.UBAfiuba
FACULTAD DE INGENIERÍA

Clases “P” y “NP”

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Introducción

Observamos

diferentes algoritmos para resolver problemas concretos

Llamamos

“tractables” a aquellos resolubles en tiempo polinomial.

Vimos

Que para algunos problemas existen algoritmos tractables (y para otros no)

Deseamos

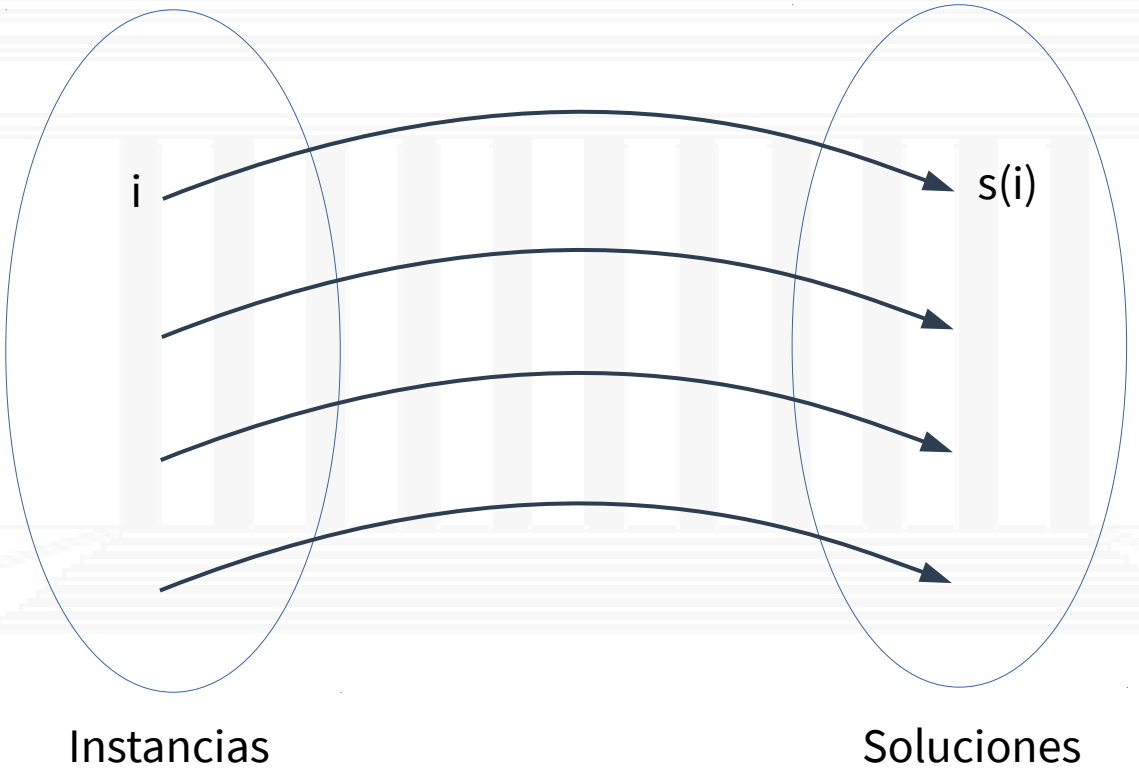
Poder clasificar a los problemas de acuerdo a la complejidad de su resolución

Y de esa forma poder compararlos.

Problema Abstracto vs Instancia de problema

Podemos definir un problema abstracto Q

Como una relación binaria entre un conjunto de instancias y de soluciones



Problema de optimización

Son aquellos problemas

que buscan la mejor solución para maximizar (o minimizar) un resultado.

