

DATOS MASIVOS I

UNIDAD II MODELO DE MAPEO Y REDUCCIÓN

TEORÍA DE LA COMPLEJIDAD

Costo de Algoritmo

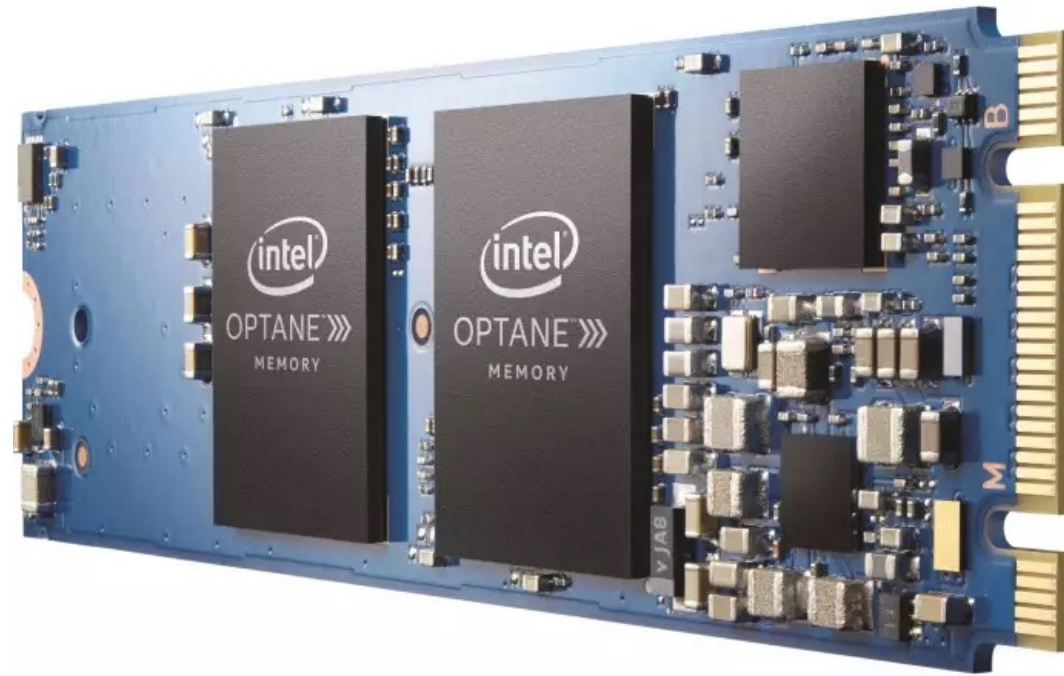
De manera general, un algoritmo puede ser descrito en los siguientes costos.



- ✓ Tiempo: ¿cuánto tiempo lleva en ejecutar una tarea?

Costo de Algoritmo

De manera general, un algoritmo puede ser descrito en los siguientes costos.



✓ Espacio: ¿cuánta memoria usa?

Costo de Algoritmo

De manera general, un algoritmo puede ser descrito en los siguientes costos.



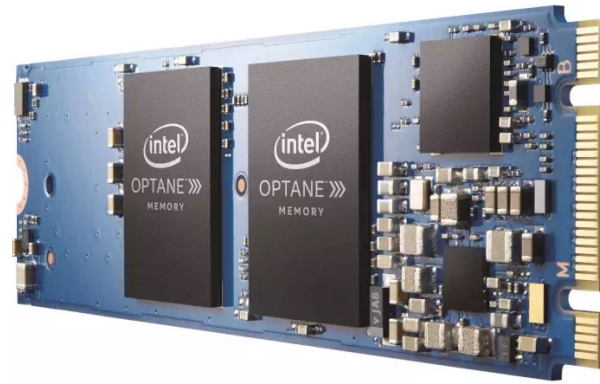
- ✓ Energía: ¿cuánta energía usa?

Costo de Algoritmo

De manera general, un algoritmo puede ser descrito en los siguientes costos.



- ✓ Tiempo: ¿cuánto tiempo lleva en ejecutar una tarea?



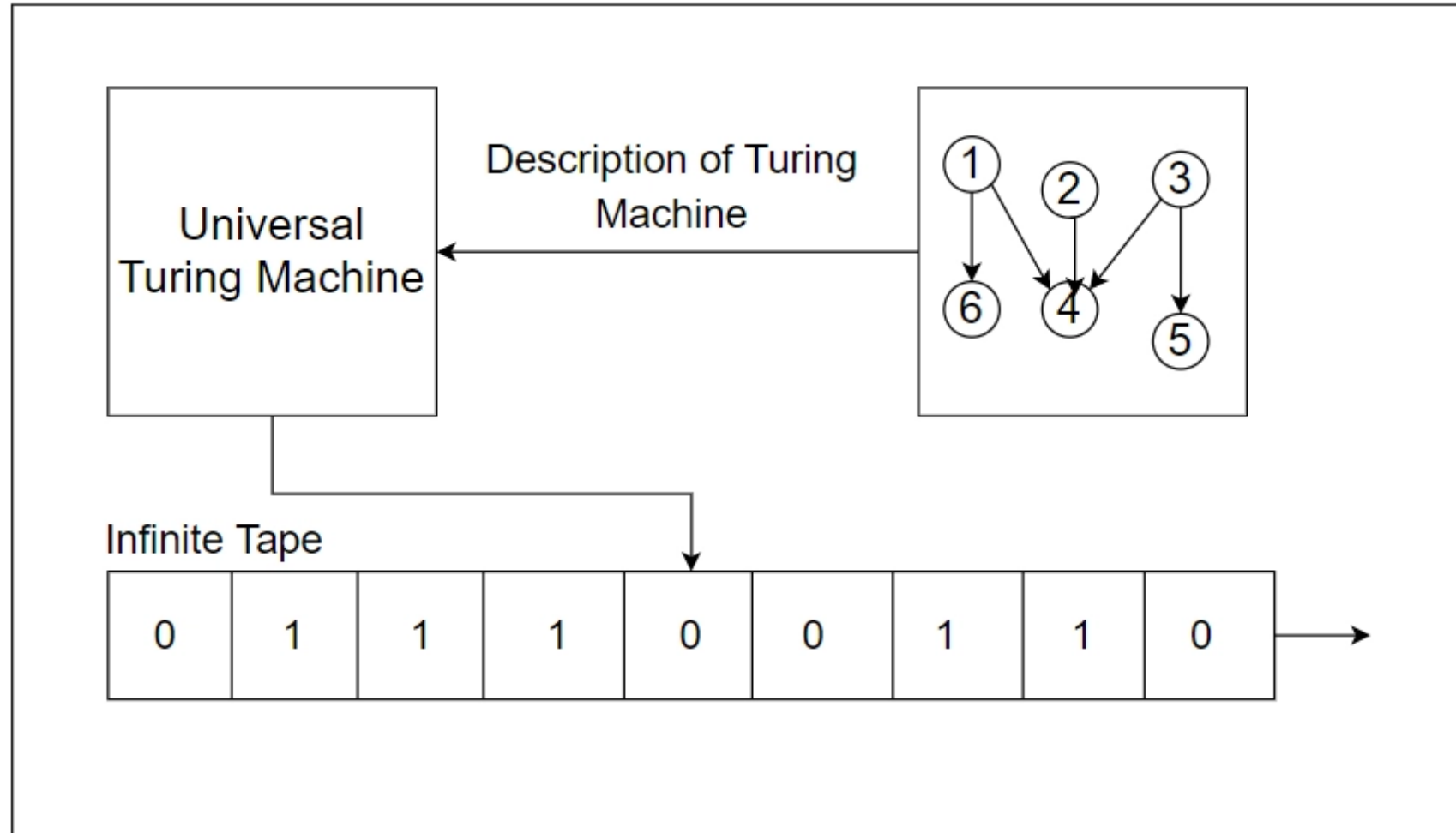
- ✓ Espacio: ¿cuánta memoria usa?



