Algoritmos de optimización - Seminario

Nombre y Apellidos: Mayra Alejandra Pullupaxi Ataballo
Url: https://github.com/MayAlejita/03MIAR_Algoritmos_de_Optimizacion
Problema:

> 1. Sesiones de doblaje
> 2. Organizar los horarios de partidos de La Liga
> 3. Combinar cifras y operaciones

Descripción del problema:

Combinar cifras y operaciones

El problema consiste en analizar el siguiente problema y diseñar un algoritmo que lo resuelva.

• Disponemos de las 9 cifras del 1 al 9 (excluimos el cero) y de los 4 signos básicos de las operaciones fundamentales: suma(+), resta(-), multiplicación(*) y división(/)

• Debemos combinarlos alternativamente sin repetir ninguno de ellos para obtener una cantidad dada. Un ejemplo sería para obtener el 4: 4+2-6/3*1 = 4

(*) La respuesta es obligatoria

(*) ¿Cuántas posibilidades hay sin tener en cuenta las restricciones?

Respuesta:

Para calcular cuantas posibilidades hay sin tener en cuenta las restricciones primero se debe tener en cuenta lo que se necesita, de un conjunto de m elementos en este caso se solicita que se tome en cuenta los número del 1-9, luego se elige solo n elementos (n < m) pudiendo repetirse, en nuestro ejemplo se requiere que sean 5 elementos. Este caso sería variaciones con repetición, y se calculan de la forma: m^n, de esta manera se obtiene (9^5).

A continuación se realiza el ejercicio en python y se muestra el valor.

```
#Se carga la librería products que permite realizar las combinaciones entre los núm
from itertools import product

#Se especifica el rango de números y cuántos van agrupados en cada conjunto de datc
lista = list(product(range(1,10), repeat = 5))
print(len(lista))
```

```
59049
```

¿Cuántas posibilidades hay teniendo en cuenta todas las restricciones.

Respuesta:

Para calcular cuantas posibilidades hay teniendo en cuenta que no se repitan los números al obtener la respuesta estamos ante una permutación en la cual tenemos un rango dado de números n y a la final quedará un conjunto de r elementos, para lo cual se necesita realizar el producto de n con n-1 y así hasta n-(r-1), de lo que se obtiene la fórmula de permutaciones: n!/(n-r)! => 9!/(9-5)!

A continuación se realiza el ejercicio en python y se muestra el valor.

```
#Se carga la librería permutations que permite realizar las permutaciones entre los
from itertools import permutations

lista_permutaciones = list(permutations(range(1,10), 5))
print(len(lista_permutaciones))
```

```
15120
```

Modelo para el espacio de soluciones

(*) ¿Cuál es la estructura de datos que mejor se adapta al problema? Arguméntalo.(Es posible que hayas elegido una al principio y veas la necesidad de cambiar, arguméntalo)

Respuesta

La estructura de datos que mejor se adapta al problema es un Dictionary formado por Arrays ya que se puede obtener las combinaciones posibles para luego solamente realizar validaciones y concatenar con los operadores matemáticos. Al inicio empecé a usar solo array pero para obtener el conjunto final de 5 elementos aumentaba cada vez la complejidad por el uso de gran cantidad de for.

Según el modelo para el espacio de soluciones

(*)¿Cual es la función objetivo?

Respuesta

Dado un conjunto de datos, se requiere obtener un resultado con los números sin que se repitan

Entonces la función objetivo sería: f(x, y) = x + y

(*)¿Es un problema de maximización o minimización?

Respuesta

Es un problema de maximización ya que quiere aprovechar o buscar la posible mejor solución en un conjunto extenso de números aleatorios al momento de dar un número y así le permita obtener los mayores resultados posibles con las restricciones dadas.

(*)Diseña un algoritmo que mejore la complejidad del algoritmo por fuerza bruta. Argumenta porque crees que mejora el algoritmo por fuerza bruta

```
#################################################################
# Definimos la función para obtener los valores dado un número
#################################################################

def Obtener_Numero(numero_dado):
  try:
    for i in lista:
      numero_obtenido = 0
      lista_repetidos = []
      operacion = []
      for index, j in enumerate(i):
        if j not in lista_repetidos:
          if index == 0:
            numero_obtenido = j
          else:
            numero_obtenido = Obtener_Operacion(j, numero_obtenido, operacion)
          lista_repetidos.append(j)
        else:
          break
        if (index == 4 and numero_dado == numero_obtenido):
          return print(f'El resultado es: {i[0]} {operacion[0]} {i[1]} {operacion[1
  except:
    print("Lo sentimos, no se puede procesar")

#################################################################
# Definimos la función para obtener la operación aritmética
#################################################################
def Obtener_Operacion(valor, numero, operacion):
  if '/' not in operacion:
    numero /= valor
    operacion.append('/')
  elif '*' not in operacion:
    numero *= valor
    operacion.append('*')
  elif '-' not in operacion:
    numero -= valor
    operacion.append('-')
```

```
  elif '+' not in operacion:
    numero += valor
    operacion.append('+')
  return numero

#################################################################
# LLamamos la función creada
#################################################################
Obtener_Numero(19)
```

```
    El resultado es: 2 / 1 * 7 - 3 + 8 = 19.0
```

