UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

Ingeniería en Computación

ALGORITMOS Y ESTRUCTURAS DE DATOS

Bv. J.A. Roca 989 / CP: 2300
Rafaela - Santa Fe - Argentina

T: +54 (03492) **501155**

info@unraf.edu.ar
www.unraf.edu.ar

ALGORITMOS Y ESTRUCTURAS DE DATOS

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

¿?

AGENDA

#IC

- Repaso
- Complejidad Computacional
  - ¿Qué es?
  - Complejidad Algorítmica
  - Tiempos
  - Costos
- Clases de Problemas
  - P
  - NP
  - NP Complejos
  - NP Duros
- Órdenes de Magnitud
- Ejemplos

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

TEORÍA DE COMPLEJIDAD COMPUTACIONAL

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Clasificar los problemas computacionales de acuerdo a su complejidad

Características de un Algoritmo

Precisión: Un algoritmo debe expresarse sin ambigüedad.

Determinista: debe responder del mismo modo antes las mismas condiciones.

Finito: La descripción de un algoritmo debe ser finita.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Complejidad Algorítmica

Representa la cantidad de **recursos** (temporales) que necesita un algoritmo para resolver un problema y por tanto permite determinar la **eficiencia** de dicho algoritmo.

Medida del tiempo independiente

- De la máquina
- Del lenguaje de programación
- De cualquier otro elemento de hardware o software que influya.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Costo de la Complejidad Algorítmica

Depende del tamaño de los datos

Costo esperado o promedio

Mejor costo

Peor costo

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Complejidad Temporal

$T(n)\ T_{max}(n)\ T_{min}(n)\ T_{med}(n)$

```
mientras b>0  hacer
    a = a + 1
    b = b - 1
```

T=T(b)

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Complejidad Costo

Búsqueda de un elemento en un vector de n posiciones

Mejor costo: el elemento este en la primera posición.

Peor costo: recorrer todo el vector.

Costo promedio: n/2

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Complejidad Computacional

Clase P

Clase NP

Clase NP-completos

Clase NP-duros

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Complejidad Computacional

Clase P: complejidad polinómica son tratables en el sentido que son abordables en la práctica.

Complejidad Computacional

Clase NP: problemas intratables pueden caracterizarse por aplicarse un algoritmo polinómico para comprobar si una posible solución es valida o no. Método de resolución no determinista consistente en aplicar heurísticas para obtener soluciones hipotéticas que se van desestimando (o aceptando) a ritmos polinómicos.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Complejidad Computacional

Clase NP-completos: se destacan por la extrema complejidad. Se hallan en la frontera externa de la clase NP. Son problemas NP y son los peores problemas posibles de clase NP.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Complejidad Computacional

Clase NP-duros: puede ser transformado a un problema NPC y tendrá la propiedad que no podrá ser resuelto en tiempo polinomial a menos que P = NP. Es tan difícil como un NP Completo.

Ejemplo

n \ T(n)	log n	n	n log n	n^2	n^3	2^n	n!
10	$3.3\ 10^{-6}$	10^{-5}	$3.3\ 10^{-5}$	10^{-4}	0.001	0.001	3.63
50	$5.6\ 10^{-6}$	$5\ 10^{-5}$	$2.8\ 10^{-4}$	0.0025	0.125	intratable	intratable
100	$6.6\ 10^{-6}$	10^{-4}	$6.6\ 10^{-4}$	0.01	1	intratable	intratable
10^3	10^{-5}	0.001	0.01	1	1000	intratable	intratable
10^4	$1.3\ 10^{-5}$	0.01	0.13	100	10^6	intratable	intratable
10^5	$1.6\ 10^{-5}$	0.1	1.6	10^4	intratable	intratable	intratable
10^6	$2\ 10^{-5}$	1	19.9	10^6	intratable	intratable	intratable