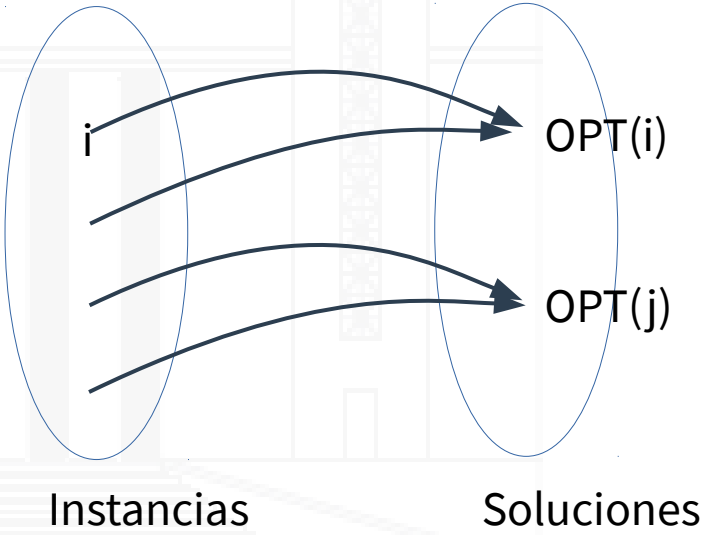
Ejemplos

Maximizar la ganancia que se puede ingresar en una mochila

Minimizar el cambio de monedas

Maximizar el flujo transportado en una red

Minimizar la longitud del circuito del viajante de comercio



Problema de decisión

Son aquellos problemas

Cuya solución solo puede tomar 2 valores: SI/NO

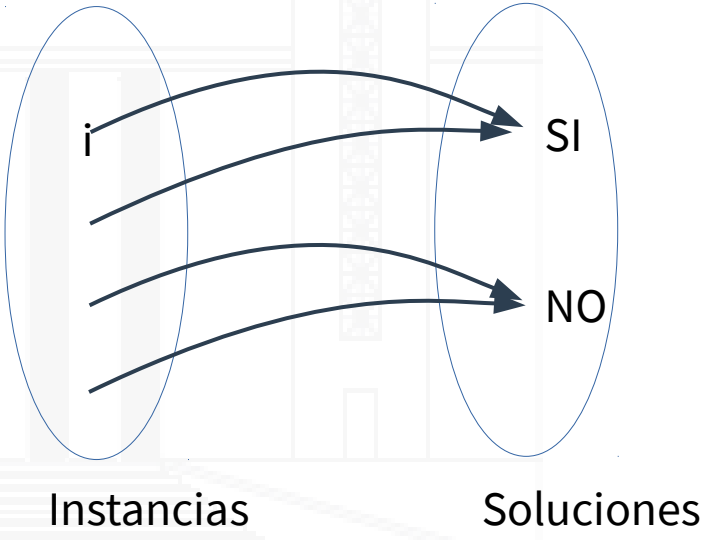
Ejemplos

¿ Es el número X primo?

¿ Existe un matching perfecto par los sets A y B ?

¿Es el grafo G fuertemente conexo?

¿ Se puede satisfacer la demanda de flujo para el problema de flujo máximo con demanda ?



Adecuación de problema optimización a decisión

Dado un problema de optimización Q

Podemos reformularlo como un problema de decisión (y viceversa)

Y encontrar la respuesta en tiempo polinómico (o pseudopolinómico)

Ejemplo

Optimización: Cambio mínimo en monedas del valor X

Decisión: Se puede dar cambio mínimo de X con m monedas?

```
Cambio mínimo en monedas del valor X
    Desde #m=1 hasta X
        Si (Se puede dar cambio cambio mínimo de X con #m monedas)
            retornar #m
```

Caracterización de la entrada de un algoritmo

Para resolver un problema concreto con una computadora

Se debe representar la instancia de modo que el programa lo entienda.

Los mismos corresponderán a los parámetros del algoritmo

Los parámetros de un problema computacional

Se codifican como una cadena binaria finita “s”

La longitud de la cadena “s”

La podemos medir como un parámetro n = |s|

Y la utilizamos para medir la complejidad del algoritmo que resuelve el problema

Resolución eficiente

Un algoritmo A

resuelve eficientemente un problema S

Si para toda instancia I de S,

Encuentra la solución en tiempo polinomial

→ existe una constante k / A = $O(n^k)$

Ejemplo:

Gale Shapley resuelve el problema “Stable Marriage Problem” en $O(n^2)$

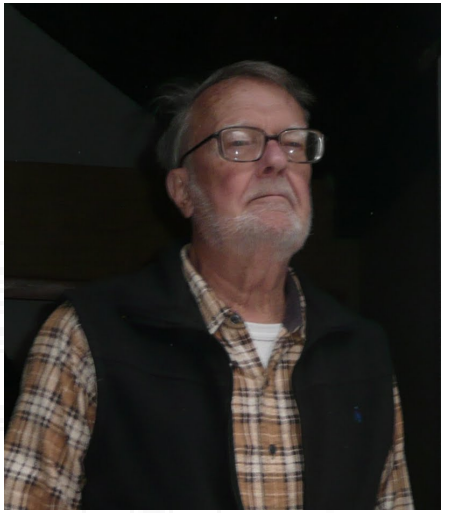
Clase “P”

Se conoce como “P”

Al conjunto de problemas de decisión

para los que existe

un algoritmo que lo resuelve en forma eficiente.



