AG2 – Actividad Guiada 2

03MIAR – Algoritmos de Optimización

Abrir el cuaderno de google colab:

https://colab.research.google.com/drive/15LKBrH_aHrbVfPCcAg6QFRkn5gC5-SL9?usp=sharing

viu Universidad Internacional de Valencia

AG2 – Actividad Guiada 2

Programación Dinámica

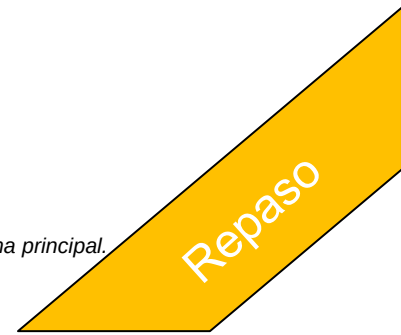
viu | Universidad Internacional de Valencia

Agenda

- Nuevo tema en el Foro
- Práctica: Programación dinámica(Viaje por el rio).
- Teoría: Ramificación y Poda
- Práctica: Búsqueda en grafos, ramificación y poda(asignación de tareas).
- Practica: Descenso del gradiente.

Programación dinámica (I)

- **Definición:** Es posible dividir el problema en subproblemas más pequeños, guardando las soluciones para ser utilizadas más adelante.
- Características que permiten identificar problemas aplicables:
  - ✓ Es posible almacenar soluciones de los subproblemas para ser reutilizadas.
  - ✓ Debe verificar el **principio de optimalidad** de Bellman: "*en una secuencia optima de decisiones, toda sub-secuencia también es óptima"* (*)
  - ✓ La necesidad de guardar la información acerca de las soluciones parciales unido a la recursividad provoca la necesidad de preocuparnos por la complejidad espacial (cuantos recursos de espacio usaremos)



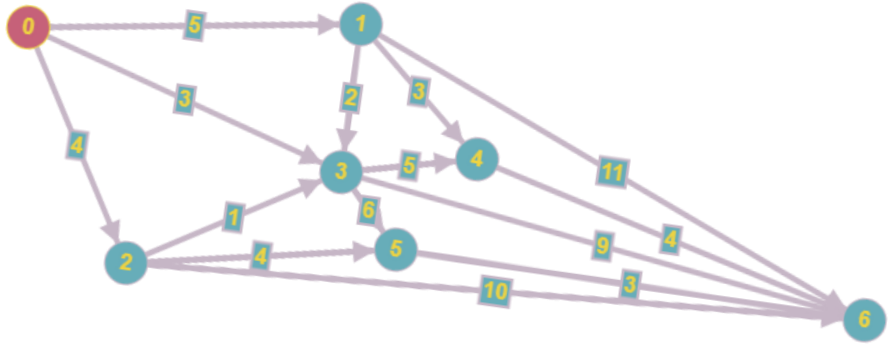
(*) Ojo! No significa que si tenemos soluciones optimas de los sub-problemas las podamos combinar para el optimo del problema principal.

Programación dinámica (II)

Problema: Viaje por el rio

- Consideramos una tabla T(i,j) para almacenar todos los precios que nos ofrecen los embarcaderos
- Si no es posible ir desde i a j daremos un valor alto para garantizar que ese trayecto no se va a elegir en la ruta óptima(modelado habitual para restricciones)
- Establecer una tabla intermedia(P(i,j)) para guardar soluciones óptimas parciales para ir desde i a j.

$$P(i,j) = \min \{T(i,j) , P(i,k)+T(k,j) \text{ para todo } i<k<=j \}$$

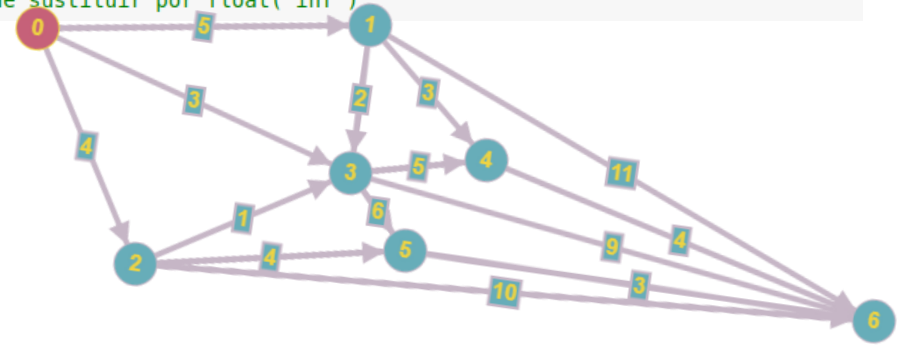


Programación dinámica (III)

