HQ26

(i) PROB1 can be solved in polynomial time.

algorithm

// A is an array of positive integers
// |A| = n [1 ... n] (|A| = number of elements in A)
// A is 2 valued ie A has only 2 known integers p and q

```
def PROB1 ( A) :
    Ap = [ ] an empty array
    Aq = [ ] an empty array

    for (i = 1 to n)
        if ( A[i] == p)
            add A[i] to array Ap
        else
            add A[i] to array Aq

condn 1 — if ( |Ap| == q && |Aq| == p )
                return True

condn 2 — if ( |Ap| % 2 == 0 && |Aq| % 2 == 0)
                return True

condn 3 — if ((|Ap| % 2 == 0 && |Aq| % 2 != 0) ||
              ( |Ap| % 2 != 0 && |Aq| % 2 == 0))
                return False
```

contd if (|Ap|%.2 != 0 && |Aq| %.2 != 0 &&

p == q)

return True

return False

time complexity

Time complexity of above algorithm is —

$$T(n) = O(n) + O(c)$$

- $T(n)$ ↓ time complexity of PROB1 with array A of n integers
- $O(n)$ ↓ due to the for loop
- $O(c)$ ↓ other constant time operations

$T(n) = O(n)$. which is polynomial time.

PROB1 (A) decides if there is a way to split A into 2 disjoint subsets A1 and A2 with equal sums. Here A contains only 2 & unique elements p and q. First we separate out the 2 unique elements in 2 arrays Ap and Aq such that Ap contains only the p valued elements and Aq has the q value elements. Now we will pick the elements from Ap and Aq and put them in A1 and A2 (in our algorithm we select the elements instead of putting it back to other arrays

A1 and A2) to achieve equal sum. We assume here that picking up 2 elements from any set Ap or Aq means picking 2 elements and putting one to A1 set and the other to A2 set. With this assumption, if we have even number of elements in any set Ap OR Aq then A1 and A2 will have equal number of that element valued p ~~and~~ or q respectively. And if odd number of elements in Ap OR Aq then any one of A1 OR A2 will have one element extra ~~wi~~ valued p or q respectively. ~~Thus~~ So if we have even sized arrays Ap and Aq then A1 and A2 will have equal number of p ~~elem~~ valued elements and q valued elements and thus their sum would be equal as ~~if~~ each p valued element in A1 will have another p valued element in A2 and the same for q valued elements. This case is handled in cond 2 in our algorithm. As explained above if one if Ap and Aq has odd number of elements then A1 or A2 will have one element extra ie if Ap or Aq 's length is odd

then A1 and A2 will have one p and one q valued elements in either A1 or A2. Thus the sum of A1 and A2 can never be equal as each p/q valued element ~~(which ever is odd) must have another term to cancel it and make sum equal.~~ But this condition can be violated only if p and q are we have one extra p and q valued element in A1 ~~and~~ OR A2 which will not have same value and thus sum must be same. This condition can be violated only when p and q have same value then again each of the set A1 and A2 will have extra element with the same value and thus sum would be same. This is handled in condn 4 of our algorithm. If one of Ap or Aq is odd and the other is even then A1 and A2 can never have equal sum as either A1 or A2 will always have one extra element which will not be balanced by the other set. Thus we return False, condn3 of our code. condn1 is a special case where we check $|Ap| \times q == |Aq| \times p$ ie p valued elements are of size q and q valued elements are of size p, then we return True as sum of 2 sets A_1 and A_2 would be same where $A_1 = Ap$ and $A_2 = Aq$. So we have a partition where sum(Ap) = sum(Aq).

(ii)

PROB 2 is NP complete..

To prove PROB2 is NP complete we need to prove it is NP and NP Hard.

Proof that PROB2 is NP.

With the help of verification algorithm we will proof that PROB2 is NP.

Proof P : 2 sets of indices of input array A; which are disjoint → SA1 and SA2.

input instance : an array A of n elements.
A[1..n]

verification algorithm : def verifyPROB2Proof(A, Proof P):

O(n) ——— sum1 = 0
sum2 = 0
for (i = 1 to n)
if (i ∈ SA1)
sum1 = sum1 + A[i]
else
sum2 = sum2 + A[i]
if (| sum1 - sum2 | ≤ 525)
return True
return False

running time

Proof P will require O(n) complexity for chosing the indices from array A which are disjoint and thus it is polynomial time. ~~in order of |A|~~. now as shown in the ~~algorithm~~ algorithm above ~~time~~ time complexity of verify PROB2Proof is $O(n) + O(1) \approx O(n)$.

