Programación Avanzada
IIC2233 2025-2

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha

Anuncios

1. Publicación T4.
2. ¡El próximo jueves tendremos el cierre!

Tópicos Avanzados

Esta semana vimos muchas herramientas

- Expresiones regulares
- Grafos
- Busqueda en grafos
- Numpy
- Pandas

¿Por qué creen que son importantes?
Discutamos…

Expresiones Regulares (RegEx)

Expresiones Regulares

- ¿Qué son las regex?

Expresiones Regulares

- ¿Qué usamos para definir regex?

Meta-Caracteres: Clases de caracteres

Sintaxis	Descripción
`[ ]`	Clases de caracteres
`[0-9]`	Cualquier dígito entre 0 y 9
`[a-zA-Z]`	Cualquier letra entre A y Z, sin importar mayúscula o minúscula
`[^arn]`	Cualquiera EXCEPTO a, r, n

Sintaxis	Descripción
`.`	Cualquier carácter
`\d`	Cualquier dígito entre 0 y 9
`\s`	Cualquier espacio
`\w`	Caracteres alfanumérico y “_”
`\`	Escapar meta-carácter

Expresiones Regulares

- ¿Qué usamos para definir regex?

Meta-Caracteres: Cuantificadores

Sintaxis	Descripción
`+`	Está 1 o más veces
`*`	Está 0 o más veces
`?`	Está o no está
`{m, n}`	Está entre m y n veces, inclusive
`{m}`	Está m veces

Meta-Caracteres: Aserciones

Sintaxis	Descripción
`^`	Inicio del *string*
`$`	Fin del *string*

Expresiones Regulares

- ¿Qué usamos para definir regex?

Meta-Caracteres: Grupos y operación binaria

Sintaxis	Descripción
`|`	Operador **OR**
`( )`	Grupo o Grupo de captura
`(?P<nombre> )`	Grupo de captura **con nombre**

Regex en Python

- ¿Para qué podemos utilizar las regex?

```
re.match(patron, string) # Encontrar la primera ocurrencia desde el inicio
re.fullmatch(patron, string) # Todo el string cumple el patrón
re.search(patron, string) # Encontrar en cualquier lado el patrón
re.sub(patron, reemplazar_por, string) # Reemplazar un patrón
re.split(patron,string) # Separar según el patrón
```

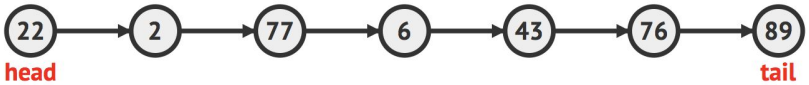
- ¿Qué es el objeto Match?

Estructuras nodales: grafos

Estructuras nodales

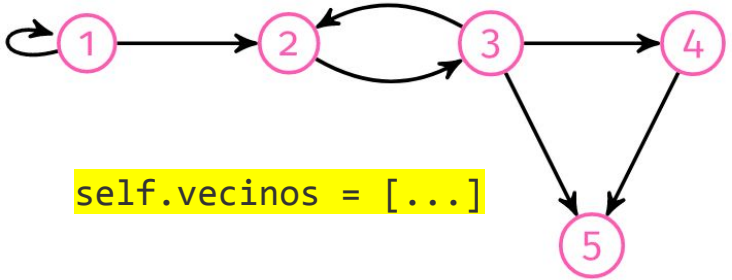
- ¿Cuáles son las diferencias y similitudes entre Lista Ligadas y Grafos?

ListaLigada




```
self.siguiente = ...
```

Grafo




```
self.vecinos = [...]
```

- Con lo anterior claro, ¿qué podemos representar mediante grafos?

Representación de Grafos

- ¿Cómo podemos representar los grafos?

