Dynamic Programming.

Edit Distance.

FOOD ⟶ MONEY

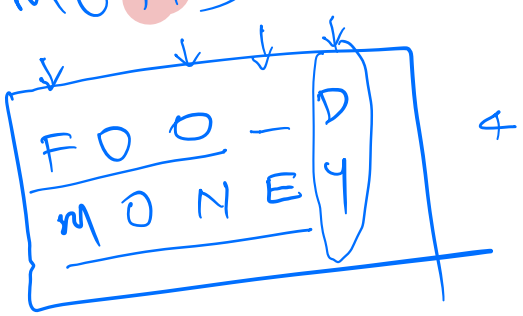
FOOD ⟶ MOOD

↓

MONED ⟵ MOND

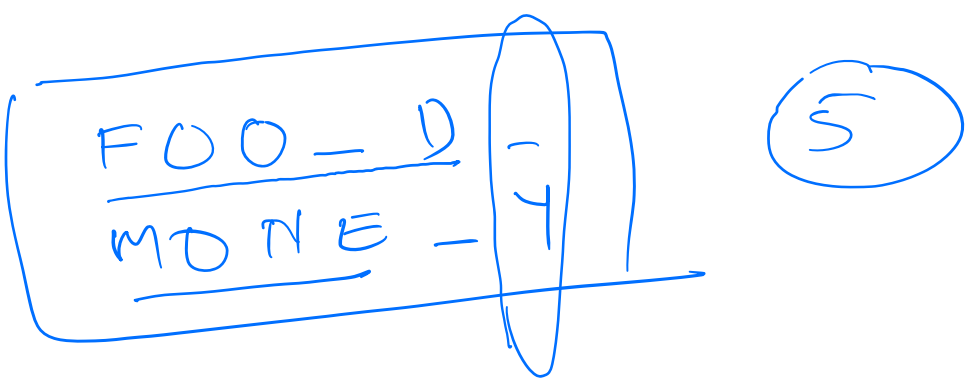
↓

MONEY



Input : $A[1 \ldots m]$, $B[1 \ldots n]$

Output : min # of edits to convert A into B



FOOD — MONEY.

A[1 . . . m]
B[1 . . . n]

$$Edit(m,n) = \begin{cases} \min \begin{cases} Edit(m,n-1)+1 \\ Edit(m-1,n)+1 \\ Edit(m-1,n-1) + [A[m] \neq B[n]] \end{cases} \end{cases}$$

Edit(i, j) = # of edit operations required to convert A[1 . . . i] to B[1 . . j]

$$Edit(i,j) \geq \begin{cases} i & \text{if } j=0 \\ j & \text{if } i=0 \end{cases}$$

$$\text{Edit}(i,j) \overset{(5)}{=} \begin{cases} \min \begin{cases} \text{Edit}(i-1,j)+1 \\ \text{Edit}(i,j-1)+1 \\ \text{Edit}(i-1,j-1)+ \\ \quad [A[i] \neq B[j]] \end{cases} & \text{o.w} \end{cases}$$

$$[A[i] \neq B[j]] = \begin{cases} 0 & \text{if } A[i] = B[j] \\ 1 & \text{if } A[i] \neq B[j] \end{cases}$$

(6) $$[P] = \begin{cases} 0 & \text{if } P \text{ is false} \\ 1 & \text{if } P \text{ is true} \end{cases}$$

```
EditDistance(A[1..m], B[1..n]):
    for j ← 0 to n
        Edit[0, j] ← j

    for i ← 1 to m
        Edit[i, 0] ← i
        for j ← 1 to n
            ins ← Edit[i, j − 1] + 1
            del ← Edit[i − 1, j] + 1
            if A[i] = B[j]
                rep ← Edit[i − 1, j − 1]
            else
                rep ← Edit[i − 1, j − 1] + 1
            Edit[i, j] ← min {ins, del, rep}

    return Edit[m, n]
```

DS: Two dimensional array.

Running time = $O(mn)$.

Lower bounds.