- ✓ Energía: ¿cuánta energía usa?

Entre menos mejor.

Decidibilidad



Decidibilidad


Algoritmo / Programa: Código de longitud constante (trabajando en una palabra en la RAM con s bits de tamaño en las palabras) para resolver un problema.



Problema. Producir la salida correcta para cada entrada, en donde la longitud del código es independiente del tamaño de la instancia.

Decidibilidad

Algoritmo / Programa: Código de longitud constante (trabajando en una palabra en la RAM con s bits de tamaño en las palabras) para resolver un problema.



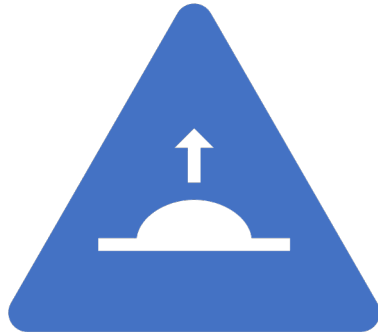
Problema. Producir la salida correcta para cada entrada, en donde la longitud del código es independiente del tamaño de la instancia.



El problema es decidable si existe un programa para resolverlo en tiempo finito.

Dada una Máquina de Turing M y una palabra w , determinar si M terminará en un número finito de pasos cuando es ejecutada usando w como dato de entrada.

Decidibilidad

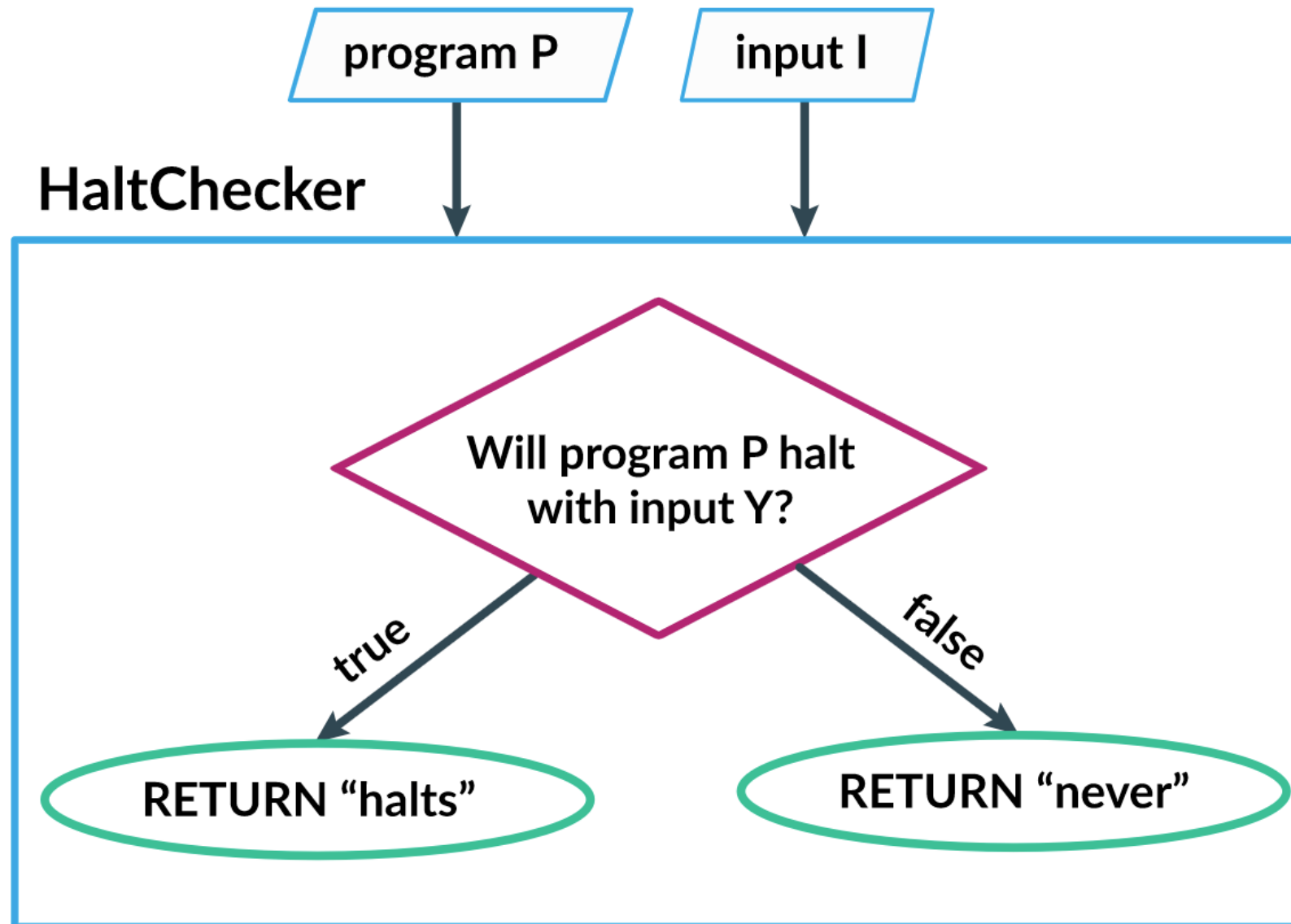


Un ejemplo clásico de un problema algorítmico irresoluble es el problema de la detención (paro – Halting problem).

