

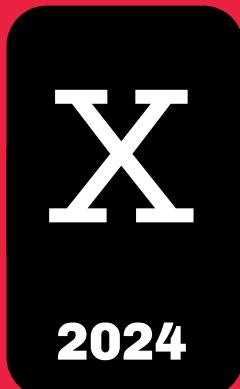
Algoritmos I

UNRN

Universidad Nacional
de **Río Negro**

rXXI

T



Algoritmos



¿Qué son?

	3	2
+	6	4
<hr/>		

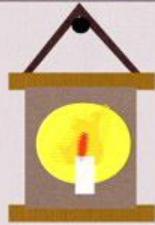
	5	2
+	6	9
<hr/>		

Suma (tradicional) de dos números enteros positivos

UNRN

Universidad Nacional
de Río Negro

Propiedades



PHASE 1 PHASE 2 PHASE 3

Collect
underpants

?

Profit



Finito*

1

Debe terminar después de un número finito de pasos.

Definido

2

Cada paso debe ser claro y sin
ambigüedad.

Eficiente*

3

Debe resolver el problema en un
tiempo razonable.

Efectivo

4

Cada paso del algoritmo debe ser
realizable y simple.

Entrada

5

Un algoritmo debe tener al menos una entrada, que es la información que necesita para procesar.

Salida

6

Un algoritmo debe tener al menos una salida, que es su resultado.

Entrada

Proceso

Salida

Y así, nos remontamos a la clase #2, de funciones

UNRN

Universidad Nacional
de Río Negro

Formas de representación

1. Pseudocódigo
2. Diagramas de flujo
3. Código en un lenguaje específico

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



Complejidad

introducción a



¿Qué es?

Búsqueda chica / Búsqueda grande

Ejemplo: Búsqueda

i

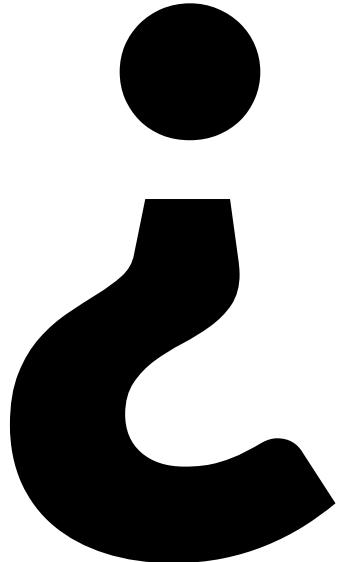
0	5	10	15	20	25	30	35	40	45	50	55	60	65	70
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1. ¿el elemento i es el que busco?
2. $i = i + 1$
3. Repetir
(hasta encontrar o agotar el conjunto)



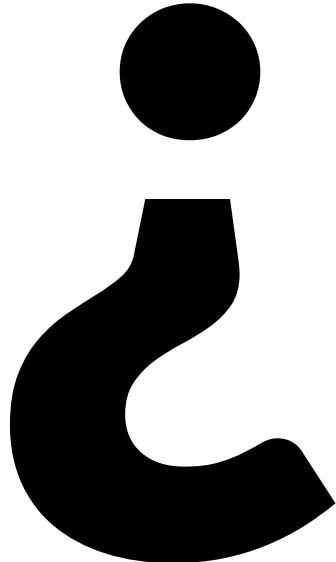
0	5	10	15	20	25	30	35	40	45	50	55	60	65	70
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1. $i = \text{largo} / 2$
2. ¿el elemento i es mayor a que busco?
a. $i = (i / 2) + i$
3. ¿el elemento i es menor a que busco?
a. $i = i / 2$
4. Repetir
(hasta encontrar o agotar el conjunto)



