Assumption

For simplicity, assume that $q[1..n]$ is sorted in ascending order.
Subtask 1: Choose between 1 3 2 and 2 1 3.

Subtask 2: Try all permutations.

Subtask 3: Do a bitmask dp.

Consider $dp[mask][last]$.

$mask$: a bitmask representing the performed acts.

$last$: the last performed act.

Easy $O(n)$ transition.

Need to memorize the "chosen" previous state, for answer retrieval.

Time complexity: $O(2^n n^2)$.

Subtask 4: Output pattern 1, $n$, 2, $(n - 1)$, ...

(Exercise: Prove that it is one of the optimal solutions.)
Easy Partials

- Subtask 1: Choose between 1 3 2 and 2 1 3.
- Subtask 2: Try all permutations. $O(n! \times n)$

Consider $dp[\text{mask}][\text{last}]$.

- $\text{mask}$: a bitmask representing the performed acts.
- $\text{last}$: the last performed act.

Easy $O(n)$ transition.

Need to memorize the "chosen" previous state, for answer retrieval.

Time complexity: $O(2^n n^2)$.

Subtask 4: Output pattern 1, $n$, 2, $(n-1)$, ...

(Exercise: Prove that it is one of the optimal solutions.)
Easy Partials

- **Subtask 1:** Choose between $1 \ 3 \ 2$ and $2 \ 1 \ 3$.
- **Subtask 2:** Try all permutations. $O(n! \times n)$
- **Subtask 3:** Do a bitmask dp.
  - Consider $dp[mask][last]$.
  - $mask$: a bitmask representing the performed acts.
  - $last$: the last performed act.
  - Easy $O(n)$ transition.
  - Need to memorize the “chosen” previous state, for answer retrieval.
  - Time complexity: $O(2^n n^2)$.
Subtask 1: Choose between 1 3 2 and 2 1 3.

Subtask 2: Try all permutations. $O(n! \times n)$

Subtask 3: Do a bitmask dp.
  - Consider $dp[mask][last]$.
  - $mask$: a bitmask representing the performed acts.
  - $last$: the last performed act.
  - Easy $O(n)$ transition.
  - Need to memorize the “chosen” previous state, for answer retrieval.
  - Time complexity: $O(2^n n^2)$.

Subtask 4: Output pattern 1, $n$, 2, $(n - 1)$, ....
  (Exercise: Prove that it is one of the optimal solutions.)
“Local” Optimality Criteria

Target: reduce the number of candidate solutions (permutations \( p[1..n] \)) we need to consider.

**Observation 1**

If \( p[i] < p[i + 1] < p[i + 2] \) for some \( i \), we can ignore the permutation.
Observation 2

If \(p[i] > p[i + 1] > p[i + 2] > p[i + 3]\) for some \(i\), we can ignore the permutation.
Observation 3

If there exists $i + 1 < j$ such that $p[i] < p[j] < p[i + 1]$ or $p[j] < p[i] < p[j + 1]$, we can ignore the permutation.
A Better Solution

- We can assume that, if $p[i] < p[i + 1]$, then it is one of the pairs $(1, n), (2, n - 1), \ldots$ (thanks to Observation 3).
- This already allows for a better solution.
- $dp[mask][pos]$: $mask$ is on “used” pairs, not individual acts.
- This passes subtask 5. Time complexity: $O(2^{n \cdot 2} \times poly(n))$
Observation 4

If we fix where to put a pair, it is easy to choose the best way to put pairs. For example, $n = 6$, $w[] = \{1, 5, 2, 3, 1\}$ (bold = want to insert a pair). Then we shall put (1, 6) to 5 and (2, 5) to 1.

- Exhaust all possible ways of putting pairs.
- Criteria:
  - Consecutive indices cannot be both chosen.
  - Distance between two adjacent chosen indices cannot exceed 3.
- Turns out there are not many ways of putting pairs (fewer than $10^6$ for $n = 50$. Exercise: derive a recurrence formula.).
- This passes subtask 6. Time complexity: $O((1.3247...)^n \times n)$. 
“Exhaustion” again!?

- We exhaust whether $4, 7, 10, \ldots, 3k + 1, \ldots$ should be given a pair.
- For the rest, it is easy to determine whether a pair should be given.
- Take $5, 6$ as example.
  - If 4 and 7 are chosen, choose neither.
  - If 4 is chosen, choose 6.
  - If 7 is chosen, choose 5.
  - If neither is chosen, compare $w[5]$ and $w[6]$, and choose the better one.
- This gets 100 points. Time complexity: $O(2^{\frac{n}{3}} \times n)$. 

Questions?

Can this task be solved in polynomial time? If yes, how? If no, why not?
Questions?
If no, then I have one for you.

**Challenge**

Can this task be solved in polynomial time?
If yes, how?
If no, why not?