¿Qué es un problema computacional?

De manera muy informal

Un problema $\mathcal{P}$ es una relación (función) entre un conjunto $\mathcal{I}$ de instancias y un conjunto $\mathcal{S}$ de soluciones.

- Si $(i, s) \in \mathcal{P}$ diremos que s es una solución para la instancia i.
- Usualmente se define $\mathcal{P}$ por comprensión: se describe el conjunto de instancias y luego se describen las soluciones para cada instancia.

Ejemplos de problemas

- Por comprensión: $\mathcal{P} = \{((a, b), x) : ax + b = 0 \text{ con } a, b, x \in \mathcal{R}\}$.
- Por extensión: $\mathcal{P} = \{(3, 1), (4, 1), (5, 9), (2, 6), (5, 4)\}$.

Tipos de problemas

De manera muy general hay:

- Problemas triviales.
- Problemas de decisión.
- Problemas de búsqueda.
- Problemas de conteo.
- Problemas de optimización.

Problema de Collatz

Función collatz(n)

- Si $n = 1$ termina.
- Si n es par entonces llama a collatz($n/2$).
- Llama a collatz($3n + 1$).

Enunciado

¿Terminará el proceso anterior para un valor dado de n? (Erdös 500 US)

$$10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1.$$

¿Qué se sabe?

Hasta 2012 se sabe[a] que termina para toda $n \leq 5 \times 2^{60}$.

[a] `http://www.ieeta.pt/~tos/3x+1.html`

¿Qué es un algoritmo?

De manera muy informal

Un algoritmo es un método efectivo expresado como una lista finita de instrucciones bien definidas. Comenzando desde un estado inicial y con una entrada (posiblemente vacía), las instrucciones describen un proceso determinista que, al ejecutarse, procederá a través de una cantidad finita de estados consecutivos, eventualmente generando una salida (posiblemente vacía) y terminando en un estado final.

Visto como función

Un algoritmo $\mathcal{A}$ es una función de su conjunto de entradas $\mathcal{I}$ a su conjunto de salidas $\mathcal{S}$. Si se ejecuta $\mathcal{A}$ con la entrada $e \in \mathcal{I}$ y al terminar produce la salida $s \in \mathcal{S}$, diremos que $\mathcal{A}(e) = s$.

Definiciones de algoritmo

Cronológicamente

- 1870: Máquina lógica de Jevon.
- 1881: Premisas de Venn.
- 1943: Cálculos de Kleene.
- 1952: Máquinas de Turing.
- 1954: Caracterización de Markov.
- 1963: Caracterización de Gödel.
- 1967: Caracterización de Minsky.
- 1968: Caracterización de Knuth.
- 1972: Caracterización de Stone.
- 1995: Caracterización de Soare.
- 2000: Caracterización de Gurevich.
- 2006: Descripciones de Sipser.

¿Cuándo no es un algoritmo?

Si viola cualquier condición

- Si la lista de instrucciones no es finita.
- Si alguna instrucción no está bien definida:
  - Indefinida.
  - Ambigua.
  - No efectiva.
  - Aleatoria.
  - No determinista.
- Estado inicial ambiguo o indefinido.
- Estado siguiente ambiguo o indefinido.
- No termina para alguna entrada.

¿Cómo expresamos algoritmos?

Entre otros:

- Lenguaje natural.
- Seudocódigo.
- Diagramas de flujo.
- Lenguajes de programación.
- Tablas de control.
- Máquinas de Turing.

Observación

Todos estos sistemas permiten expresar cosas que no son algoritmos (por ejemplo, procedimientos no deterministas o que nunca terminan).

Problemas y algoritmos

Algoritmos que resuelven problemas

Diremos que un algoritmo $\mathcal{A} : \mathcal{I} \to \mathcal{S}$ resuelve un problema $\mathcal{P} : \mathcal{I} \to \mathcal{S}$ si $\mathcal{A}(e) = \mathcal{P}(e)$ para toda $e \in \mathcal{I}$.

Observación

En realidad basta que el dominio de $\mathcal{A}$ incluya al dominio de $\mathcal{P}$ y que el contradominio de $\mathcal{A}$ incluya a la imagen de $\mathcal{P}$. Es decir, un algoritmo $\mathcal{A}$ puede resolver un problema más general que $\mathcal{P}$.

Ejemplo de algoritmo

Función euclides(A, B)

- Mientras $B \neq 0$:
  - $C \leftarrow B$.
  - $B \leftarrow A \mod B$.
  - $A \leftarrow C$.
- Regresa A.

Máximo común divisor

No es difícil ver que euclides$(A, B) = mcd(A, B)$.