Problema: Viaje por el rio

- Establecemos las tarifas:

```
#Viaje por el rio - Programación dinámica
##########################################################################

TARIFAS = [
[0,5,4,3,999,999,999],
[999,0,999,2,3,999,11],
[999,999, 0,1,999,4,10],
[999,999,999, 0,5,6,9],
[999,999, 999,999,0,999,4],
[999,999, 999,999,999,0,3],
[999,999,999,999,999,999,0]
]

#999 se puede sustituir por float("inf")
```



Programación dinámica (IV)

```
#Calculo de la matriz de PRECIOS y RUTAS
#  PRECIOS - contiene la matriz del mejor precio para ir de un nodo a otro
#  RUTAS   - contiene los nodos intermedios para ir de un nodo a otro
################################################################
def Precios(TARIFAS):
################################################################
  #Total de Nodos
  N = len(TARIFAS[0])

  #Inicialización de la tabla de precios
  PRECIOS = [ [9999]*N for i in [9999]*N]  #n x n
  RUTA = [ [""]*N for i in [""]*N]

  #Se recorren todos los nodos con dos bucles(origen - destino)
  #  para ir construyendo la matriz de PRECIOS
  for i in range(N-1):
    for j in range(i+1, N):
      MIN = TARIFAS[i][j]
      RUTA[i][j] = i

      for k in range(i, j):
        if PRECIOS[i][k] + TARIFAS[k][j] < MIN:
            MIN = min(MIN, PRECIOS[i][k] + TARIFAS[k][j] )
            RUTA[i][j] = k
        PRECIOS[i][j] = MIN

  return PRECIOS,RUTA
```

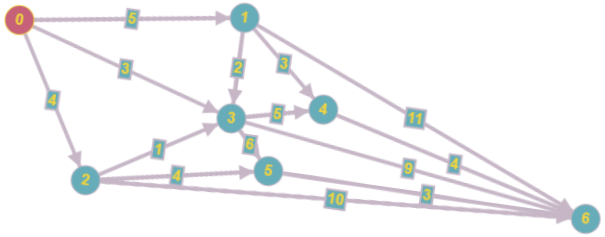
Operaciones

n

n

n

$3 \cdot n^3$




```
[ ]  TARIFAS = [
     [0,5,4,3,999,999,999],
     [999,0,999,2,3,999,11],
     [999,999, 0,1,999,4,10],
     [999,999,999, 0,5,6,9],
     [999,999, 999,999,0,999,4],
     [999,999, 999,999,999,0,3],
     [999,999,999,999,999,999,0]
     ]
```

(*) En lugar de 999

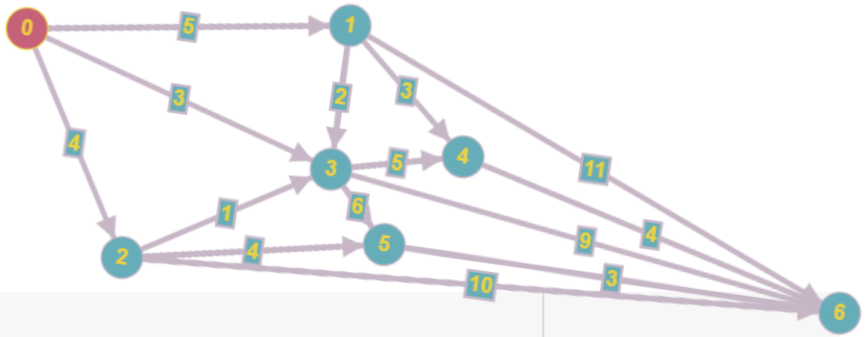
```
import math
math.inf
```

$$P(i,j) = \min \{T(i,j) , P(i,k)+T(k,j) \text{ para todo } i<k<=j \}$$

Programación dinámica (V)