SORTING:

There are algorithms for SORTING with worst-case running time $O(n \log n)$.

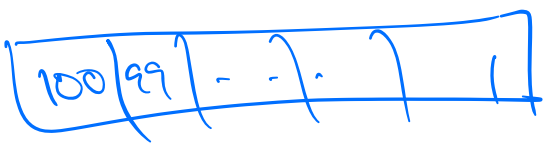
SORTING cannot be solved in time better than $O(n \log n)$

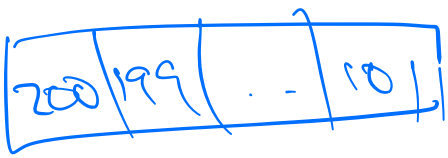
Any comparison based

[illegible] SORTING

algorithm for sorting uses at least $\Omega(n \log n)$ comparison.

Input





Adversary arguments.

Algorithm

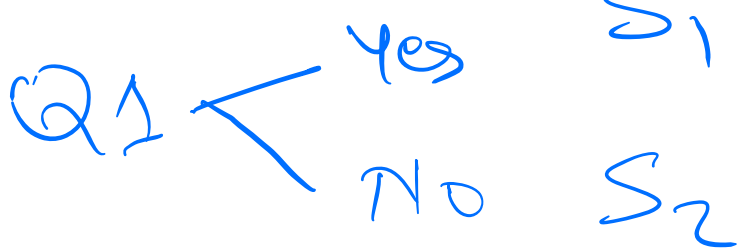
Player 1

Player 2

Guess a no. b/w 1, 2 . . . 100

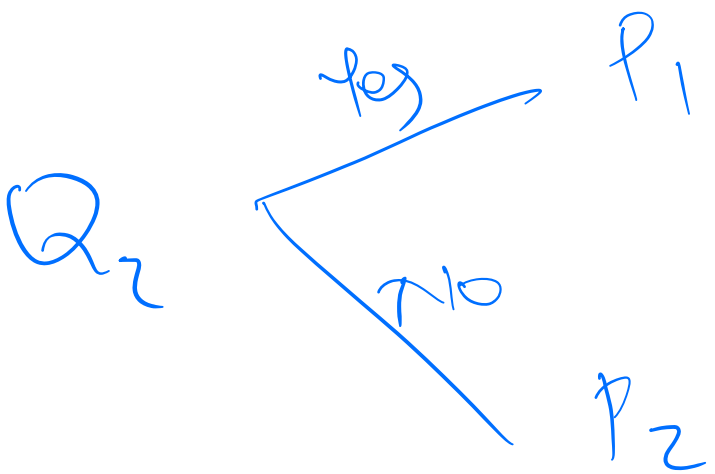
Yes/No questions to player 2.

$S = \{1 \; - \; - \; - \; - \; 100\}$



$S_1 \cup S_2 = S$

Assume $|S_2| \geq |S_1|$



$P_1 \cup P_2 = S$

$\lceil \log_2 100 \rceil$

Search in a sorted array.

I/P: A sorted array $A[1 \ldots n]$ and x

Qm: Is $x \in A[1 \ldots n]$.

Can we do $< \log n - 1$?

For all comparison-based algorithms there exist an input for which we need at least $\log_2 n$ comparisons

Is $x < A[i]$

$i < n/2$

SORTING :-

For any comparison based algorithm for sorting, there is an input for which the algorithm takes

