# Inductively Defined Sets

- We can define sets inductively/recursively as well.

- *Example* 1: The set $S$ is defined as

  $0 \in S$

  $\forall n \in S, \; n + 1 \in S$

  Note that $S = \mathbf{N}$ (the set of natural numbers)

- *Example* 2: The set $T$ is defined as

  $(0, 3) \in T$

  $\forall (w, x) \in T, \forall (y, z) \in T, \; (w + y, x + z) \in T$

  $\forall (w, x) \in T, \forall (y, z) \in T, \; (w - y, x - z) \in T$

- Note that $T = \{ (0, 3m) \mid m \in \mathbf{Z} \}$

# Comparing two functions

- Which of these two procedures is faster?

```
procedure NAE1(a : array of n reals) : bool
  for i := 1 to n-1
    for j := i+1 to n
      if a[i] ≠ a[j]
        return true
  return false
```

```
procedure NAE2(a : array of n reals) : bool
  for i := 1 to n-1
    if a[i] ≠ a[i+1]
      return true
  return false
```

- Convince yourself that both procedures do the same thing. Suppose NAE1 takes $T_1(n)$ steps and NAE2 takes $T_2(n)$ steps on an array of size $n$

- It is clear that $\forall n \in \mathbf{N}, \; T_1(n) \leq T_2(n)$, so NAE1 is "faster". We will extend this notion of "faster" to something more useful.

# Big O

- Let $f : \mathbf{N} \to \mathbf{N}$ and $g : \mathbf{N} \to \mathbf{N}$ be two functions
- We say that $f$ is $O(g)$ if $\exists m \in \mathbf{N}, \exists c > 1, \forall n \geq m, \; f(n) \leq c \cdot g(n)$
- In other words, $f$ is $O(g)$ if “eventually” $f(n)$ is at most some constant times $g(n)$
- “Eventually” ($\forall n \geq m$) means we don’t care about small values of $n < m$
- “Constant” ($\exists c > 1$) means that the value of $c$ does not depend on $n$
  - Note the order of the quantifiers!
- *Example*: $2n^2$ is $O(n^3)$ because $\exists m = 1, \exists c = 2, \forall n \geq m, 2n^2 \leq c \cdot n^3$
- Note that $m$ and $c$ do not have to be as small as possible – any value that works is enough for the definition to apply.