
   find median
   split A into <, =, > median
   for each part P, QUICKSORT*(P)
   return sorted parts concatenated
 else
   return A
```

Mergesort Example



Tree for Mergesort



Suppose $T(1) = c_1, \ldots, T(m) = c_m$; $T(n) = 2 \cdot T(\frac{n}{2}) + cn$ for $n \geq m$

Find c s.t. a) $T(n) \leq c \cdot n \log_2 n$ for all base cases
b) divide/combine steps work in $\leq cn$ time

Suppose $n \geq m$ and $T(k) \leq c \cdot k \log_2 k$ for $k = 1, \ldots, n-1$

Then $T(n) = 2 \cdot T(\frac{n}{2}) + c \cdot n$

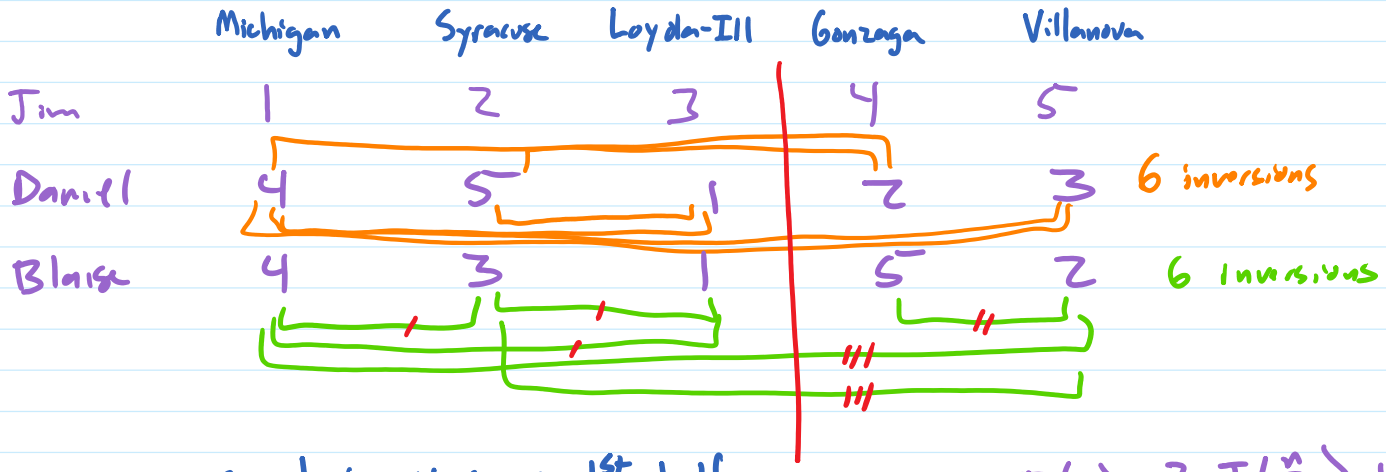
$\leq 2 \cdot c \frac{n}{2} \log_2 \frac{n}{2} + cn$

$= cn(\log_2 n - 1) + cn$

$= cn \log_2 n - cn + cn$

Counting Inversions

Rank the following



count inversions in 1st half
count inversions in 2nd half
→ count inversions between halves
brute force
all pairs of something in 1st half, something in 2nd

$\frac{n}{2}$ $\frac{n}{2}$

$\frac{n^2}{4}$ pairs

$T(n) = 2 \cdot T(\frac{n}{2}) + \frac{n^2}{4}$

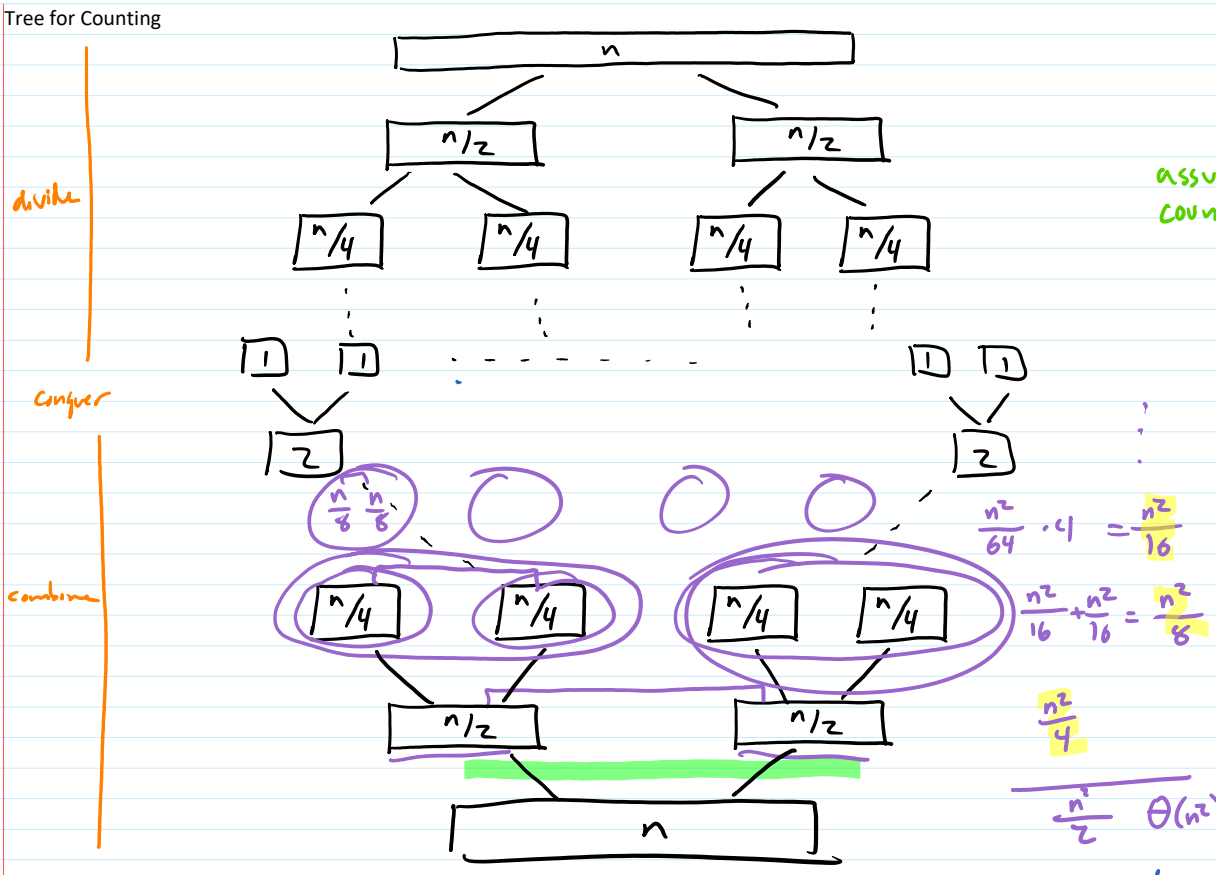