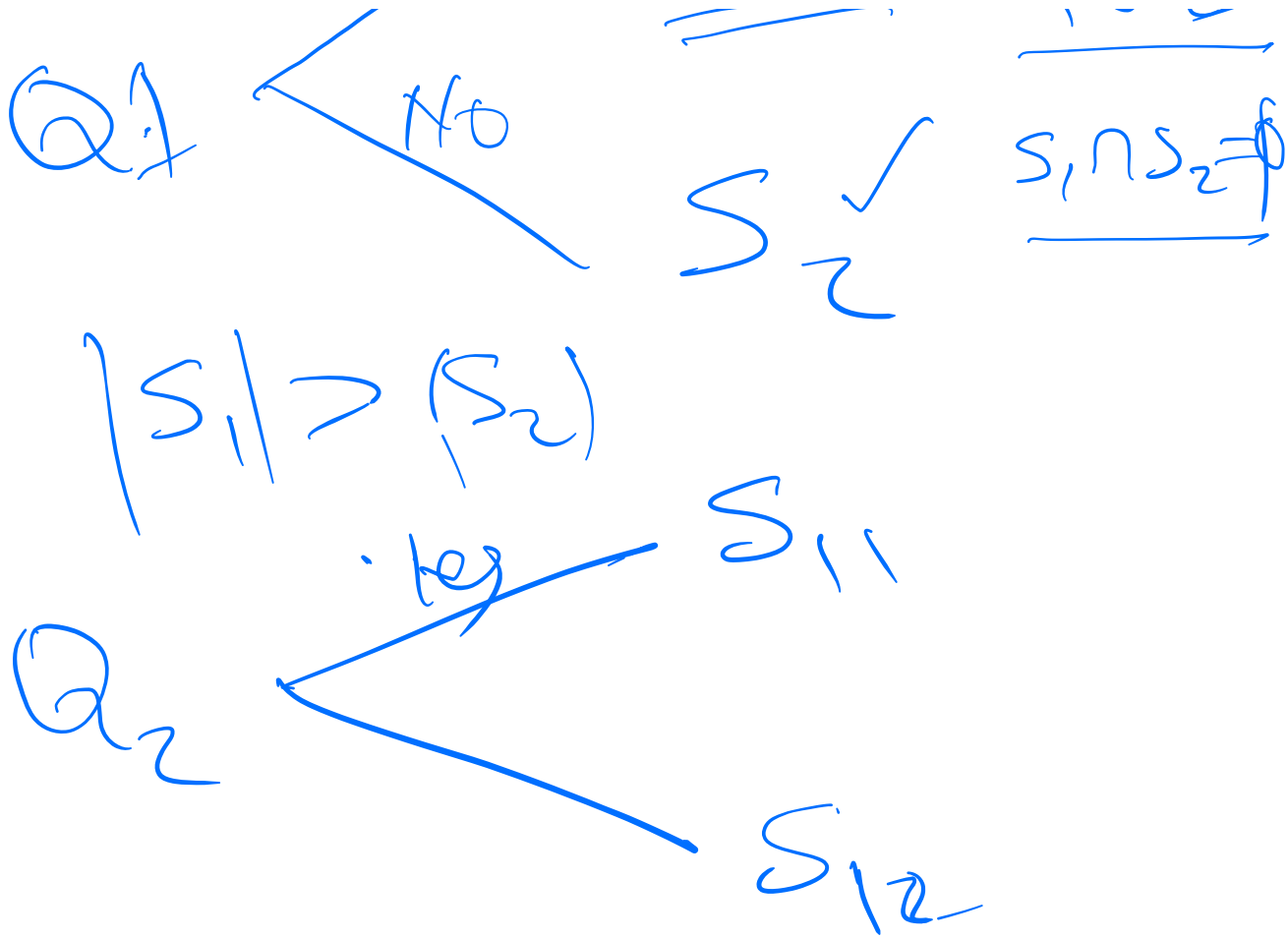
at least $\Omega(n \log n)$ comparisons.