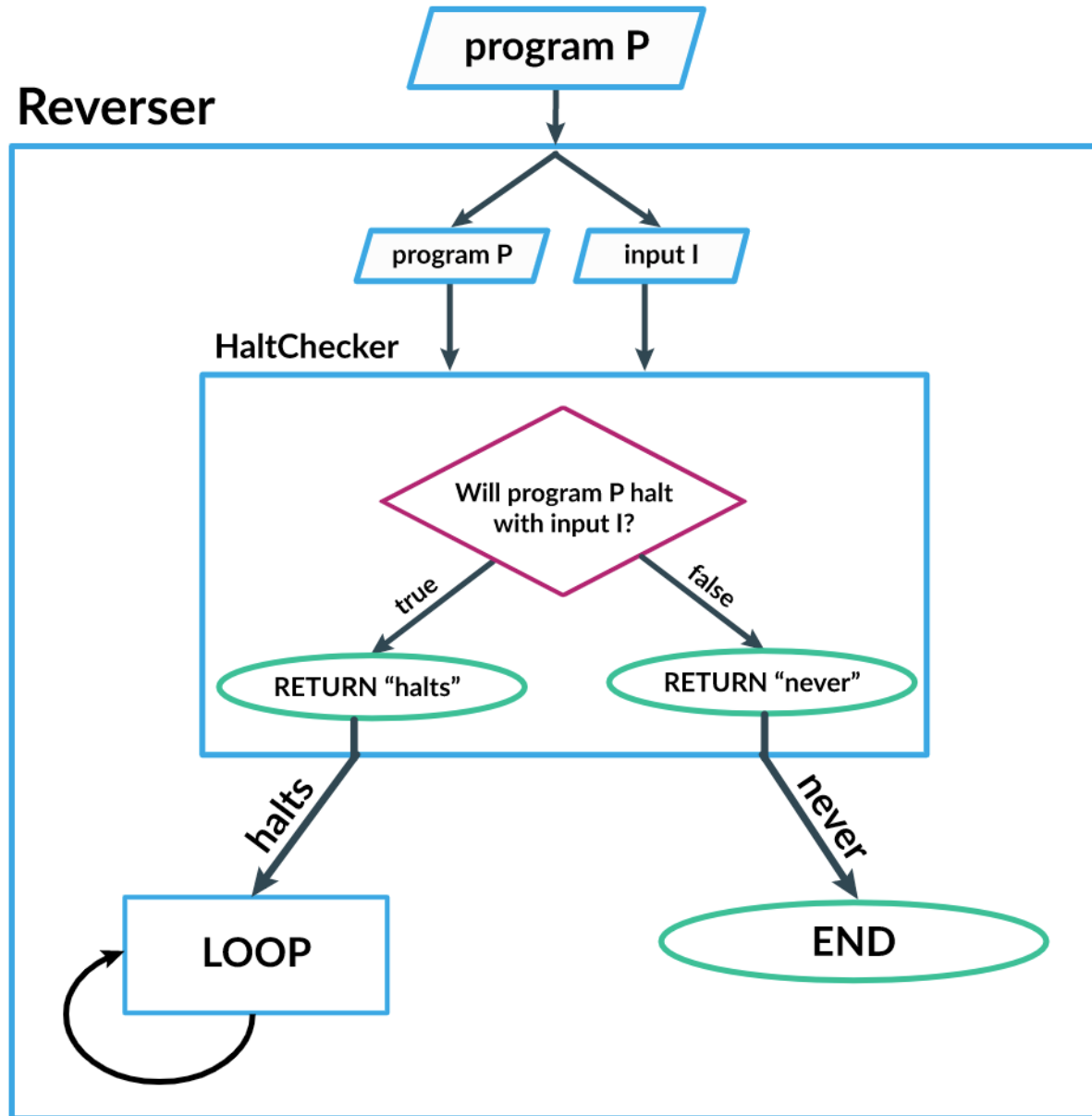


Establece que no se puede escribir ningún programa que pueda predecir si otro programa se detiene o no después de un número finito de pasos.