(*)Calcula la complejidad del algoritmo

Respuesta :

La complejidad del algoritmo en la función es O(n^2), debido a que al utilizar la librería itertools se ahorra el realizar el número de iteraciones en cuyo caso la complejidad sería O(n^5)

Según el problema (y tenga sentido), diseña un juego de datos de entrada aleatorios

```
# Importamos la libreria para generar números aleatorios
import random
numeros_aleatorios = []

#Se obtiene un juego de 30 números aleatorios en una lista
for i in range(30):
  numeros_aleatorios.append(random.randrange(1, 77))

print(f'El juego de datos es: {numeros_aleatorios}')
```

```
    El juego de datos es: [56, 22, 64, 43, 72, 71, 24, 17, 53, 75, 40, 16, 26, 33
```

Aplica el algoritmo al juego de datos generado

```
#Aplicamos el juegoo de datos a nuestro algoritmo
for i in numeros_aleatorios:
  Obtener_Numero(i)
```

```
    El resultado es: 6 / 1 * 9 - 2 + 4 = 56.0
    El resultado es: 2 / 1 * 8 - 3 + 9 = 22.0
    El resultado es: 7 / 1 * 9 - 2 + 3 = 64.0
    El resultado es: 5 / 1 * 8 - 3 + 6 = 43.0
    El resultado es: 8 / 1 * 9 - 3 + 2 = 71.0
    El resultado es: 3 / 1 * 6 - 2 + 8 = 24.0
    El resultado es: 2 / 1 * 6 - 3 + 8 = 17.0
```

```
El resultado es: 6 / 1 * 8 - 2 + 7 = 53.0
El resultado es: 8 / 1 * 9 - 2 + 5 = 75.0
El resultado es: 4 / 1 * 9 - 2 + 6 = 40.0
El resultado es: 2 / 1 * 5 - 3 + 9 = 16.0
El resultado es: 3 / 1 * 7 - 4 + 9 = 26.0
El resultado es: 3 / 1 * 9 - 2 + 8 = 33.0
El resultado es: 7 / 1 * 9 - 2 + 3 = 64.0
El resultado es: 6 / 1 * 8 - 2 + 9 = 55.0
El resultado es: 3 / 1 * 9 - 2 + 8 = 33.0
El resultado es: 2 / 1 * 6 - 3 + 8 = 17.0
El resultado es: 6 / 1 * 9 - 2 + 7 = 59.0
El resultado es: 4 / 1 * 8 - 2 + 7 = 37.0
El resultado es: 3 / 1 * 9 - 2 + 7 = 32.0
El resultado es: 6 / 1 * 9 - 2 + 5 = 57.0
El resultado es: 3 / 1 * 9 - 2 + 7 = 32.0
El resultado es: 1 / 2 * 4 - 3 + 8 = 7.0
El resultado es: 4 / 1 * 8 - 2 + 9 = 39.0
El resultado es: 4 / 1 * 9 - 2 + 7 = 41.0
El resultado es: 2 / 1 * 5 - 3 + 9 = 16.0
El resultado es: 7 / 1 * 9 - 2 + 8 = 69.0
El resultado es: 1 / 2 * 6 - 4 + 3 = 2.0
El resultado es: 5 / 1 * 8 - 3 + 6 = 43.0
```

Enumera las referencias que has utilizado(si ha sido necesario) para llevar a cabo el trabajo

Respuesta

Se ha utilizado las siguientes referencias:

https://medium.com/@joseguillermo_/qu%C3%A9-es-la-complejidad-algor%C3%ADtmica-y-con-qu%C3%A9-se-come-2638e7fd9e8c

https://www.cs.us.es/~jalonso/cursos/i1m-19/temas/tema-28.html

✓ 1 s se ejecutó 18:14