Imaginemos un microprocesador que puede realizar 1000 instrucciones por nanosegundo
En 1 segundo ejecuta 10^{12} instrucciones

n=50 - $O(2^n)$ -> 19 minutos

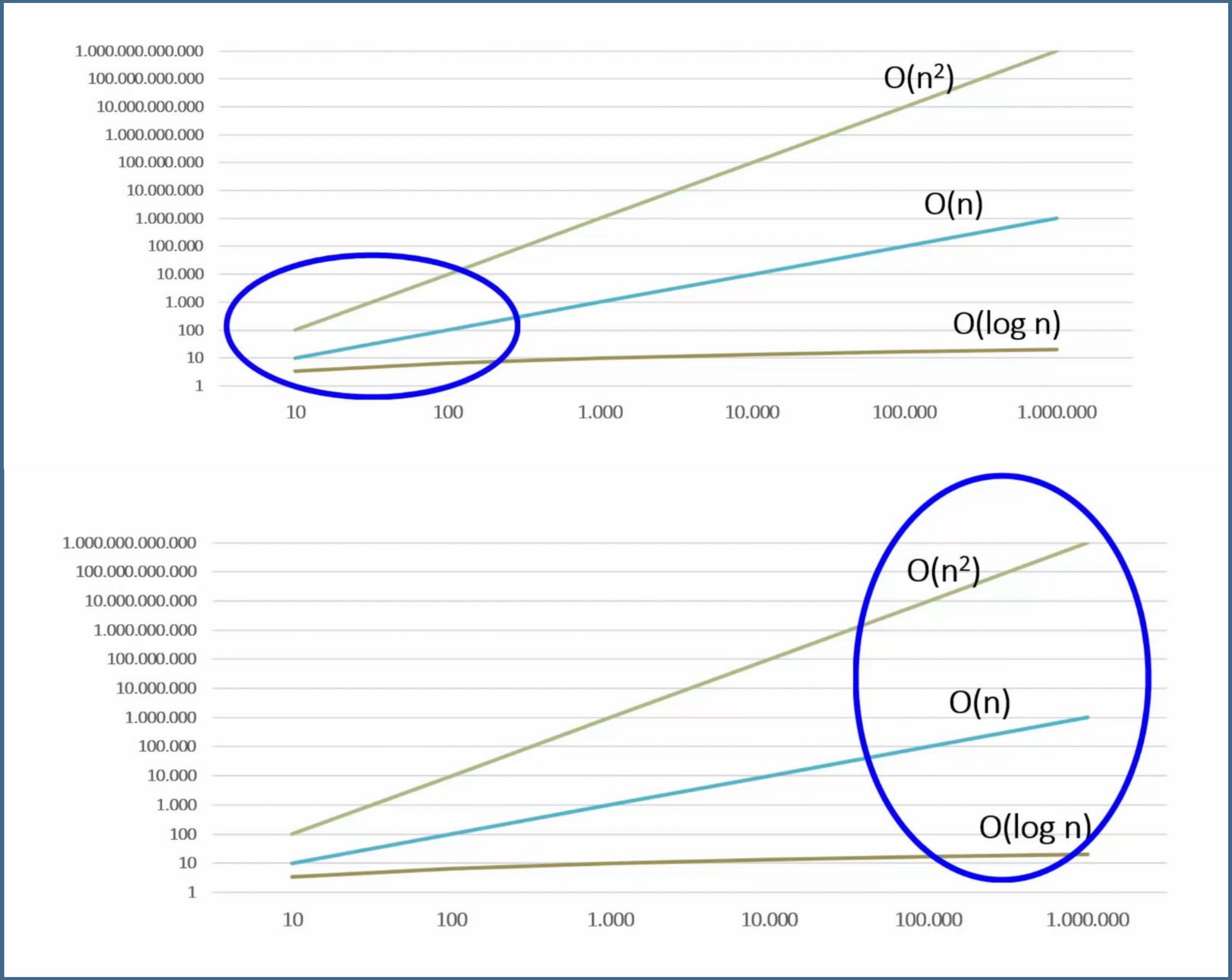
n=60 - $O(2^n)$ -> 19.215 minutos ~ 13 días