**Qué
diferencia,
no**





Qué tipos de complejidades hay

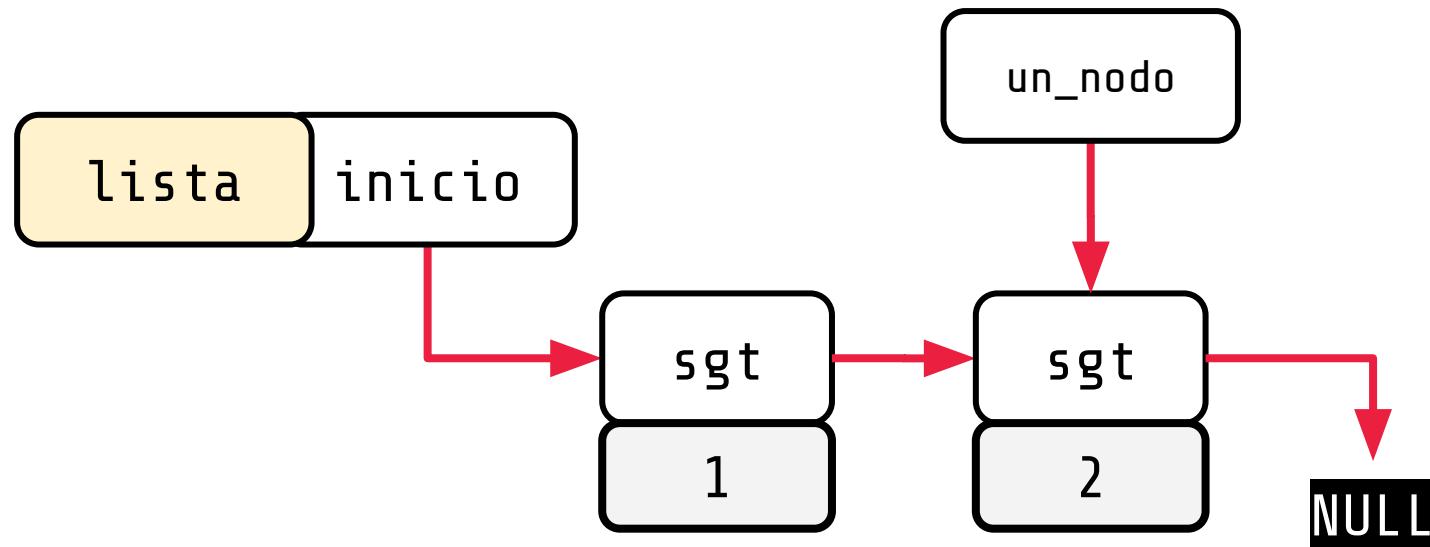


Complejidad temporal

Complejidad espacial

*No se habla tanto
de esta por ser
fácilmente
*observable**

¿cuánto se ‘gasta’ extra en una lista?





**Para qué se
diferencian**



¿Preguntas?