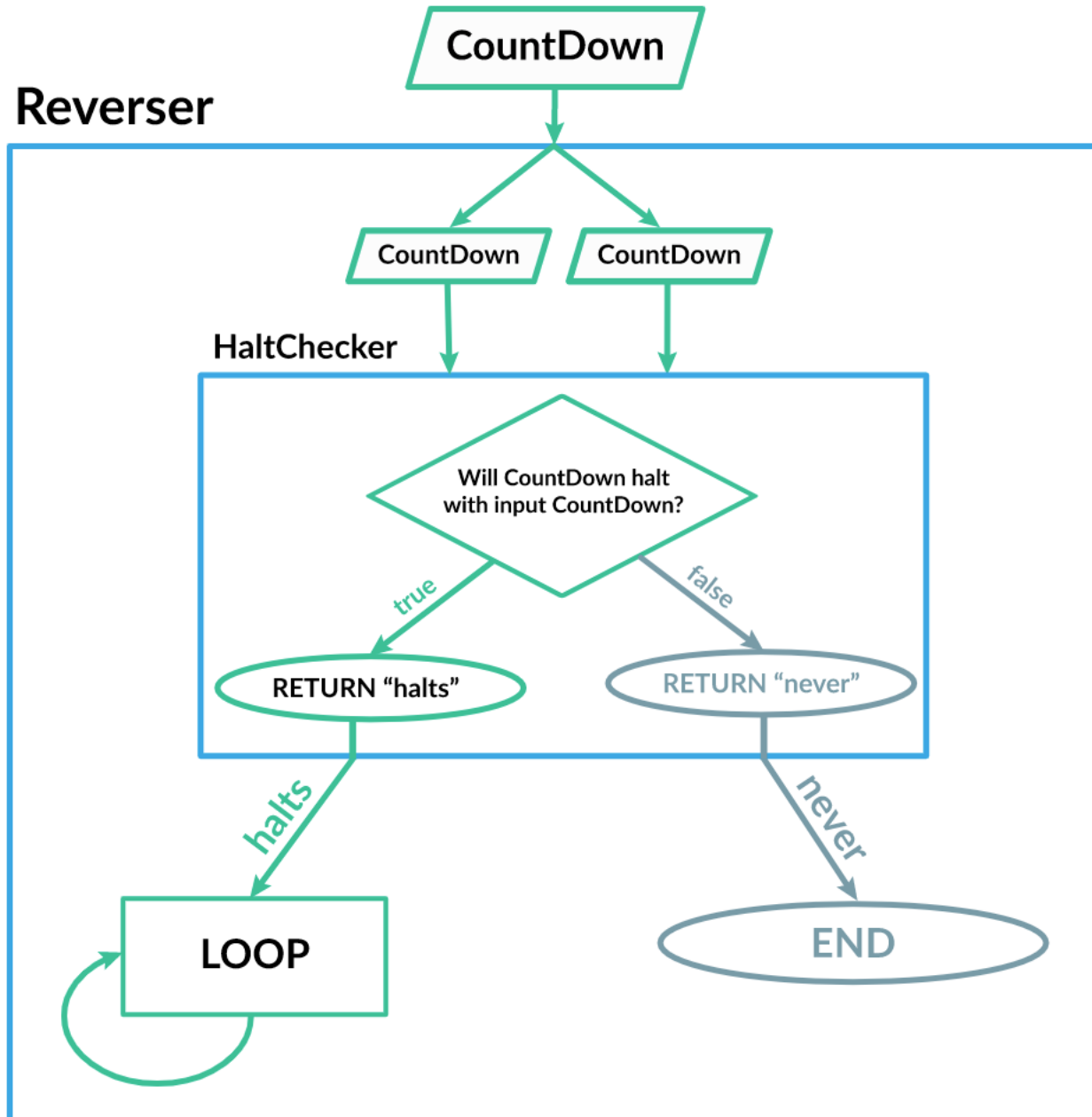
Decidibilidad



Decidibilidad



Decidibilidad



def Termina(p, x):

*#Supongamos que aquí se encuentra un código maravilloso que
soluciona el problema de la parada*

#Esta función regresa True si $p(x)$ termina o False en otro caso

```
def Diagonal(w):  
    if Termina(w, w):  
        while True: pass #Esta instrucción es un bucle infinito
```



```
def Diagonal(Diagonal):  
    if Termina(Diagonal, Diagonal):  
        while True: pass
```

Diagonal(Diagonal) termina \iff Diagonal(Diagonal) nunca termina

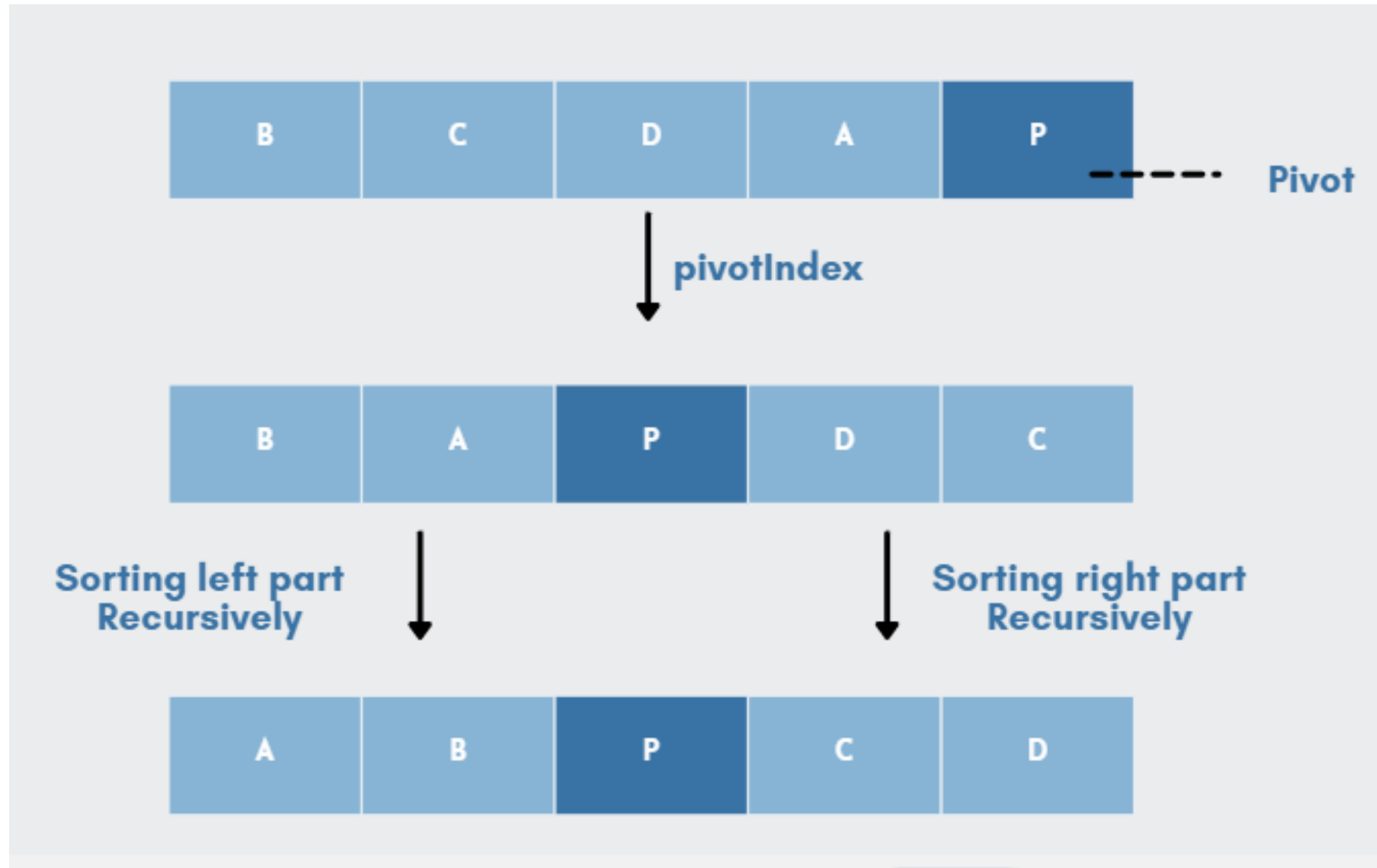
La imposibilidad de resolver el problema de la detención tiene una influencia práctica inmediata en el desarrollo de software.

