📓 Array Partition Problem Documentation

1. Problem Statement

Given an integer array nums of 2n integers, group these integers into n pairs $(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)$ such that the sum of min(ai, bi) for all i from 1 to n is maximized. Return the maximized sum.

Constraints:

- $1 \leq n \leq 10^4$
- nums.length = 2 * n
- $-10^4 \leq nums[i] \leq 10^4$

2. Intuition

To maximize the sum of the minimum values in each pair, we should try to keep smaller numbers from being wasted. Pairing elements cleverly is the key — sorting helps achieve this by organizing numbers so that smaller numbers are matched with slightly larger ones, ensuring we don't lose their contribution to the total.

3. Key Observations

- Sorting allows natural grouping into pairs.
- Pairing nums[i] and nums[i+1] (where i is even) ensures we get the smaller of every adjacent pair.
- Any other combination can result in lower min(a, b) values.

4. Approach

- Step 1: Sort the array.
- Step 2: Iterate through the sorted list, jumping two steps at a time.
- Step 3: Sum every even-indexed element, which represents the minimum in each pair.

5. Edge Cases

Edge Case	Description
Smallest input	Array with only two elements
All same values	All elements are the same
Negative values	All or some elements are negative
Mixed values	Positive, negative, and zero included

6. Complexity Analysis

 Time Complexity

- Sorting: O(n log n)
- Iteration for sum: O(n)
- Total: O(n log n)

 Space Complexity

- If in-place sorting is allowed: O(1)
- Otherwise (depends on language's sort implementation): O(n)

7. Alternative Approaches

Approach	Description	Time	Space
Sorting (used)	Sort and pick even indices	O(n log n)	O(1)
Counting sort	Use frequency array for range $[-10^4, 10^4]$	O(n + k)	O(k)

Where k = 20001 for range -10000 to 10000.

8. Algorithm

- Sort the nums array.
- Initialize sum = 0.
- Loop through the sorted list with step of 2.
- Add each even-indexed element to sum.
- Return sum.

9. Test Cases

Test Case	Input	Output	Explanation
Case 1	[1, 4, 3, 2]	4	Pairs: (1,2), (3,4) → 1+3 = 4
Case 2	[6,2,6,5,1,2]	9	Pairs: (1,2), (2,5), (6,6) → 1+2+6 = 9
Case 3	[-1, -2, -3, -4]	-6	Pairs: (-4,-3), (-2,-1) → -4+-2 = -6
Case 4	[0, 0, 0, 0]	0	All zeros
Case 5	[10000, -10000]	-10000	Only one pair

10. Final Thoughts

- This is a greedy problem, and sorting is the optimal first step.
- The key is realizing that pairing small numbers early ensures they contribute positively.
- For even faster solutions when dealing with bounded integers, counting sort can be used.