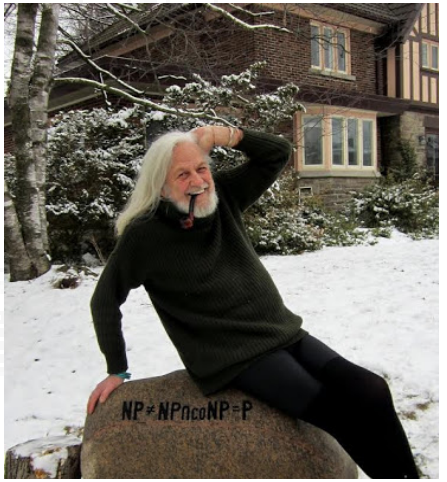
El concepto fue propuesto por:

“The intrinsic computational difficulty of functions”, Alan Cobham en 1964.

https://www.cs.toronto.edu/~sacook/homepage/cobham_intrinsic.pdf

“Paths, trees, and flowers”, Jack Edmonds en 1965.

https://math.nist.gov/~JBernal/p_t_f.pdf



Certificación eficiente

Un algoritmo B

Certifica (o verifica) eficientemente un problema de decisión S

Si para toda instancia I de S,

Dado un certificado t que contiene evidencia de la solución s(I) es “si”

Puede verificar esta afirmación en tiempo polinomial

→ existe una constante k / $B \leq O(n^k)$

El algoritmo B

Recibe 2 parámetros: la instancia I y t

Responde: “Si” o “No”

Certificación eficiente - Ejemplo

Dado el problema de decisión

¿Existe para el viajante de comercio un camino de longitud menor a L para un grafo G=(C,R) (que representa a ciudades y rutas)?

Un algoritmo de certificación recibe como parámetros

El grafo G (instancia del problema)

Un circuito t de ciudades para el viajante (certificado)

Calcula y verifica (en tiempo polinomial)

La longitud del circuito y que el mismo cubra todas las ciudades solo 1 vez

Retorna

“Si” si la evidencia otorgada satisface el problema.

Clase “NP”

Se conoce como “NP”

Al conjunto de problemas de decisión

para los que existe

un algoritmo que lo certifique en forma eficiente.

El concepto fue propuesto por:

“Paths, trees, and flowers”, Jack Edmonds en 1965.

https://math.nist.gov/~JBernal/p_t_f.pdf

¿ “P” ⊆ “NP” ?

Si el problema Q ∈ P

Existe un algoritmo A = $O(n^k)$ que lo resuelve

Podemos definir B(I,t) como

```
B(I,t)
    s = A(I)
    Si s == t
        retornar “si”
    retornar “no”
```

que certifica el problema Q

Y lo hace en tiempo polinomial

Si Q ∈ P ⇒ Q ∈ NP

¿ “NP” ⊆ “P” ?

Si el problema Q ∈ NP

Existe un algoritmo B = O(nk) que lo certifica

¿ Podemos asegurar que la existencia de B

Garantiza la existencia de un algoritmo A polinomial que lo resuelva?

En caso afirmativo: P=NP

“Tiene la misma complejidad resolver un problema que verificarlo”

Sino: P≠NP

“P” VS “NP”

Es un problema

sin resolver dentro de la ciencia de la computación

Propuesto en

"The Complexity of Theorem Proving Procedures", Stephen Cook en 1971

http://www.cs.toronto.edu/~sacook/homepage/1971.pdf

Una amplia mayoría considera que P≠NP

Pero no existe prueba

Millennium Problems

7 problemas matemáticos

Propuestos por Clay Mathematics Institute

El 24 de Mayo de 2000

Una solución correcta acredita un premio de 1millon de dólares

(hasta el momento solo 1 problema fue resuelto)

P vs NP es uno de esos problemas

https://www.claymath.org/millennium-problems

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020