Notación O oh grande

 $O(n)$

Para la búsqueda secuencial

$O(\log n)$

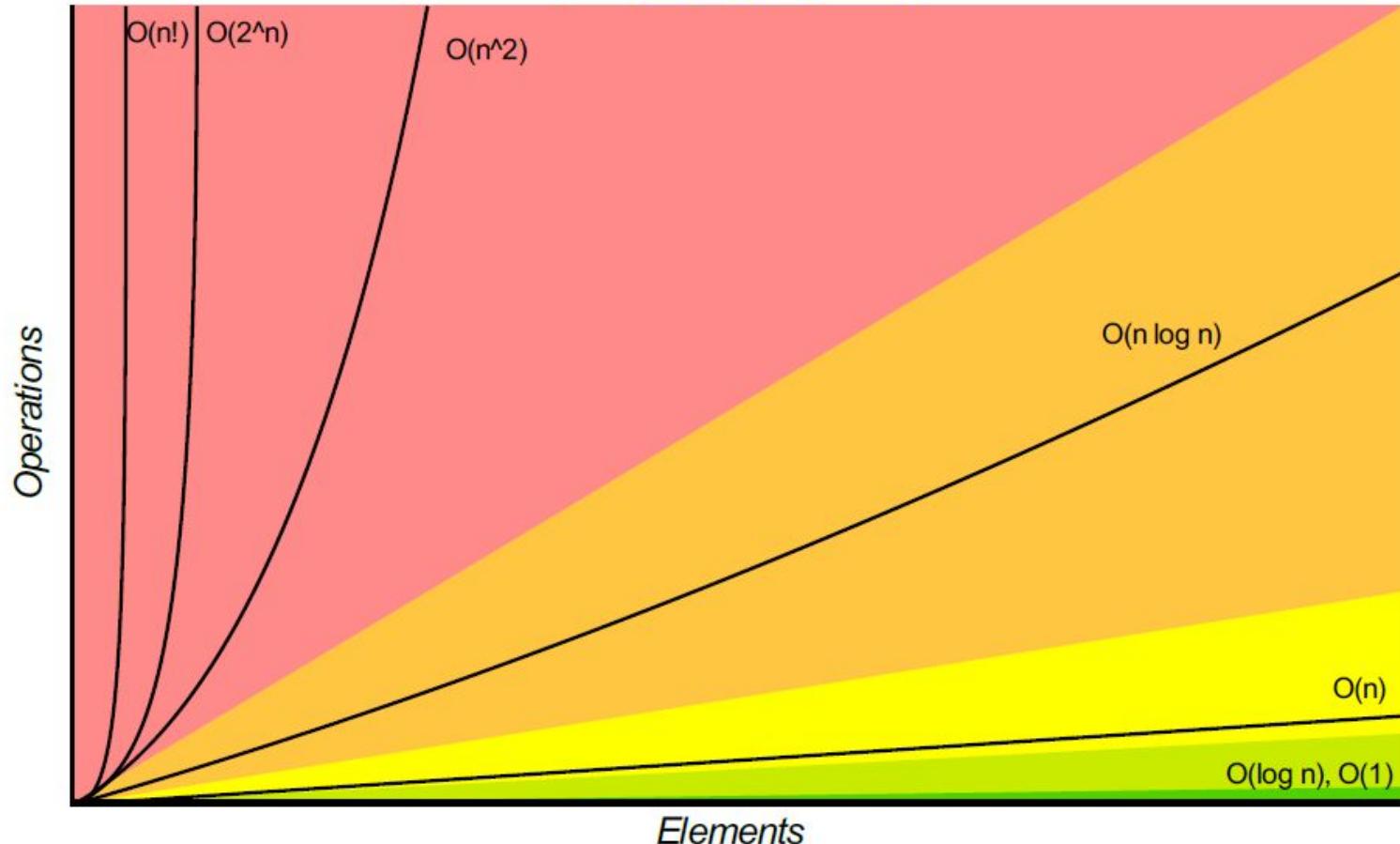
Para la búsqueda binaria

Algunas

Clases de complejidad

- $O(1)$ - constante
- $O(n)$ - lineal
- $O(\log n)$ - logarítmica
- $O(n^2)$ - cuadrática
- $O(n!)$ - factorial
- $O(2^n)$ - exponencial

Análisis asintótico

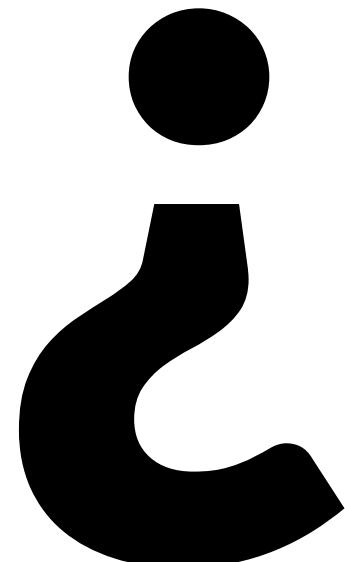
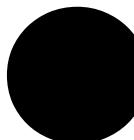


Complejidad con n grande

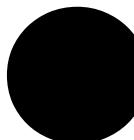
A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?





**Por qué
vemos esto**



**Que significa
intuitivamente**

$O(1)$

No es necesario recorrer para obtener el resultado esperado

$O(n)$

significa que hay que mirar cada elemento una vez

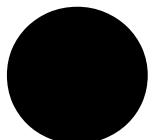
$O(n^2)$

significa que cada vez que se examina un elemento hay que examinar también todos los demás.



¿Preguntas?





**Como
aplicamos esto
a lo que hemos
implementado**

1

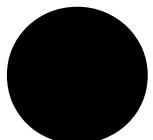
**Podemos medir el
tiempo de CPU que
le lleva completar la
tarea**

Cronómetro



2

**Podemos contar las
operaciones
elementales que
requiere**



**Qué es una
operación
elemental**

Contando operaciones elementales

Se suele llamar **complejidad algorítmica**.

Se trata de acotar la cantidad de operaciones elementales que toma resolver un problema en función del tamaño de la entrada.

Operaciones elementales

1

Todas las operaciones aritméticas sobre enteros y números de punto flotante.

Operaciones elementales

2

Las operaciones de acceso a elementos específicos en un arreglo.

Operaciones elementales

3

La lectura o escritura de una variable.

**Les asignaremos
costo 1**

Lazos - $O(1)$

1

Si el lazo va siempre hasta un valor constante c , el costo del mismo es fijo.