```
COUNT_INVERSIONS(A)
    if len(A)<2 return 0
    L ← 1st half of A
    R ← 2nd half of A

    countL ← COUNT(L)
    countR ← COUNT(R)

    countM ← COUNT-ACROSS(L,R)

    return    countL + countR + countM
```

Tree for Counting



assuming straightforward count inversions between L, R

$\frac{n^2}{64} \cdot 4 = \frac{n^2}{16}$

$\frac{n^2}{16} + \frac{n^2}{16} = \frac{n^2}{8}$

$\frac{n^2}{4}$

$\frac{n^2}{2}$ $\Theta(n^2)$

$$T(n) \leq \sum_{i=0}^{\log_2 n - 1} \left(\frac{1}{2}\right)^i \cdot cn^2$$

$$\leq cn^2 \cdot \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i$$

Counting Inversions

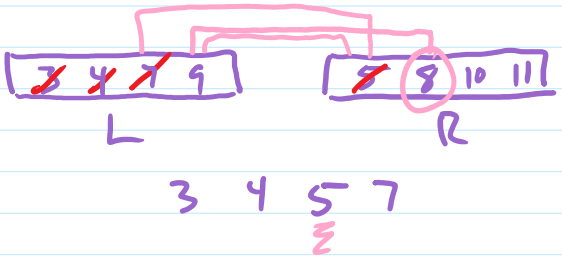
```
COUNT_AND_SORT(A)

    L ← 1st half of A
    R ← 2nd half of A

    L, countL ← COUNT_AND_SORT(L)
    R, countR ← COUNT_AND_SORT(R)

    M, countM ← MERGE_AND_COUNT(L,R)

    return M, countL + countR + countM
```



3 4 5 7

$T(n) = T(\frac{n}{2}) + \Theta(n)$

$T(n)$ is $\Theta(n \log n)$

```
MERGE_AND_COUNT(L,R)
    count ← 0
    A ← empty list
    i ← 0
    j ← 0
    while i < len(L) and j < len(R)
        if L[i] ≤ R[j]
            append L[i] to A
            i ← i+1
        else
            count ← count + len(L) - i
            append R[j] to A
            j ← j+1

    append L[i ... len(L)-1] to A
    append R[j ... len(R)-1] to A

    return A
```