- RUTA contiene la mejor opción intermedia para ir de un nodo a otro

```
RUTA
['', 0, 0, 0, 1, 2, 5]
['', '', 1, 1, 1, 3, 4]
['', '', '', 2, 3, 2, 5]
['', '', '', '', 3, 3, 3]
['', '', '', '', '', 4, 4]
['', '', '', '', '', '', 5]
['', '', '', '', '', '', '']
```




```
def calcular_ruta(RUTA, desde, hasta):
  if desde == hasta:
    #print("Ir a :" + str(desde))
    return desde
  else:
    return str(calcular_ruta(RUTA, desde, RUTA[desde][hasta]) ) +  ',' + str(RUTA[desde][hasta])

print("\nLa ruta es:")
calcular_ruta(RUTA, 0,6)
```

Recursividad

AG2 – Actividad Guiada 2

Descenso del Gradiente

Agenda

- Nuevo tema en el Foro
- Práctica: Programación dinámica(Viaje por el rio).
- Teoría: Ramificación y Poda
- Práctica: Búsqueda en grafos, ramificación y poda(asignación de tareas).
- Practica: Descenso del gradiente.

viu Universidad Internacional de Valencia

Descenso del gradiente. Práctica

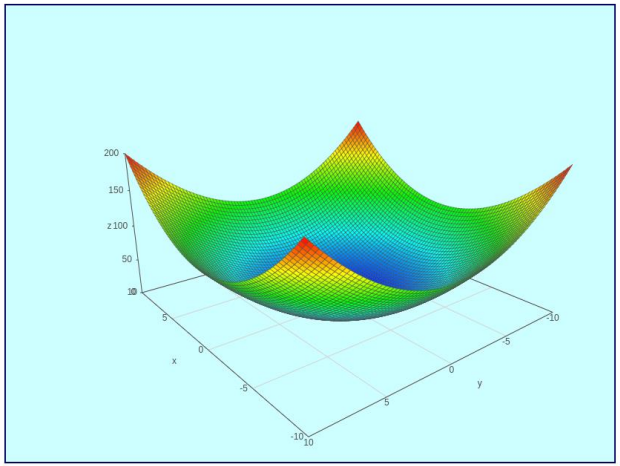
Preparar entorno

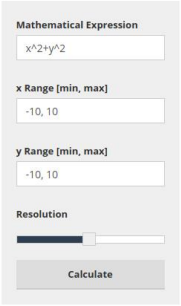
```
import math                          #Funciones matematicas
import matplotlib.pyplot as plt      #Generacion de gráficos (otra opcion seaborn)
import numpy as np                   #Tratamiento matriz N-dimensionales y otras (fundamental!)
#import scipy as sc

import random
```

Descenso del gradiente. Práctica

La función a minimizar. Paraboloide

```
f  = lambda X:      X[0]**2+X[1]**2      #Funcion
df = lambda X: [2*X[0] , 2*X[1]]         #Gradiente
```





http://al-roomi.org/3DPlot/index.html

Descenso del gradiente. Práctica

Prepara los datos para el gráfico

```
#Prepara los datos para dibujar mapa de niveles de Z
resolucion = 100
rango=2.5
X=np.linspace(-rango,rango,resolucion)
Y=np.linspace(-rango,rango,resolucion)
Z=np.zeros((resolucion,resolucion))
for ix,x in enumerate(X):
  for iy,y in enumerate(Y):
    Z[iy,ix] = f([x,y])

#Pinta el mapa de niveles de Z
plt.contourf(X,Y,Z,resolucion)
plt.colorbar()
```

Descenso del gradiente. Práctica

Generamos un Punto aleatorio

```
#Generamos un punto aleatorio
P=[random.uniform(-2,2  ),random.uniform(-2,2 ) ]
plt.plot(P[0],P[1],"o",c="white")
```