Lazos - $O(1)$

```
for (int i = 0; i <= c; i++)
{
    //operaciones O(1)
}
```

Lazos - $O(n)$

2

Si el lazo va hasta un valor variable n, y contiene operaciones $O(1)$ dentro, el mismo tendra un costo de $O(n)$

Lazos - $O(n)$

```
for (int i = 0; i <= n; i++)
{
    //operaciones O(1)
}
```

Lazos - $O(n)$

```
for (int i = 0; i <= n; i++)
{
    //operaciones O(1)
}
```

Lazos - $O(n)$

```
for (int i = 0; i <= c * n; i++)
{
    //operaciones O(1)
}
```

El valor c que multiplica a n , queda absorbido.

Lazos - $O(\log n)$

3

Si el lazo va hasta un valor variable n ,
contiene operaciones $O(1)$ dentro, y la
iteración aumenta en un factor de 2.
El mismo tendrá un costo de $O(\log n)$

Lazos - $O(\log n)$

```
for (int i = 0; i <= n; i = i * 2)
{
    //operaciones O(1)
}
```

En incremento y decremento

Lazos - $O(n^2)$

4

Si el lazo contiene otro en donde ambos van hasta n.

Lazos - $O(n^2)$

```
for (int i = 0; i <= n; i++)
{
    for (int j = 0; j <= m; j++)
    {
        //operaciones O(1)
    }
}
```

**Cada lazo anidado
sube el valor del
exponente**

Lazos - $O(n+m)$

5

La complejidad de dos lazos consecutivos, es la suma de los términos de ambos.

Lazos - $O(m+n)$

```
for (int i = 0; i < m; i++) {  
    // Operaciones O(1)  
}  
for (int j = 0; j < n; j++) {  
    // Operaciones O(1)  
}
```

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



Verificación de palíndromo

```
bool es_palindromo(char palabra[]) {  
    int inicio = 0;                                // O(1)  
    int fin = strlen(palabra) - 1;                  // O(n)  
  
    bool es_pal = true;                            // O(1)  
  
    while (inicio < fin) {                          // O(n)  
        if (palabra[inicio] != palabra[fin]) { // O(1)  
            es_pal = false;                    // O(1)  
            break;                           // O(1)  
        }  
        inicio++;                         // O(1)  
        fin--;                            // O(1)  
    }  
  
    return es_pal;                                // O(1)
```

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



Condicionales

1

Se toma la complejidad más alta en un condicional.

Se toma la ‘rama’ más costosa, pero una sola vez.

```
if (condicion)
{
    // instrucciones O(1)
}
else
{
    // instrucciones O(n)
}
```

Pero, el análisis
asintótico
simplifica los
términos

**Quedando en las
clases de
complejidad
*generales***

Casos de análisis

Peor caso

$O(n)$

Mejor caso $\Omega(n)$

Crecimiento exacto

$\Theta(n)$

Ejemplo $O(f(n)) = \Theta(f(n))$

```
int suma_elementos(int arreglo[], int n) {
    int suma = 0;                                // O(1)
    for (int i = 0; i < n; i++) {
        suma += arreglo[i];                      // O(n)
    }
    return suma;                                  // O(1)
}
```

Pase lo que pase, se recorren los n elementos

O(f(n)) ≠ Θ(f(n))

```
int busqueda_lineal(int arr[], int n, int buscado) {  
    for (int i = 0; i < n; i++) { // O(n)  
        if (arr[i] == buscado) { // O(1)  
            return i;           // O(1)  
        }  
    }  
    return -1;                  // O(1)  
}
```

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



El problema de la “computabilidad”



O por qué algunos
programas no son
finitos

¿Que significa
que no sea
computable?