$A[i] < A[j]$ ✓

$S = \{$ All possible permutations of $1 \ldots n$

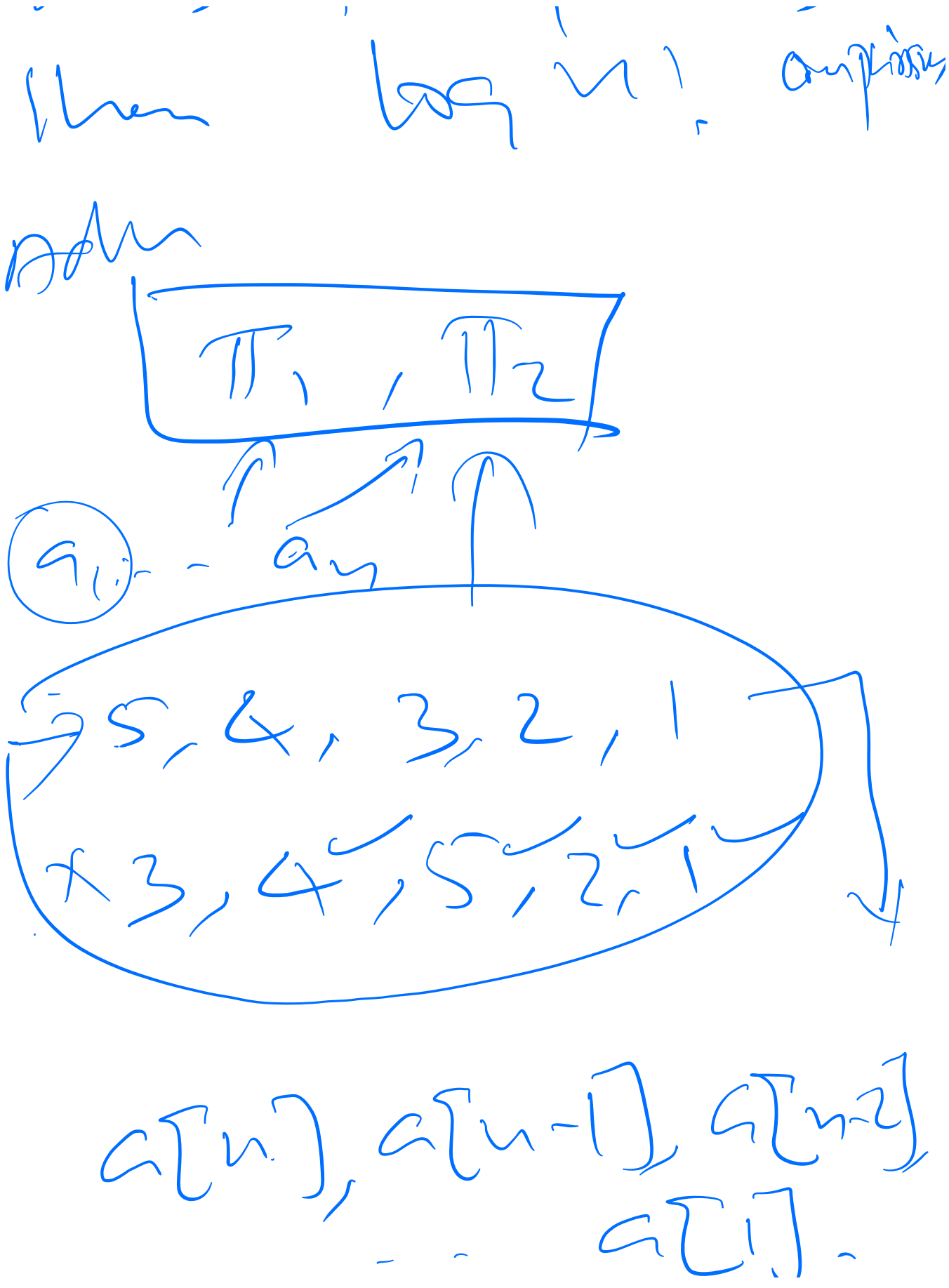
$a_1, a_2 \ldots a_n$

Yes S_1 ✓ $\quad S_1 \cup S_2 = S$



$$\log_2 n! = \Omega(n \log n)$$

suppose algorithm does strictly less


π_1, π_2
$a_1, \ldots, a_n$
5, 4, 3, 2, 1
3, 4, 5, 2, 1
a[n], a[n-1], a[n-2]
a[1].

$$c n \log n \leq \log n! \leq n \log n$$

$$1 \cdot 2 \cdots n!$$

$$\geq n/2 \cdot n/2 \cdot n/2 \cdots n/2$$

$$\geq (n/2)^{n/2}$$

$$\log n!$$

$$\geq n/2 \log n/2$$