1. **Input:** *data*: array of integers

**Input:** *t*: target value

**Input:** S: subset of *data* that sums to *t*

**Output:** Verifies S is a subset sum

1. **Algorithm:** SSD
2. Check *S* ⊆ *data* // O(n)
3. Set *sum* to 0 // O(1)
4. **for** *s* in *S* // n iter. × O(1) = O(n)
5. Add *s* to *sum* // O(1)
6. **end**
7. **if** *sum* is equal to *t* // O(1)
8. **return** true // O(1)
9. **else**
10. **return** false // O(1)
11. **end**

Since SSD runs in polynomial time **O(n + n)**, SSD ∈ NP.

1. **Input:** data: array of integers

**Input:** n: size of data

**Input:** t: target value

**Output:** S: subset of data that sums to t, or ; if no such set exists

1. **Algorithm:** SSReduction
2. **if** ~SSD(data, t) **then** // O(S(n))
3. **return** ∅ // O(1)
4. **end**
5. sub = 0 // O(1)
6. top = n // O(1)
7. **while** sub < top **do** // n iter. × O(S(n))
8. **if** ~SSD(data[1..(top - 1)], t) **then** // O(S(n))

/\* Add data[top] to subset \*/

1. sub = sub + 1 // O(1)
2. Swap data[top] and data[sub] // O(1)
3. **else**

/\* Delete data[top] from array \*/

1. top = top - 1 // O(1)
2. **end**
3. **end**
4. **return** data[1..sub] // O(1)

The worst-case complexity of SS is **O(S(n) + nS(n))** if the worst-case complexity of SSD is **O(S(n))**.