Adivinar contraseñas



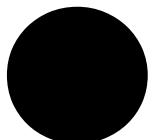
P vs NP

el problema del milenio



P vs NP

**o como ganar
un millón de USD**



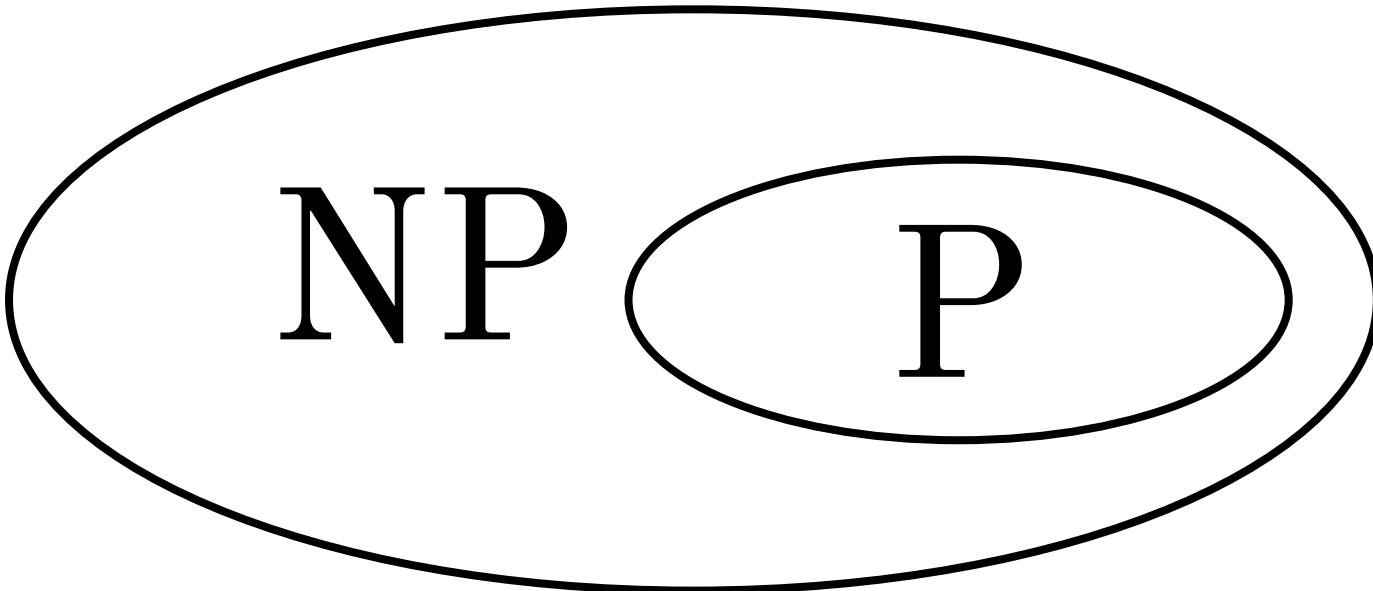
**Por qué es tan
importante**

Clases de complejidad

P tiempo polinómico

NP

tiempo no determinista polinómico

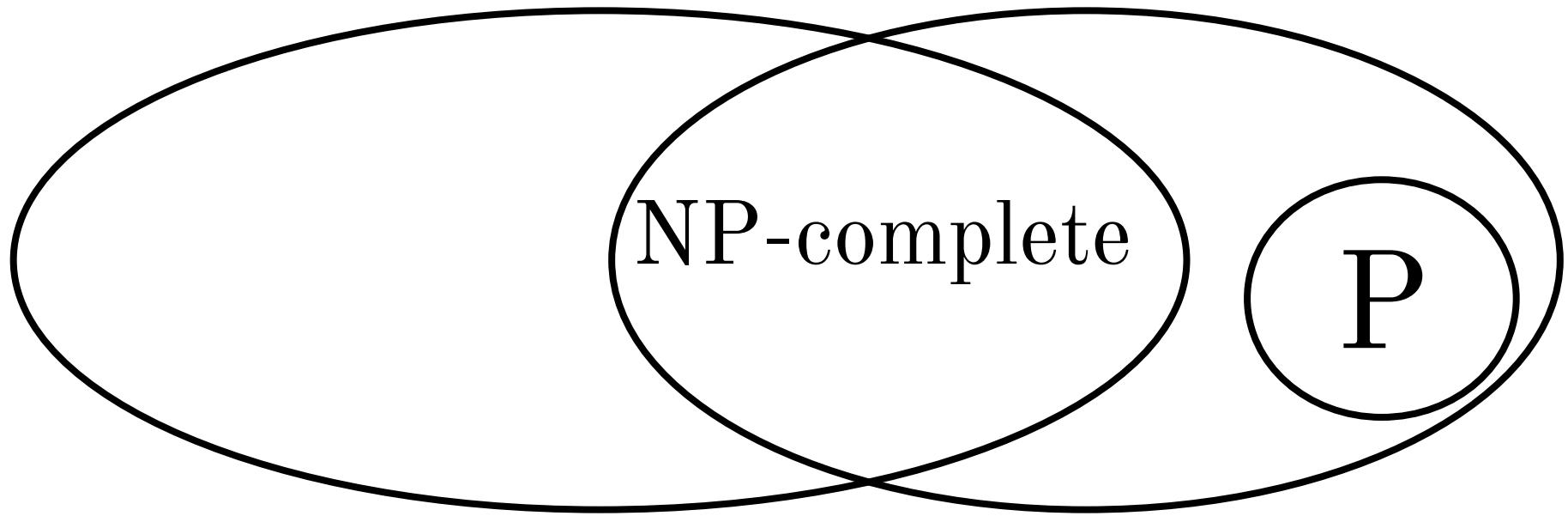


¿Como se relacionan entre sí?

UNRN

Universidad Nacional
de Río Negro

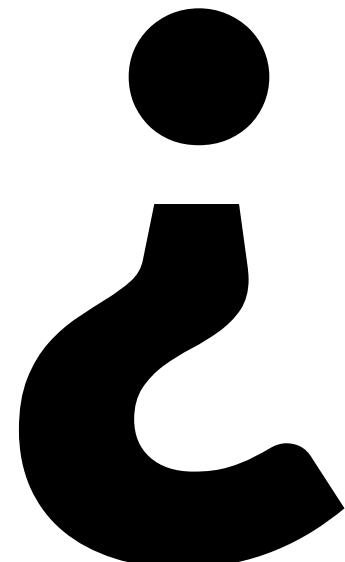
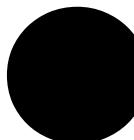
NP-Complete



¿Cómo se relacionan entre sí?