Descenso del gradiente. Práctica

Iteramos el algoritmo

```
#Tasa de aprendizaje
TA=.1

#Iteraciones
for _ in range(500):
  grad = df(P)
  #print(P,grad)
  P[0],P[1] = P[0] - TA*grad[0] , P[1] - TA*grad[1]
  plt.plot(P[0],P[1],"o",c="red")
```

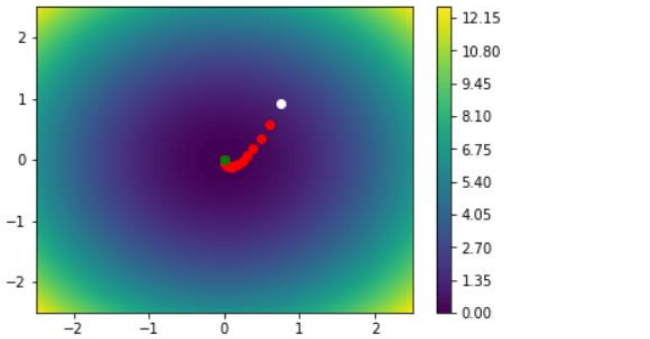
$$p_{t+1} = p_t - \alpha_t \nabla f(p_t)$$

Descenso del gradiente. Práctica

Pintamos el gráfico con las iteraciones

```
plt.plot(P[0],P[1],"o",c="green")
plt.show()
print("Solucion:" , P , f(P))
```

```
Punto Inicial: [0.7569403161678125, 0.9223403659681089]
```

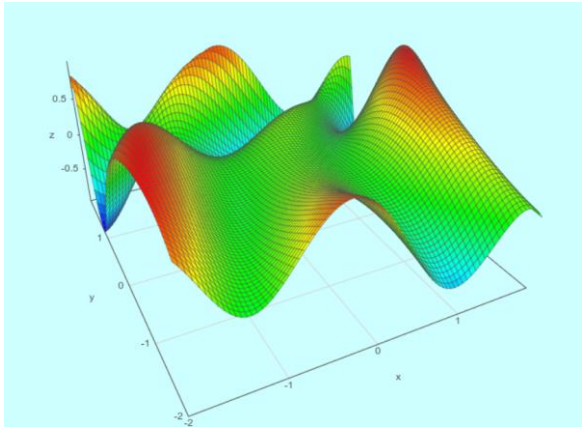



```
Solucion: [-0.004999999999999998, -0.005000000000000004] 5.000000000000002e-05
```

Descenso del gradiente. Práctica

Otra función a minimizar.

```
#Definimos la funcion
#sin(1/2 * x^2 - 1/4 * y^2 + 3) * cos(2*x + 1 - E^y)
f  = lambda X: np.sin(1/2 * X[0]**2 - 1/4 * X[1]**2 + 3) * np.cos(2 * X[0] + 1 - np.e**X[1])
```



sin(1/2 * x^2 - 1/4 * y^2 + 3) * cos(2*x + 1 - E^y)

Online 3D Function Visualizer: http://al-roomi.org/3DPlot/index.html

Descenso del gradiente. Práctica

¿Y el gradiente? !!


```
#Definimos la funcion
#sin(1/2 * x^2 - 1/4 * y^2 + 3) * cos(2*x + 1 - E^y)
f  = lambda X: np.sin(1/2 * X[0]**2 - 1/4 * X[1]**2 + 3) * np.cos(2 * X[0] + 1 - np.e**X[1])
```

```
#Aproximamos el valor del gradiente en un punto por su definición
def df(PUNTO):
  h = 0.01
  T = np.copy(PUNTO)
  grad = np.zeros(2)
  for it, th in enumerate(PUNTO):
    T[it] = T[it] + h
    grad[it] = (f(T) - f(PUNTO)) / h
  return grad
```