Lemma 1

If A has a Yes instance then there must be a proof P for which verify PROB2 Proof gives true.

If A has a yes instance then A can be splitted into two disjoint sets A1 and A2 such that $|sum(A1) - sum(A2)| \leq 525$; then according to our proof we have 2 sets of indices which ~~are~~ are disjoint which satisfies the 1st criteria of our problem. Now after adding the values of these indices ~~if A~~ the 2nd condition ie $|sum(A1) - sum(A2)|$ ~~then our algorithm will be return False else True~~ will be checked and will give ≤ 525 and hence will return True, because our two subset SA1 and SA2 are disjoint sets of indices from Array A itself. Thus a proof exist.

Lemma 2

If A has a NO instance then for every P verify PROB2 Proof will return False.

If A has a no instance ie A cannot be broken down to 2 disjoint subsets A1 and A2 s.t $|sum(A1) - sum(A2)| \leq 525$. Our Proof has indices of array A which are disjoint. Then when values of these 2 sets will be added it and then subtracted will give an answer of > 525 and hence return False as there exist no such combination of indices in A to form 2 disjoint subsets as proof such that $|sum(SA1) - sum(SA2)|$ is < 525.

Proof that PROB2 is NP Hard

To prove that PROB2 is NP Hard we need to reduce a known NP Hard problem to PROB2

We will reduce PARTITION Problem to PROB2. PARTITION algorithm is a known NP complete and thus a known NP Hard problem.

reduction algorithm

```
def reduce ( array A ) :
    A' = new empty array A
    for ( i = 1 to n )
        add A[i] * 526 to A'[i]
    return A'
```

explanation

we are taking a new empty array A' and copying all the elements of A[i] * 526 ie element of A[i] multiplied by a constant 526 to A'[i].

A' is my reduced Aarray ie the input instance of PROB 2.

running time

Running time of reduce is — O(n) due to the for loop which is polynomial in time.

Lemma

- we get a yes instance of PARTITION iff we get a Yes instance of PROB2.

If PARTITION returns a yes instance then array A can be broken / split into 2 disjoint subsets A1 and A2 s.t $|sum(A1) - sum(A2)| = 0$. After reduction we get A' which is $A'\{1 \ldots ny = 526 * A\{1 \ldots ny$. Thus as $A\{1 \ldots ny$ could be broken into 2 parts A1 and A2 (disjoint) as mentioned above, here also we have the same set of elements and thus we can split it into 2 disjoint subsets. So in PROB2 we will also have 2 disjoint sets A1' and A2' after taking 526 as common —

$$|526 * sum(A1') - 526 * sum(A2')|$$

[taking 526 as common from all elements of A']

$$= 526 |sum(A1') - sum(A2')|$$

$$= 526 (|sum(A1) - sum(A2)|)$$

[as A1' & A2' will be same as A1 & A2 with 526 taken as common]

$$= 526 (0)$$ [PARTITION always gives $|sum(A1) - sum(A2)| = 0$ and it is our assumed fact in lemma]

$$= 0$$

which is $<= 525$. Thus PROB 2 will give a yes instance.

If PARTITION returns a NO instance then array A cannot be splitted into 2 disjoint subsets A1 and A2 s.t $|sum(A_1) - sum(A_2)| \neq 0$ ie $|sum(A_1) - sum(A_2)| \geq 1$. Now after reduction we get $A'\{1..n\} = 526 * A\{1..n\}$. So if $A\{1..n\}$ could be broken into 2 parts here also in A' in PROB2 we can split A' into 2 disjoint sets A_1' and A_2'. We can write —

$$|sum(A_1') - sum(A_2')|$$

2) $526 \mid sum(A_1') - sum(A_2') \mid$ [taking 526 as common]

2) $526 [|sum(A_1) - sum(A_2)|]$ [as A_1' and A_2' is same as A_1 and A_2 after taking 526 as common]

2) $526 * (\text{min value here can be } 1)$ [as shown above]

2) $526 * (\text{min value})$

Thus $526 \geq 525$ will return a NO instance in PROB2 as well.

(c) we have proved that PROB2 is also NP Hard. and we know that PARTITION is NP Hard. Thus they can be reducible to each other. So if PARTITION can be solved in polynomial time ie if PARTITION is easy then ~~NP Hard~~ PROB2 will also be easy.