UNRN

Universidad Nacional
de Río Negro



Porque son importantes

¿Qué significa qué? $P = NP?$

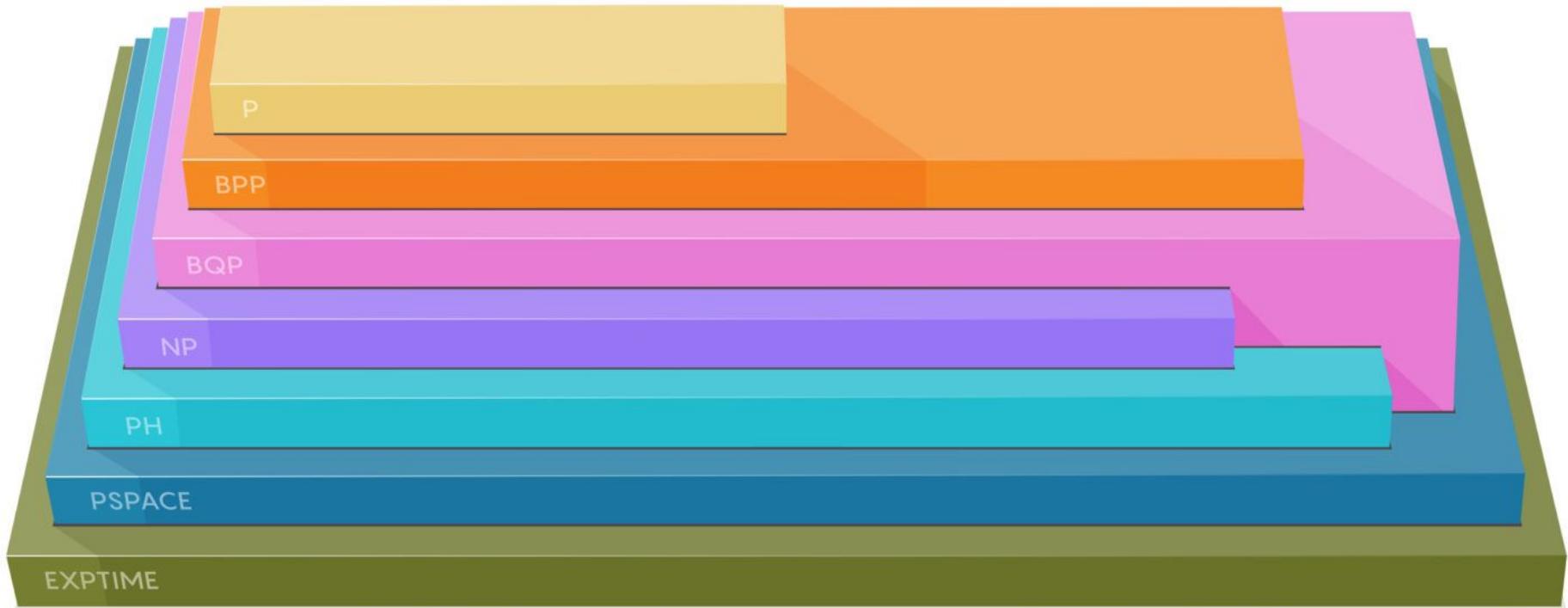
¿Qué significa qué? $P \neq NP?$

Alcanzabilidad en grafos



SAT

asignación de verdad satisfactoria en dos variables



Las categorías son más detalladas

UNRN

Universidad Nacional
de Río Negro

Son todavía más,
tango que tienen
un zoológico

(~50 tipos diferentes)

