DP - exam question

Theory Task T3. / 9 P

You are given an array of n natural numbers $a_1, \ldots, a_n \in \mathbb{N}$, and two natural numbers $A, B \in \mathbb{N}$. You want to determine whether there is a subset $I \subseteq \{1, \ldots, n\}$ satisfying

$$\sum_{i \in I} a_i = A \quad \text{and} \quad \sum_{i \in I} a_i^2 = B.$$

Subset Sum

For example,

- The answer for the input $(a_i)_{i \le n} = [2, 4, 8, 1, 4, 5, 3]$, $A = 8$ and $B = 30$ is *yes* because the set of indices $I = \{1, 4, 6\}$, which corresponds to $(a_i)_{i \le I} = [2, 1, 5]$, yields the *sum* $2 + 1 + 5 = 8$ and the *sum-of-squares* $2^2 + 1^2 + 5^2 = 30$.
- The answer for the input $(a_i)_{i \le n} = [2, 4, 8, 1]$, $A = 6$ and $B = 15$ is *no*.

Provide a *dynamic programming* algorithm that determines whether such a subset I exists. In order to get full points, your algorithm should have an $O(n \cdot A \cdot B)$ runtime. Address the following aspects in your solution:

1) *Definition of the DP table*: What are the dimensions of the table $DP[\ldots]$? What is the meaning of each entry ?
2) *Computation of an entry*: How can an entry be computed from the values of other entries ? Specify the base cases, i.e., the entries that do not depend on others.
3) *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps ?
4) *Extracting the solution*: How can the final solution be extracted once the table has been filled ?
5) *Running time*: What is the running time of your algorithm ? Provide it in Θ-notation in terms of n, A and B, and justify your answer.

Size of the DP table / Number of entries: $[0 \ldots n] \times [0 \ldots A] \times [0 \ldots B]$

Meaning of a table entry:

DP[i][a][b] := true if there is $I \subseteq \{1 \ldots i\}$ s.t. $\sum_{i \in I} a_i = a$ $\sum_{i \in I} a_i^2 = b$

false otherwise

Scheme continues on the next page.

Computation of an entry (initialization and recursion):

DP[0][0][0] = true DP[0][x][y] = false for $x, y \geq 1$

DP[i][a][b] = DP[i-1][a][b] ∨ DP[i-1][a-a_i][b-a_i^2]

Consider every entry out-of-bounds to be false

Order of computation:

We compute with increasing order of i, a, b.

Extracting the result:

If DP[n][A][B] == true then the answer is yes

DP[n][A][B] == false then the answer is no

Running time:

We need to fill n+1 × A+1 × B+1 entries

Computation time for each of them is Θ(1)

Therefore running time is Θ(n·A·B)