Is there a set which is not decidable?

\[ \{ \langle p, o \rangle : p \text{ halts on } o \} \text{ is not decidable}. \]

The set of all programs, \( \{ p : p \text{ always returns} \} \) is not decidable.

Def: A set \( A \) is called recursively enumerable (r.e.) if \( \exists \) an algorithm that eventually prints all elements of \( A \).

\[
\begin{align*}
    n &= 0 \\
    \{ \text{while (true) } &\text{ system.out.println (n);}
    \quad n + 1 \} \tag{This shows that \( \{n\} \) is recursively enumerable.}
\end{align*}
\]

- \( A \) is decidable:–

we know: \( A \) is decidable.

means: there exists an algorithm that, given \( n \), checks whether \( n \notin A \).

\[
\begin{align*}
    \text{public static boolean inA (int n) } &\quad \{ \quad \\
    \text{we want to prove:– \( A \) is decidable.} \\
    \quad n \rightarrow \square \quad \text{we want to have:} \\
    \quad \text{an algorithm inCompToA(n)}: \\
    \quad \text{inComplementToA(n)} = \begin{cases} 
    \text{true} & \text{if } n \notin A \\
    \text{false} & \text{otherwise}
    \end{cases} \\
    \quad \text{public static boolean inCompToA (int n) } &\quad \{ \text{return } ! \text{inA(n)}; \} \\
    \quad n \in - A \iff n \notin A \\
    \end{align*}
\]

A decidable \( \iff \exists \) algorithm that, given \( n \), check whether \( n \in A \).

A r.e. \( \iff \exists \) algorithm that eventually prints all elements of \( A \).

\( \text{if semi-decidable.} \)

\( \text{ } \)

\( \text{N is r.e.} \)
2. \( \emptyset \) is r.e.

3. Every decidable set r.e.
   
   Let's say, \( A \) is decidable.
   
   ```java
   public static boolean inA(int n)
   n = 0;
   while (true) {
     if (inA(n))
       System.out.println(n);
     n++;
   }
   ```
   
   Thus we can print all n \( \in A \).

   Another way is by flowchart:

4. Every finite set is r.e. \( \rightarrow \) since finite set is decidable.

5. Every co-finite (completely not to finite) set is r.e. \( \rightarrow \) since (finite set) is decidable.

6. If \( A \) is r.e. \& \( B \) is r.e. then \( A \cup B \) is r.e.?

   We have \( n \in A \) \& \( n \in B \).

   **Proof:**
   
   - Run \( inA \) for 1 hour
   - Run \( inB \) for 1 hour
   
   **time-sharing algorithm:**
   
   Run \( inA \) for 1 more hr
   
   Run \( inB \) for 1 more hr

   Thus, we print...

   \( A \cup B \).

7. If \( A \) is r.e \& \( B \) is r.e. then \( A \cap B \) is r.e.?

   Run \( inA \) for 1 hr. (make partial list)
   
   Run \( inB \) for 1 hr. (in)

   Print all common elements.

   Run \( inA \) for 1 more hr. (make longer list)
   
   Run \( inB \) for 1 more hr. (in)

   \( n \in A \cap B \)

   Print all common elements.

   \( t_A \) \& \( t_B \) \& time to appear in \( A \)

   \( t_A \) \& \( t_B \) \& time to appear in \( B \)

   Thus, appearance time in \( A \cap B \)

   is always \( 2 \times \max (t_A, t_B) \)

   Here: \( 2 \times 11 = 22 \) ms.
If $A$ is r.e. $\overline{A}$ is r.e., then $A$ is decidable.

Decidable means checks if any $n \in$ set.

Stops when $n$ appears in one of the lists.

If $A$ is r.e., is the $-A$ is r.e.? This is wrong hypothesis.

We had an example of a set which is not decidable.

$H = \mathcal{E}(p, d): p \text{ halts on } d$

All rational numbers

irrespective of the order, we can

$\frac{1}{2} - \frac{1}{2} - \frac{1}{3} - \frac{1}{4} - \frac{1}{5}$

print all $n \in \mathbb{N}$

run program #0 and #1 for 1 hr on data 0, 1.

if halts print $(p, d)$.

run programs 0, 1, and 2 for 2 hrs on data 0, 1, 2.

if halts print $(p, d)$.

run programs 0, 1, 2, 3 for 3 hr on data 0, 1, 2, 3.

if halts print $(p, d)$.

Thus half-checker is r.e.

Thus there exist a set which is r.e., but not decidable.

Thus there exist $A$ which is r.e., but $-A$ is not r.e.

Is every set r.e.?

No, there is a set which is not r.e.

halft-checker
Semi-decidable:

\[ \text{in} A(n) = \begin{cases} \text{true} & \text{if } n \in A \\ \text{false} & \text{if } n \notin A \end{cases} \]

If \( A \) is r.e., then \( A \) is semi-decidable.

Suppose we have an algorithm that prints all elements of \( A \). It waits for every hour checks whether \( n \) was printed.

If \( A \) is semi-decidable, run \( \text{in} A \) on 0, 1, 2 for 1hr.

If it halts we print the corr #

run \( \text{in} A \) on 0, 1, 2 for 2hr.

If it halts print

...