1. **Exact Algorithm**

   The input is an array of integers, a. Let $b(k, s)$ denote whether we can pick up from first k integers that sums up to exactly s, or half the sum of all integers in given set. Then we apply dynamic programming to obtain an answer to the decision problem.

Algorithm 1 Partition(a)

```
k ← a.length
s ← 0
for i ← 0 to k-1 do
  s ← s+a[i]
end for
if s is odd then
  return false
else
  s ← s/2
end if
for i ← 1 to s do
  b(0,i) ← false
end for
for i ← 0 to k do
  b(i,0) ← true
end for
for i ← 1 to k do
  for j ← 1 to s do
    if i-a[j-1] ≥ 0 then
      b(i,j) ← b(i,j-1) or b(i-a[j-1],j-1)
    else
      b(i,j) ← b(i,j-1))
    end if
  end for
end for
return b(k,s)
=0
```

2. **2-Partition is in NP.**

   A certificate of this problem is a subset of positive integers that makes up S (or T, does not matter). We can check in polynomial time whether the sum of S equals that of T.

3. **2-Partition is NP-Complete.**

   Now that 2-Partition is in NP, we only need to prove that it is also NP-Hard as follows.

   - **Reduction**

     Given a 2-Subset-Sum instance x with a set of numbers A, we are able to construct a 2-Partition instance x' with B in the way describe below. For each number in A, we add it to B. The target sum s in x' is half the sum of all integers in A.

     The reduction function runs in polynomial time O($\mid A \mid$).

   - **Claim 1: If $x \in$ 2-PARTITION, then $x' \in$ 2-Subset-Sum.**

     If N can be partitioned into two subsets with equal sum, then there exists two subsets of N, each of which sums up to s. If an integer in x belongs to one certain partition, then we choose its related integer in x'. Thus we have a set of integers that sums up to half the total sum of the original N. By construction, this number equals s. Therefore x' is satisfactory.

- **Claim 2: If $x' \in$ 2-Subset-Sum, then $x \in$ 2-PARTITION.**

  If x' is satisfactory, we can pick up a set of integers in A that sums up to s. If an integer in x' is picked, we put its related integer to the first partition in x; otherwise, if it is not picked, we put it to the second one. We realized that both partitions sum up to s. Therefore x is satisfactory.