Listas de Adyacencia

```
grafo = {
  1: [2, 3],
  2: [4, 5],
  3: [],
  4: [5],
  5: [4],
}
```

Matrices de Adyacencia

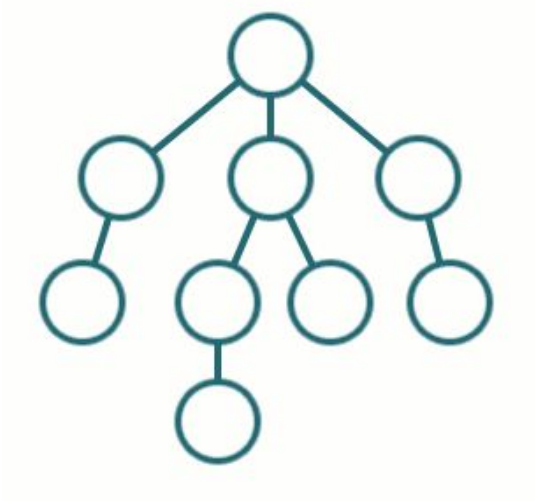
```
grafo = [
  [0, 1, 1, 0, 0],
  [0, 0, 0, 1, 1],
  [0, 0, 0, 0, 0],
  [0, 0, 0, 0, 1],
  [1, 0, 0, 1, 0]
]
```

Recorrido de grafos

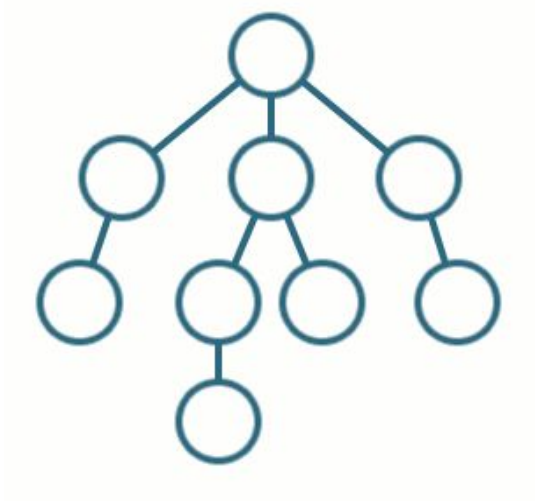
Recorrido de grafos

- ¿Por qué los grafos no se recorren igual que las Listas Ligadas?
- ¿Cuáles son los principales algoritmos para recorrer grafos?

Búsqueda por amplitud (BFS)



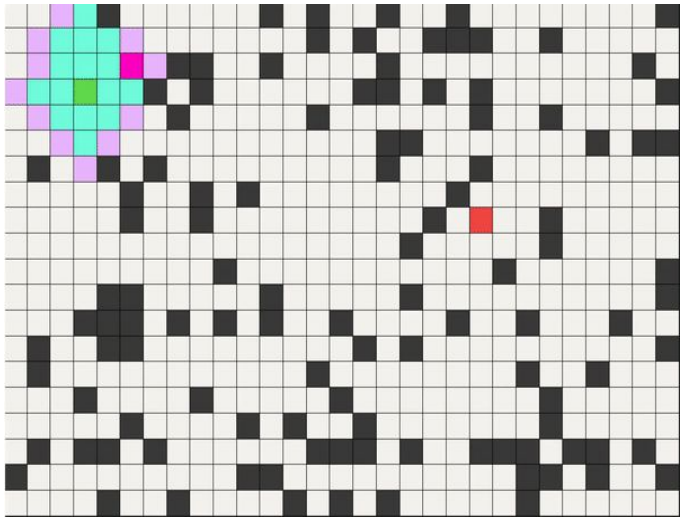
Búsqueda por profundidad (DFS)



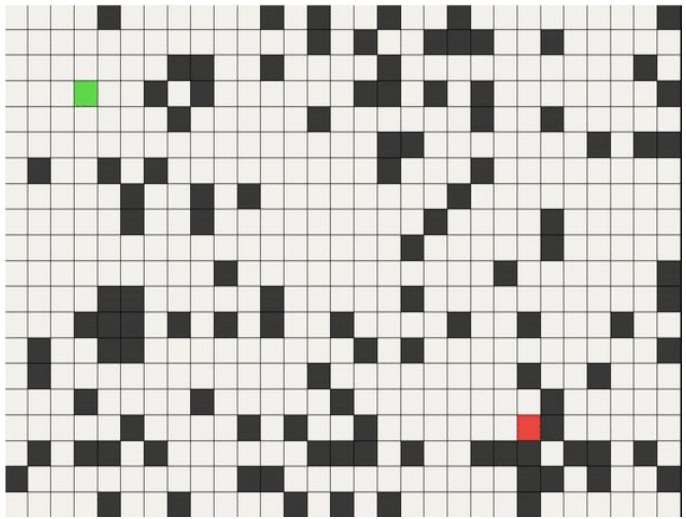
Recorrido de grafos

- ¿Qué caracteriza cada uno de estos algoritmos?
- ¿Qué estructuras usamos para implementar cada uno?

