(a) SMALLSUM $\in P$: Consider the following algorithm.

> SS(S,t):
> **for** $i \leftarrow 1,\ldots,n$:
> **if** $x_i \leqslant t$:
> **return** TRUE
> **return** FALSE

The algorithm examines each number once and performs one comparison for each one, in linear time. The total time is therefore $\mathcal{O}(m^2)$, where m is the total bit-size of (S,t).

Moreover, if $x_i \leqslant t$ for any i, then $\{x_i\}$ is a subset whose sum is at most t, and if $x_i > t$ for every i, then no non-empty subset has sum at most t (because every non-empty subset contains at least one x_i).

(b) LARGESUMS $\in P$: Consider the following algorithm.

> LS(S,t):
> **for** $i \leftarrow 1,\ldots,n$:
> **if** $x_i < t$:
> **return** FALSE
> **return** TRUE

The algorithm examines each number once and performs one comparison for each one, in linear time. The total time is therefore $\mathcal{O}(m^2)$, where m is the total bit-size of (S,t).

Moreover, if $x_i < t$ for any i, then $\{x_i\}$ is a subset whose sum is *not* at least t, and if $x_i \geqslant t$ for every i, then every non-empty subset has sum at least t (because every non-empty subset contains at least one x_i).

(c) EXACTSUM $\in NP$: Consider the following verifier.

> ES(S,t,c):
> **if** $c \subseteq S$ **and** $\sum_{x\in c} x = t$:
> **return** TRUE
> **return** FALSE

The verifier checks that its certificate c is a subset of S (time $\mathcal{O}(m^2)$ if we search for each element of c in S), then adds up the elements of c: each addition takes linear time, for total time $\mathcal{O}(m^2)$, where m is the total bit-size of (S,t).

Also, if ES(S,t,c) = TRUE for some c, then c is a subset of S whose sum is exactly t. And if there is some subset c of S whose sum is exactly t, then ES(S,t,c) = TRUE.