Decidibilidad

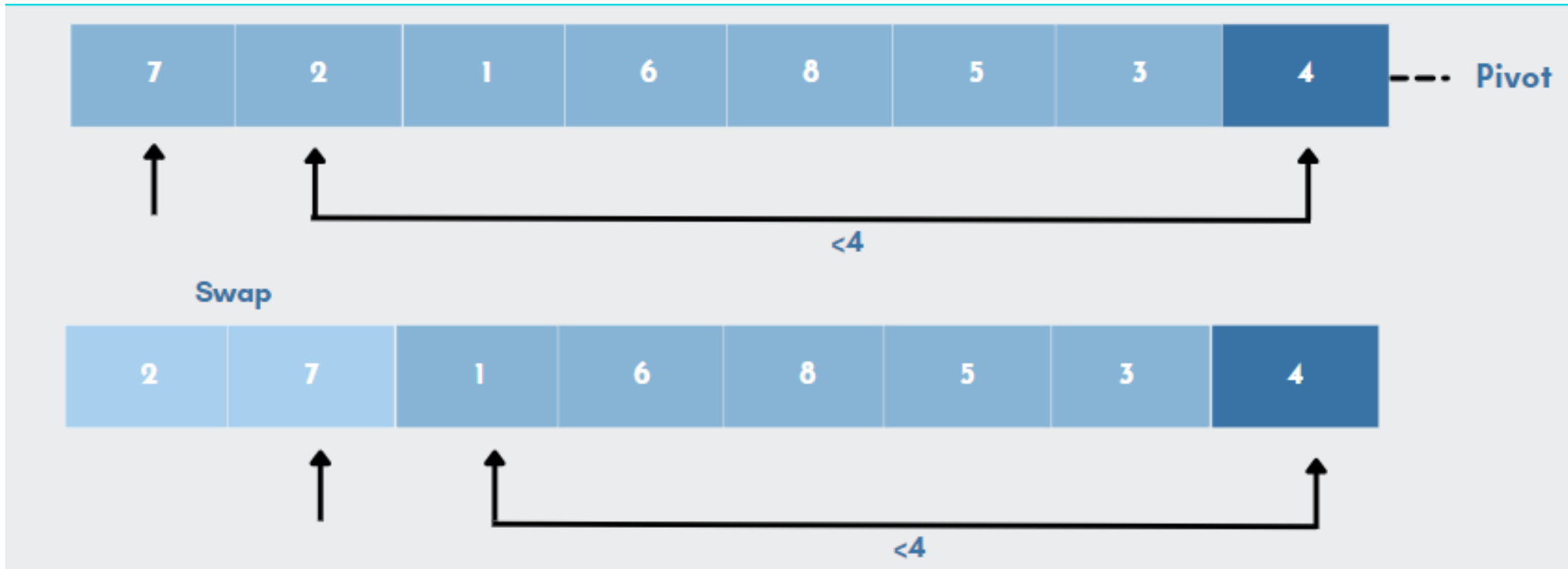
La imposibilidad de resolver el problema de la detención tiene una influencia práctica inmediata en el desarrollo de software.

Por ejemplo, sería riesgoso tratar de desarrollar una herramienta de software que prediga si otro programa que se está desarrollando tiene un ciclo infinito (aunque tener tal herramienta sería inmensamente beneficioso).

Costo de Algoritmo: Ejemplo (Quick Sort)