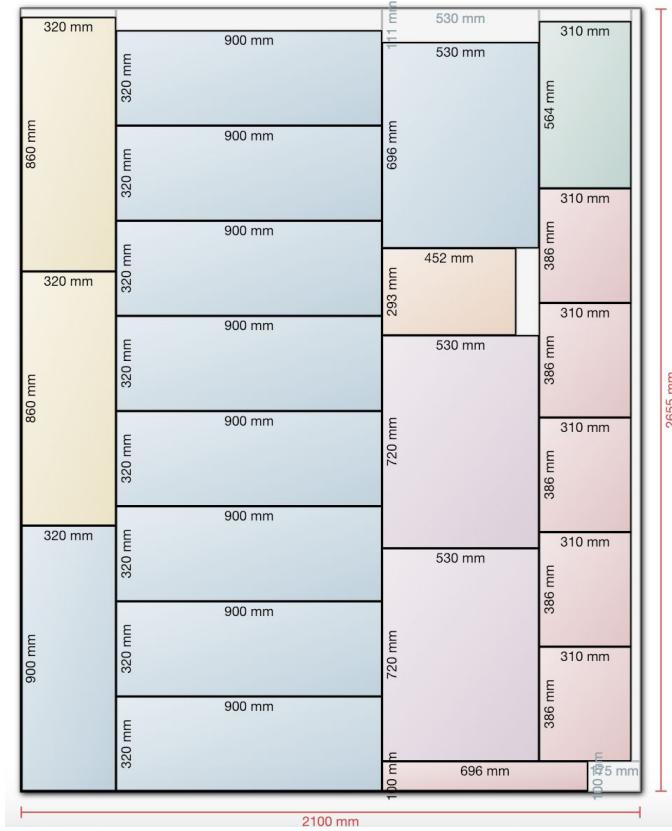
Empaquetado

https://en.wikipedia.org/wiki/Bin_packing_problem

Unidimensional

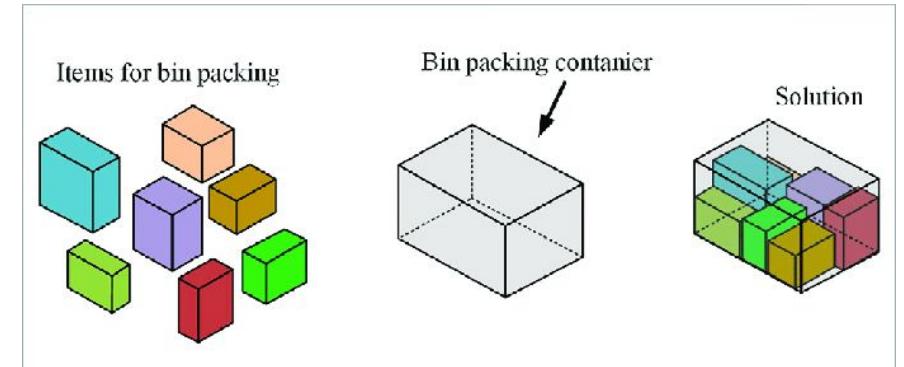
Colocar segmentos de una línea en un segmento más grande.

Bidimensional



¿Cómo reducimos el desperdicio de placa cortada?

Tridimensional



¿Como reducimos el espacio sin utilizar?

Hasta la próxima



unrn.edu.ar

UNRN

Universidad Nacional
de Río Negro



| unrnionegro