



Clase 4: 17 Agosto Actividad 1

🕒 Created	@August 17, 2023 6:23 PM
📅 Class	Algoritmos evolutivos 🐛

1. ¿Qué es un problema de optimización?

- Son aquellos problemas donde se busca encontrar la mejor solución o el mejor conjunto de soluciones al problema respetando ciertas restricciones y bajo ciertas condiciones.
- Donde basicamente se busca conseguir la mejor o el mejor conjunto de soluciones posibles para nuestro problema

2. Si un problema de optimización utiliza variables discretas. ¿Qué nombre recibe este problema?

- Combinatorios

3. ¿Qué diferencia hay entre una heurística y una metaheurística?

- Los algoritmos heurísticos buscan obtener una buena solución, que se podría decir como casi la óptima, esto en relación al tiempo computacional razonable. Las metaheurísticas son procedimientos que buscan la solución al problema con la menor cantidad de recursos computacionales, pero la diferencia es que estas pueden crear una que guía y modifica otras heurísticas para producir nuevas soluciones más allá de las que se generan comúnmente de las óptimas locales.

4. ¿Qué son las metaheurísticas basadas en población?

- Son un tipo de algoritmo de optimización basado más en procesos evolutivos naturales, la forma que funcionan, son primero con un conjunto de soluciones candidatas llamada "población" y se aplican operadores de búsqueda y selección para mejorar gradualmente la calidad de las soluciones a lo largo de iteraciones.
- Existen 3 tipos de ellas basadas en población:
 - EA → Algoritmos evolutivos
 - ACO → Colonias de Hormigas
 - PSO → Optimización por Enjambre de Partículas

5. ¿Qué son las metaheurísticas basadas en trayectoria?

- A comparación con los de población estos se enfocan en explotar y explorar el espacio de soluciones al modificar y ajustar continuamente una única solución a medida que se avanza a través de iteraciones, con estos se consiguen algoritmos que mejoran continua de nuestro algoritmos.
 - TS → Búsqueda Tabu.
 - ILS → Búsqueda Local Iterada
 - VNS → Búsqueda en Vecindarios Variables

6. ¿Cuál es la diferencia entre la complejidad de un algoritmo y la complejidad de un problema?

- La complejidad de un algoritmo se refiere a una medida de la eficiencia o ineficiencia de nuestro algoritmo según la cantidad de recursos como podría ser tiempo y espacio, estos a menudo se expresan con la notación Big O.
- Por otra parte la complejidad del problema, es más la complejidad del problema en sí, hablando que tan difícil es el problema por resolver, independientemente del algoritmo que se utilice. En este apartado entran punto con NP-Hard, NP Completo, NP y P, como formas para evaluarlos.

7. ¿Qué diferencia hay entre un problema del tipo P y uno del tipo NP?

- a. Problemas P → Existe un algoritmo eficiente que los resuelve.
- b. Problemas NP → Son los cuales es posible verificar una solución dada en tiempo polinómico.

8. **¿Cuáles son los pasos que sigue un algoritmo evolutivo para solucionar un problema?**

- a. Normalmente, el aprendizaje evolutivo puede resumirse en cuatro pasos:
 - i. Generar un conjunto inicial de soluciones (población)
 - ii. Reproducir nuevas soluciones basadas en la población actual
 - iii. Remover soluciones pobres en la población
 - iv. Repetir el proceso a partir del paso 2 hasta que se cumpla algún criterio de finalización.

9. **Se tienen los siguientes cromosomas padres. Si aplico el procedimiento de recombinación de un solo punto en la flecha marcada. ¿Cómo serían los cromosomas hijos?**

- a. Hijos

0	0	1	1	1	0	0	0
1	0	1	0	1	0	1	0

10. **A los cromosomas hijos de la pregunta 9, aplique el operador de mutación en el bit 7 a cada uno de ellos.**

0	1	1	1	1	0	0	0
1	1	1	0	1	0	1	0

11. **¿Qué es la función de aptitud?**

- a. También conocida como función de aptitud. Esta función matemática determina que "tan buena" es la solución (cromosoma) dentro del conjunto de soluciones (población).

12. **¿Qué son las restricciones en una función de aptitud?**

- a. Se refieren a las condiciones o limitaciones que se aplican a las soluciones candidatas posibles en un problema de optimización.

13. **Ejercicio**

```
def main():
    x = int(input("Cantidad de calificaciones a ingresar "))
    sum_grade = 0

    for i in range(x):
        y = int(input("Ingresa la calificación "))

        while y < 0 or y > 10:
            print("Error, ingrese nuevamente el valor ")
            y = int(input())

        sum_grade += y

    final_grade = sum_grade / x

    if final_grade >= 6:
        print("Felicidades, aprobaste ->", final_grade)
    else:
        print("Reprobado ->", final_grade)

if __name__ == "__main__":
    main()
```

14. **Ejercicio**

```

def main():
    numeros_suma_r = []
    numeros_suma_i = []
    numeros_resta_r = []
    numeros_resta_i = []
    numeros_mult_i = []
    numeros_mult_r = []
    print("1 -> Sumar")
    print("2 -> Restar")
    print("3 -> Multiplicar")
    print("4 -> Al cuadrado")
    x = int(input("Que tipo de operacion va a realizar? "))
    if( x == 1):
        for i in range(2):
            num1_suma = int(input("Ingresa los numeros reales -> "))
            numeros_suma_r.append(num1_suma)
        for i in range(2):
            num2_suma = int(input("Ingresa los numeros imaginarios-> "))
            numeros_suma_i.append(num2_suma)
        sumas_r_parcial = numeros_suma_r[0]+ numeros_suma_r[1]
        sumas_i_parcial = numeros_suma_i[0] + numeros_suma_i[1]

        print(sumas_r_parcial , str(sumas_i_parcial)+"i")
        sumas_totales = sumas_r_parcial , sumas_i_parcial
        #print(sumas_totales)
    if (x ==2):
        for i in range(2):
            num1_resta = int(input("Ingresa los numeros reales -> "))
            numeros_resta_r.append(num1_resta)
        for i in range(2):
            num2_resta = int(input("Ingresa los numeros imaginarios-> "))
            numeros_resta_i.append(num2_resta)

        resta_parcial_r = numeros_resta_r[0] - numeros_resta_r[1]
        resta_parcial_i = numeros_resta_i[0] - abs(numeros_resta_i[1])

        print(resta_parcial_r,str(resta_parcial_i)+"i")

    if (x ==3):
        for i in range(2):
            num1_mult = int(input("Ingresa los numeros reales -> "))
            numeros_mult_r.append(num1_mult)
        for i in range(2):
            num2_mult = int(input("Ingresa los numeros imaginarios-> "))
            numeros_mult_i.append(num2_mult)

        mult_parcial_r = (numeros_mult_r[0] * numeros_mult_r[1]) - (numeros_mult_i[0] * numeros_mult_i[1])
        mult_parcial_i = (numeros_mult_r[0] * numeros_mult_i[1]) + (numeros_mult_i[0] * numeros_mult_r[1])

        print(mult_parcial_r,str(mult_parcial_i)+"i")

    if (x ==4):
        num1_sq = int(input("Ingresa el numero real -> "))
        num2_sq = int(input("Ingresa el numero imaginario -> "))

        square_r = num1_sq * num1_sq - (num2_sq * num2_sq)
        square_i = 2 * num1_sq * num2_sq

        result_parcial_r = square_r * 2
        result_parcial_i = square_i * 2

        print(result_parcial_r,str(result_parcial_i)+"i")

if __name__ == "__main__":
    main()

```