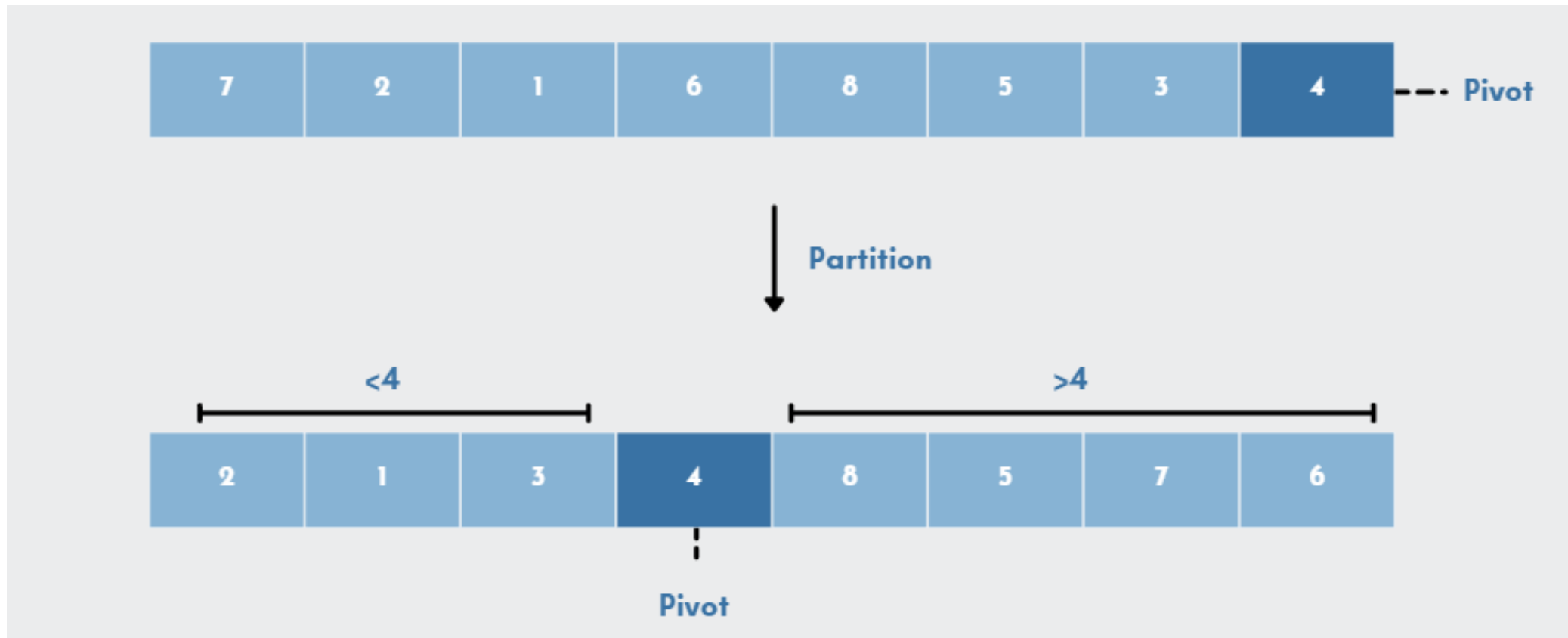
Costo de Algoritmo: Ejemplo



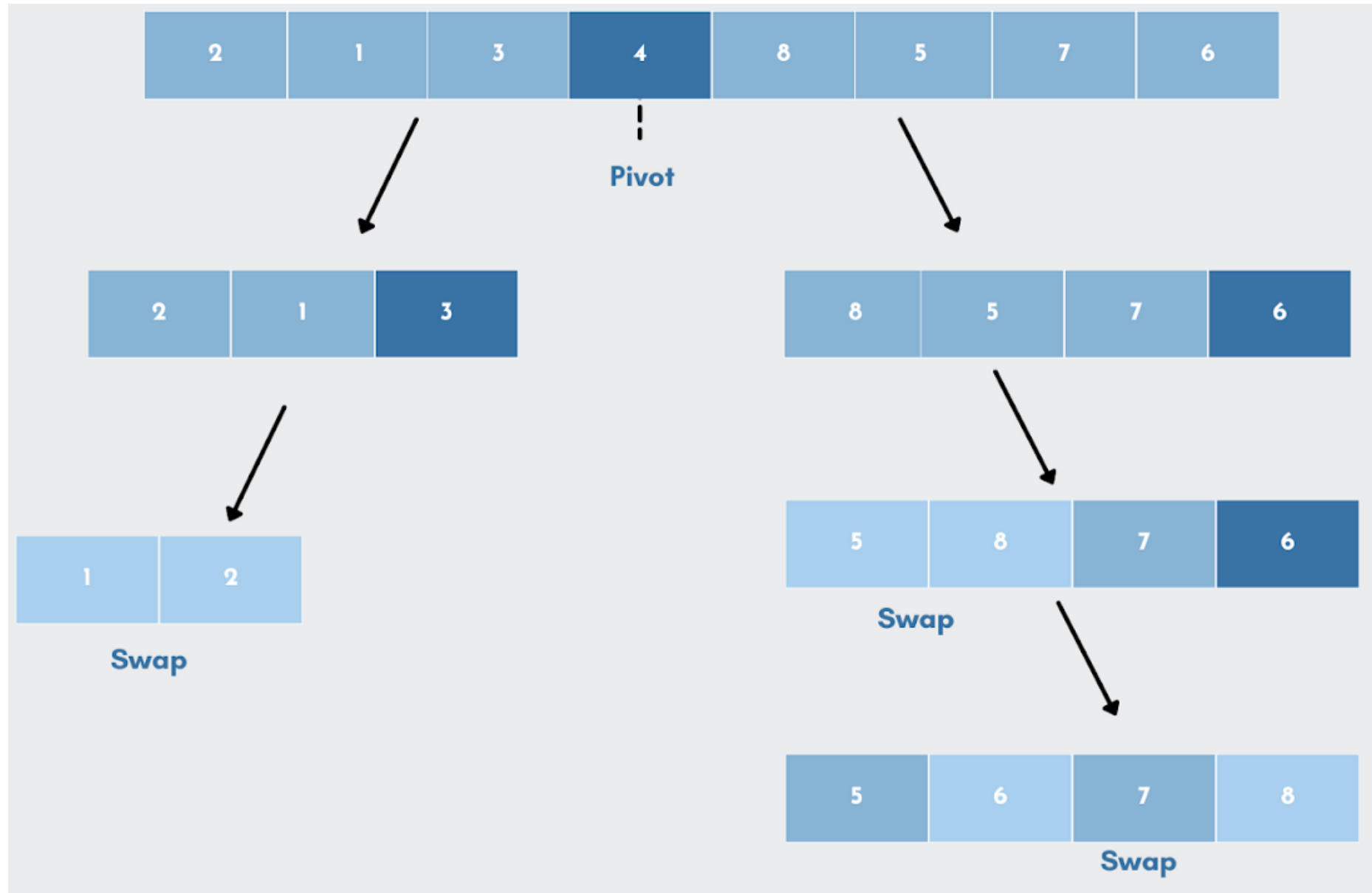
Costo de Algoritmo: Ejemplo



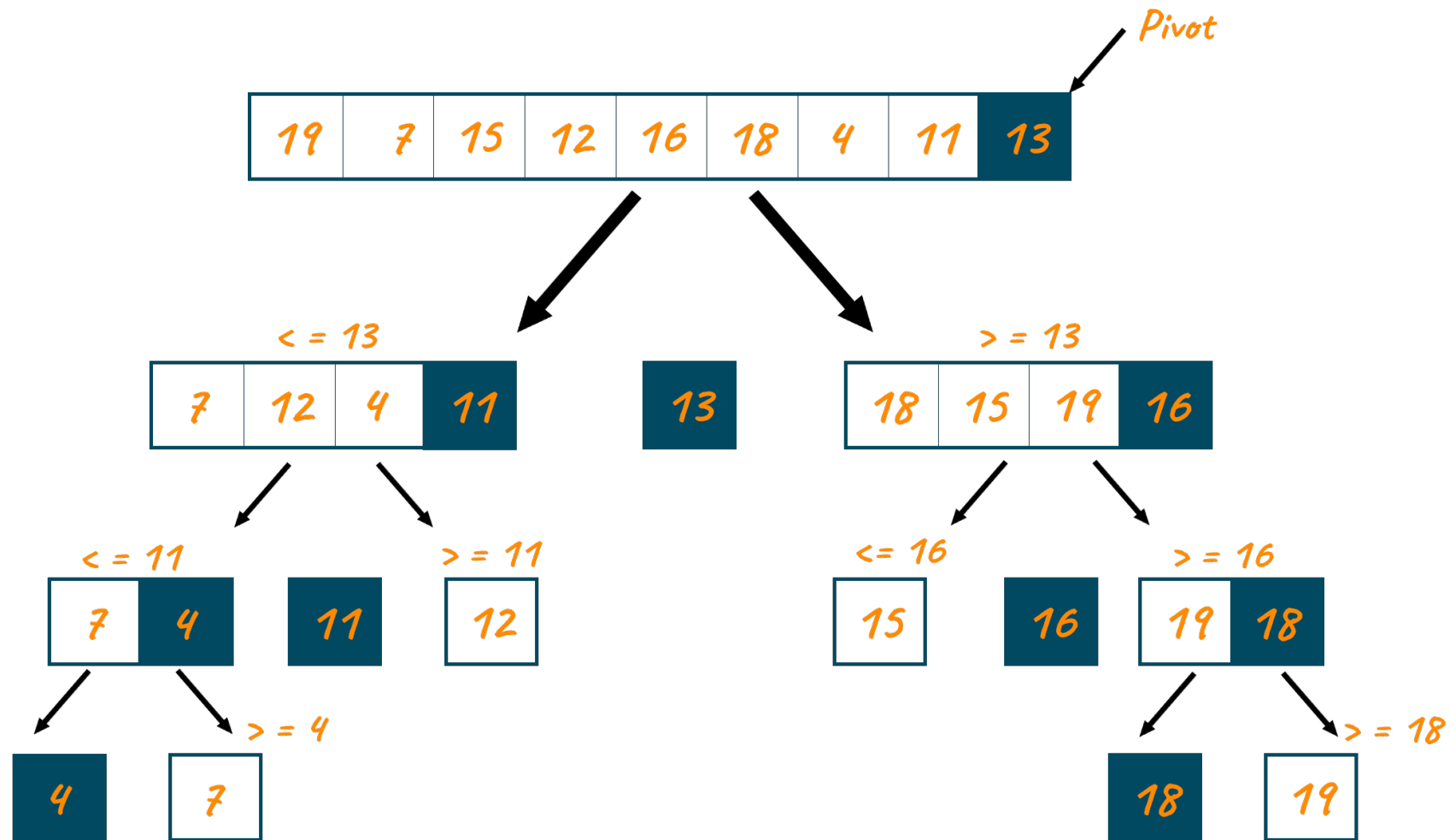
Costo de Algoritmo: Ejemplo



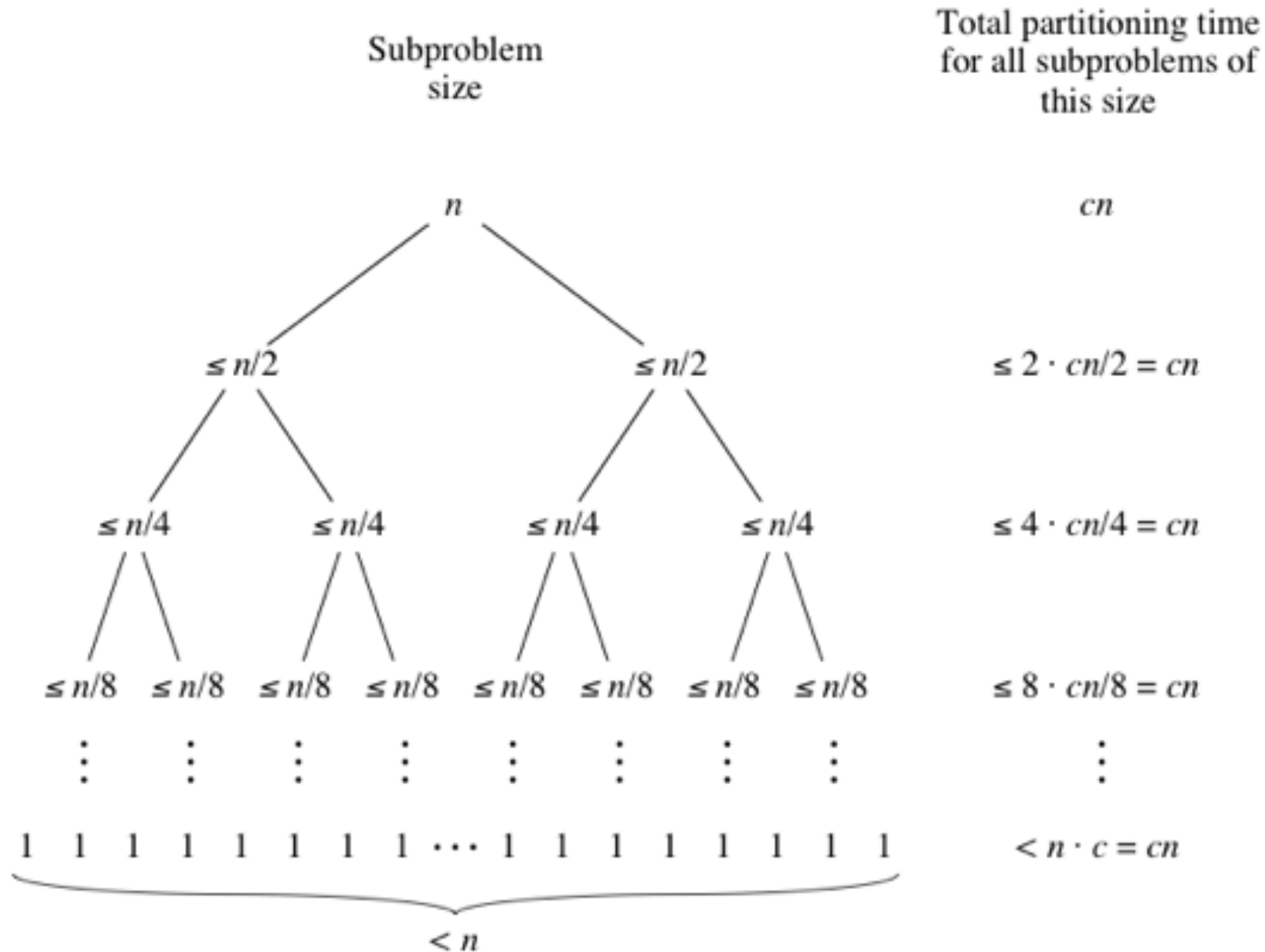
Costo de Algoritmo: Ejemplo



Costo de Algoritmo: Ejemplo



Costo de Algoritmo: Ejemplo



Costo de Algoritmo: Ejemplo

Case	Time Complexity
Best Case	$O(n \cdot \log n)$
Worst Case	$O(n^2)$
Average Case	$O(n \cdot \log n)$

Costo de Algoritmo: Ejemplo

Theorem: $PATH \in P$

Proof: $M =$ "On input $\langle G, s, t \rangle$

1. Mark s

2. Repeat until nothing new is marked:

For each marked node x :

Scan G to mark all y where (x, y) is an edge

3. *Accept* if t is marked. *Reject* if not.