+1 para mejoar

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

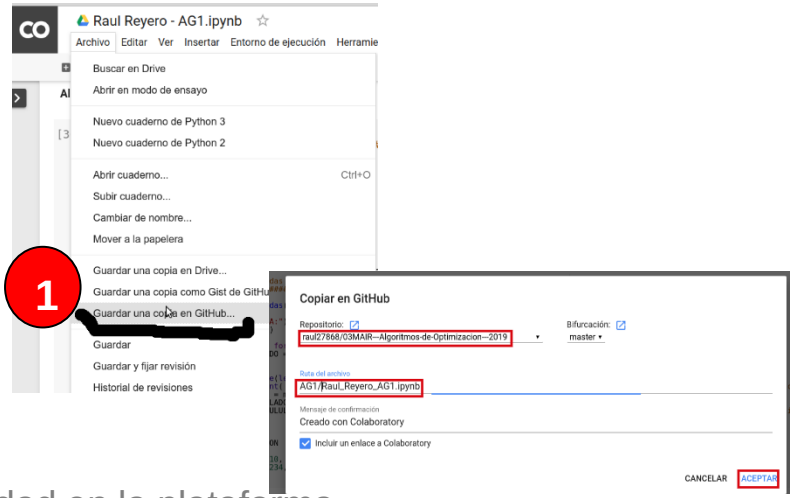
Aproximamos la derivada

Finalizar la actividad. Grabar, subir a GitHub, Generar pdf (I)

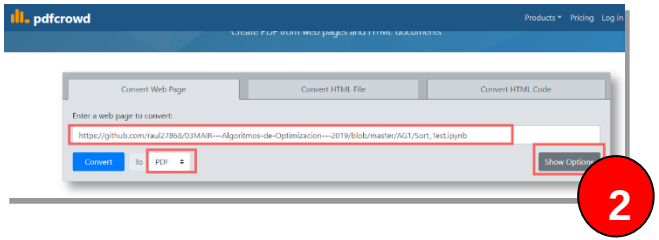
- Guardar en GitHub

  Repositorio: 03MIAR ---Algoritmos de Optimizacion

  Ruta de Archivo con **AG2**



- Generar pdf (con https://pdfcrowd.com)



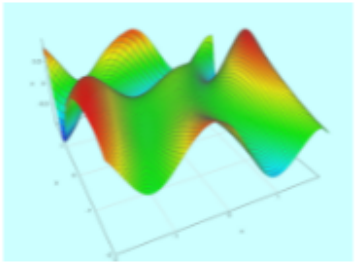
- Descargar pdf y adjuntar el documento generado a la actividad en la plataforma
  - Adjuntar .pdf en la actividad
  - URL GitHub en el texto del mensaje de la actividad

Descenso del gradiente. Reto

- Minimizar la función por descenso del gradiente

¿Te atreves a optimizar la función?:

$$f(x) = sin(1/2 * x^2 - 1/4 * y^2 + 3) * cos(2 * x + 1 - e^y)$$



Ampliación de conocimientos y habilidades

- Bibliografía

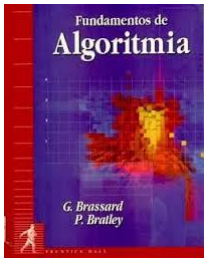
-Brassard, G., y Bratley, P. (1997). Fundamentos de algoritmia. *ISBN 13: 9788489660007*

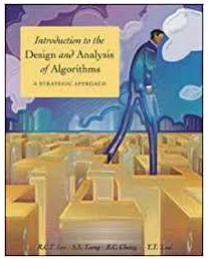
-Guerequeta, R., y Vallecillo, A. (2000). Técnicas de diseño de algoritmos.(http://www.lcc.uma.es/~av/Libro/indice.html)

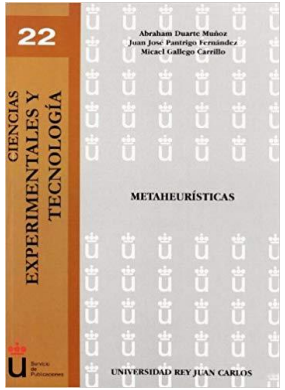
-Lee, R. C. T., Tseng, S. S., Chang, R. C., y Tsai, Y. T. (2005). Introducción al diseño y análisis de algoritmos. *ISBN 13: 9789701061244*

-***Abraham Duarte,..*** **Metaheurísticas.** ***ISBN 13: 9788498490169***

- Practicar







Gracias

juanfrancisco.vallalta@campusviu.es

viu Universidad Internacional de Valencia