n=100 - $O(2^n)$ -> 40.196.936.841 años

UNRaf
UNIVERSIDAD
NACIONAL DE
RAFAELA

#IC

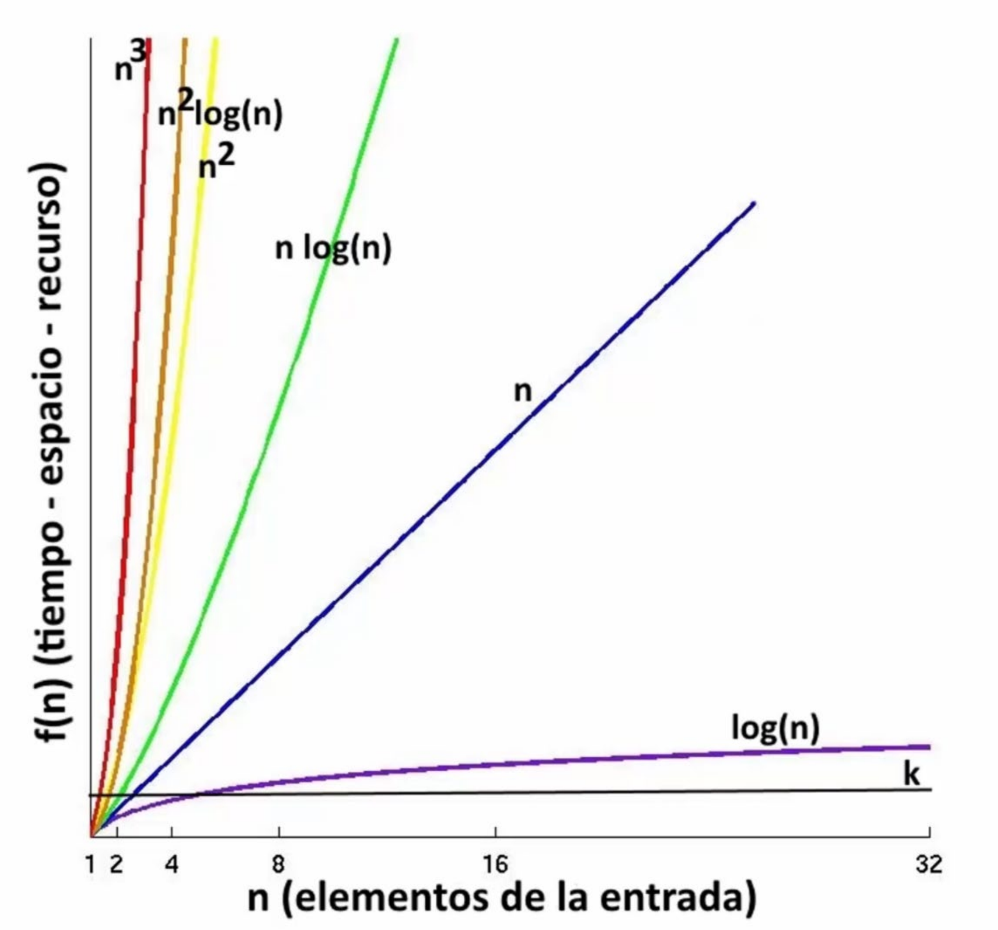
Orden de Magnitud



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Orden de Magnitud

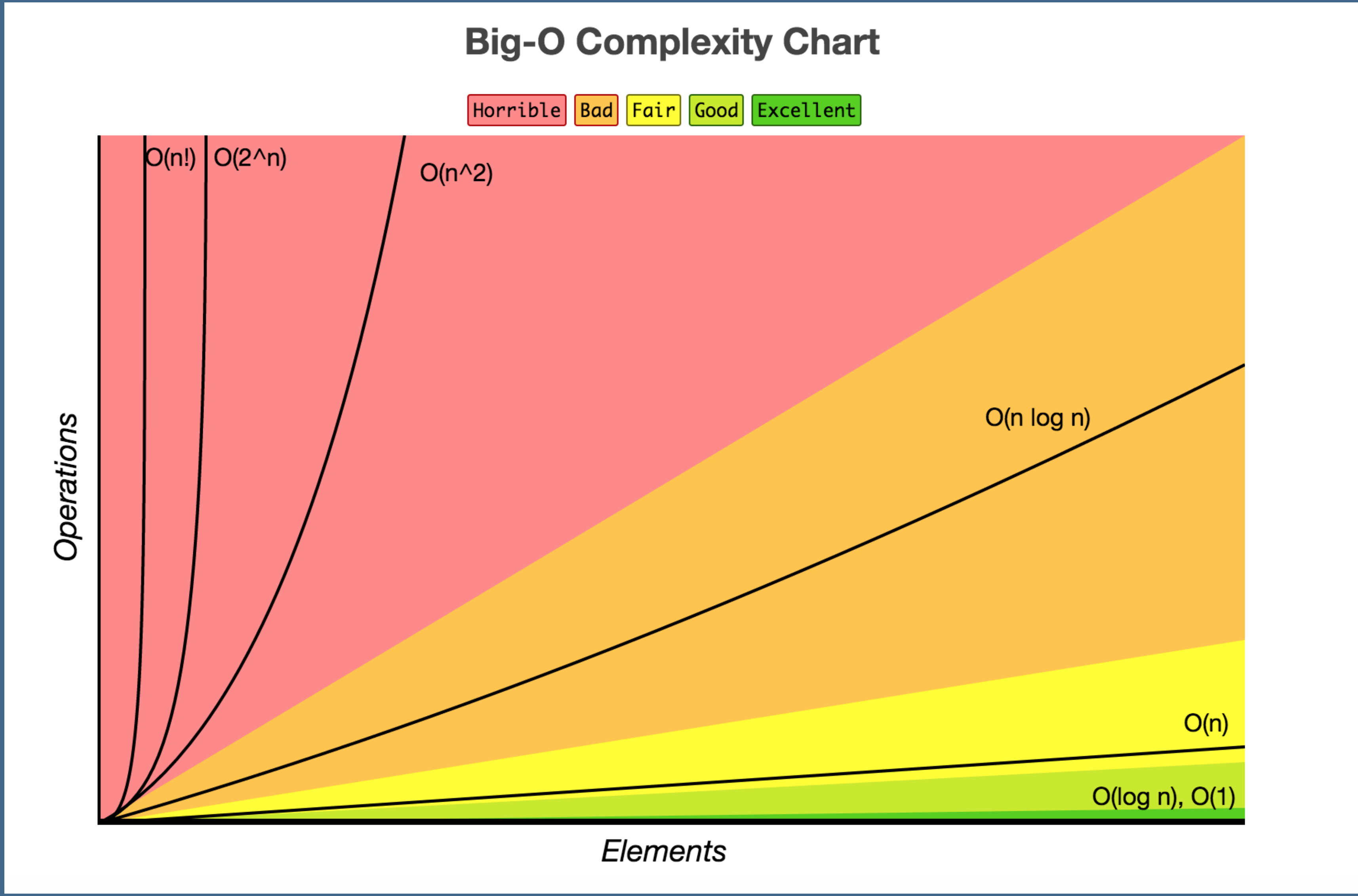


Big-O	Nombre
$O(k)$	Constante
$O(\log n)$	Logarítmica
$O(n)$	Lineal
$O(n \log n)$	Log-lineal
$O(n^2)$	Cuadrática
$O(n^3)$	Cúbica
$O(2^n)$	Exponencial
$O(n!)$	Factorial

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Orden de Magnitud



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

Orden de Magnitud

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	O(n log(n))	O(n log(n))	O(n^2)	O(log(n))
Mergesort	O(n log(n))	O(n log(n))	O(n log(n))	O(n)
Timsort	O(n)	O(n log(n))	O(n log(n))	O(n)
Heapsort	O(n log(n))	O(n log(n))	O(n log(n))	O(1)
Bubble Sort	O(n)	O(n^2)	O(n^2)	O(1)
Insertion Sort	O(n)	O(n^2)	O(n^2)	O(1)
Selection Sort	O(n^2)	O(n^2)	O(n^2)	O(1)
Shell Sort	O(n)	O((nlog(n))^2)	O((nlog(n))^2)	O(1)
Bucket Sort	O(n+k)	O(n+k)	O(n^2)	O(n)
Radix Sort	O(nk)	O(nk)	O(nk)	O(n+k)