$\leq n$ iterations

$\times \leq n$ iterations

$\times O(n^2)$ steps

 $O(n^4)$ steps

Costo de Algoritmo: Big – O notation

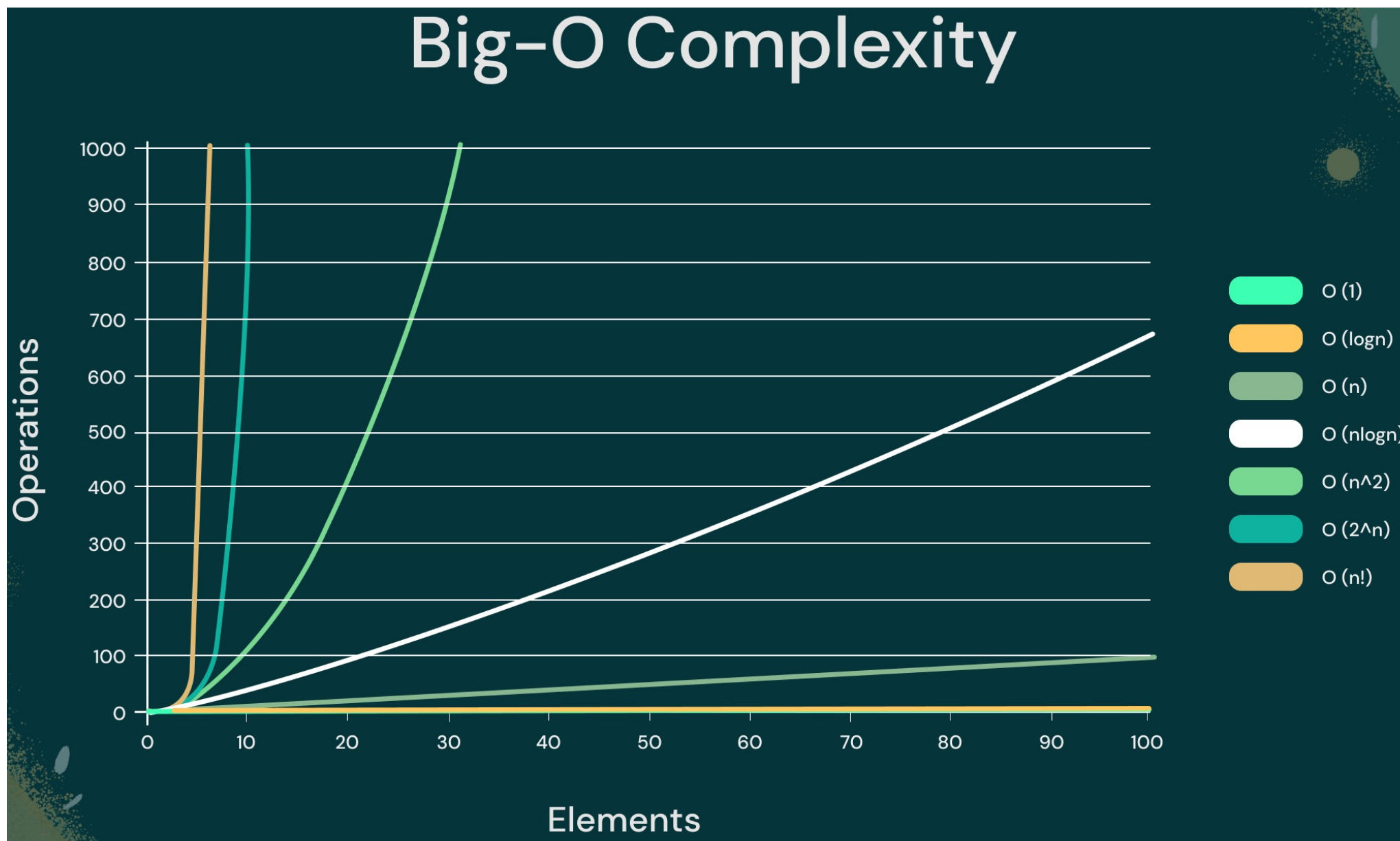


Imagem tomada de <https://www.crio.do/>

Costo de Algoritmo: Funciones para Calcular Costos

Para un algoritmo dado, el objetivo es encontrar las siguientes funciones:

$T(n)$: el costo del tiempo para resolver el problema.

$S(n)$: el costo del espacio para resolver el problema.

$E(n)$: el costo de la energía para resolver el problema.

El costo de un algoritmo depende de varias características.

El costo de un algoritmo depende de varias características.

- La semántica del modelo de programación.
- La topología de la red.
- Los tipos de datos usados.
- Protocolos de comunicación.

Costo de Algoritmo: Ejemplos

- $T(n)$: el costo del tiempo para resolver el problema
 - $O(n)$
 - $O(n^2)$
 - $O(2^n)$
- $S(n)$: el costo del espacio para resolver el problema
 - $O(n)$
 - $O(n^3)$
 - $O(2^n)$

Costo de Algoritmo: Ejemplo de Costo de Energía

$E(n)$: el costo de la energía para resolver el problema.

Alg	Dataset	Acc (%)	Energy (J)		
			Total	Processor	DRAM
HAT	RandomRBF (0.001)	65.75	578.03	558.34	19.69
HAT	RandomTree	97.75	758.80	728.32	30.49
VFDT	RandomRBF (0.001)	56.45	516.92	496.43	20.49
VFDT	RandomTree	97.40	363.30	348.60	14.70

2. Modelo de mapeo y reducción

2.1 Sistema de almacenamiento y procesamiento distribuido

2.2 Modelo de programación

2.3 Algoritmos con el modelo de mapeo y reducción

2.4 Extensiones

2.5 El modelo costo-comunicación

2.6 Teoría de la complejidad para el modelo de mapeo y reducción