Ejemplo con Biblioteca

#IC

```python
import big_o

def buscar_max(x):
    max = 0
    for nro in x:
        if nro > max:
            max = nro
    return max

nros_positivos = lambda n: big_o.datagen.integers(n, 0, 10000)

mejor, otras_soluciones = big_o.big_o(buscar_max, nros_positivos, n_repeats=100)

print(mejor)

for tipo_tiempo, residuo in otras_soluciones.items():
    print('{!s:<60s}    (res: {:.2G})'.format(tipo_tiempo, residuo))
```

```
Linear: time = 0.0029 + 1.3E-06*n (sec)
Constant: time = 0.067 (sec) (res: 0.017)
Linear: time = 0.0029 + 1.3E-06*n (sec) (res: 8E-05)
Quadratic: time = 0.025 + 1.2E-11*n^2 (sec) (res: 0.0016)
Cubic: time = 0.036 + 1.1E-16*n^3 (sec) (res: 0.0033)
Polynomial: time = -13 * x^0.97 (sec) (res: 0.03)
Logarithmic: time = -0.098 + 0.016*log(n) (sec) (res: 0.0065)
Linearithmic: time = 0.006 + 1.1E-07*n*log(n) (sec) (res: 0.00014)
Exponential: time = -5.6 * 4.5E-05^n (sec) (res: 15)
```

Ejemplo con POO

#IC

```python
import big_o

class MaxFinder:
    def __init__(self):
            self.mejor = None
            self.otras_soluciones = None

    def buscar_max(self, x):
        max_val = 0
        for nro in x:
            if nro > max_val:
                max_val = nro
        return max_val

    def generar_datos(self, n):
        return big_o.datagen.integers(n, 0, 10000)

    def calcular_complejidad(self, n_repeats=100):
        self.mejor, self.otras_soluciones = big_o.big_o(self.buscar_max, self.generar_datos, n_repeats=n_repeats)

    def mostrar_resultados(self):
        print(self.mejor)
        for tipo_tiempo, residuo in self.otras_soluciones.items():
            print('{!s:<60s}    (res: {:.2G})'.format(tipo_tiempo, residuo))
```

```python
max_finder = MaxFinder()

max_finder.calcular_complejidad()

max_finder.mostrar_resultados()
```

Ejemplo con POO

#IC

```
class Tiempo:
    def __init__ (self, h, m, s):
        self._hora = h
        self._minuto = m
        self._segundo = s

    def devolver_minutos( self):
        return self._minuto

    def __sub__(self, other):
        h = self._hora - other._hora
        m = self._minutos - other._minutos
        s = self._segundos - other._segundos
        return Tiempo(h, m, s)
```

Ejemplo con POO

#IC

```
class Progression:
    def __init__ (self, start=0):
        self._current = start

    def next (self):
        answer = self._current
        self._advance()
        return answer

    def _advance( self):
        self._current += 1

    def get_progression( self, n):
        answer = []
        for i in range(n):
            answer.append( self.next())
        return answer
```

Ejemplo con POO

#IC

```
class ArithmeticProgression(Progression):
    def __init__(self, increment=1, start=0):
        super( ). __init__(start)
        self._increment = increment

    def _advance( self):
        self._current += self._increment

class GeometricProgression(Progression):
    def __init__ (self, base=1, start=1):
        super(). __init__(start)
        self._base = base

    def _advance( self):
        self._current *= self._base
```

```
serie_generica = Progression()
serie_aritmetica = ArithmeticProgression(4)
serie_geometrica = GeometricProgression(4)

print (serie_generica.get_progression(10))
print (serie_aritmetica.get_progression(10))
print (serie_geometrica.get_progression(10))
```

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

¿?

UNRaf

UNIVERSIDAD
NACIONAL DE
RAFAELA

Bv. J.A. Roca 989 / CP: 2300
Rafaela - Santa Fe - Argentina

T: +54 (03492) **501155**

info@unraf.edu.ar
www.unraf.edu.ar