Búsqueda por amplitud (BFS)



Búsqueda por profundidad (DFS)



Numpy y Pandas

Numpy y Pandas

- ¿Qué son Numpy y Pandas?
- ¿Para qué se utiliza cada uno?

Uso de Numpy

- ¿Que son los Arrays de Numpy?

```
>>> np.array((1,2,3,4))               # Creamos array desde una tupla.
>>> np.array([1,2,3,4])               # Creamos array desde una lista.
>>> np.array([[1, 2, 3], [4, 5, 6]])  # Creamos matriz desde lista de listas.


>>> np.linspace(0, 1, 5)   # Crea array de 5 números equidistantes entre 0 y 1.

>>> np.zeros((3, 4))         # Crea matriz de 3x4 de ceros.
>>> np.ones((2, 3))          # Crea matriz de 2x3 de unos.
>>> np.eye(4)                # Crea matriz identidad de 4x4.
>>> np.random.random((2, 3)) # Crea matriz de 2x3 con núm. aleat. entre 0 y 1.
```

Uso de Numpy

- Permite realizar operaciones de Matrices y Vectores.

$$A \cdot B = A_1 \times B_1 + A_2 \times B_2 + \cdots + A_n \times B_n$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \rightarrow \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

main.py

```
1.  matriz1 = np.array([[1, 2], [3, 4]])
2.  matriz2 = np.array([[5, 6], [7, 8]])
3.  producto_punto = matriz1.dot(matriz2)
4.  print(producto_punto)
```

terminal

```
> python3 main.py
[[19 22]
 [43 50]]
```

Uso de Pandas

- ¿Qué son los DataFrames?

	Título	Año_publicacion	Autor
0	The Martian	2011	Andy Weir
1	Red Rising	2016	Pierce Brown
2	Mockingjay	2010	Suzanne Collins

```
1.  datos_libros = {
2.      'Título': ['The Martian', 'Red Rising', 'Mockingjay'],
3.      'Año_publicacion': [2011, 2016, 2010],
4.      'Autor': ['Andy Weir','Pierce Brown','Suzanne Collins']
5.  }
6.  df_libros = pd.DataFrame(datos_libros)
```

Uso de Pandas

- Obtener información

	Título	Año_publicacion	Autor
0	The Martian	2011	Andy Weir
1	Red Rising	2016	Pierce Brown
2	Mockingjay	2010	Suzanne Collins

En colab

```
df_libros.info() # Obtener información general.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Autor            3 non-null      object
 1   Año_publicacion  3 non-null      int64
 2   Título           3 non-null      object
dtypes: bool(1), int64(1), object(2)
```

Uso de Pandas

- Obtener información

	Título	Año_publicacion	Autor
0	The Martian	2011	Andy Weir
1	Red Rising	2016	Pierce Brown
2	Mockingjay	2010	Suzanne Collins

En colab

```
df_libros.head(2) # Ver las primeras 2 filas.
```

```
    Título       Año_publicacion  Autor
0   The Martian  2011             Andy Weir
1   Red Rising   2016             Pierce Brown
```

Uso de Pandas

- Obtener información

	Título	Año_publicacion	Autor
0	The Martian	2011	Andy Weir
1	Red Rising	2016	Pierce Brown
2	Mockingjay	2010	Suzanne Collins

En colab

```
df_libros.describe() # Obtener estadísticas descriptivas.
```

```
       Año_publicacion
count         3.000000
mean       2012.333333
std           3.214550
min        2010.000000
25%        2010.500000
50%        2011.000000
75%        2013.500000
max        2016.000000
```

Uso de Pandas

	Título	Año_publicacion	Autor
0	The Martian	2011	Andy Weir
1	Red Rising	2016	Pierce Brown
2	Mockingjay	2010	Suzanne Collins

- Acceder a todos los datos según su índice.

En colab

```
df_libros.iloc[2,0]
```

```
'Mockingjay'
```

- Acceder a todos los datos según su etiqueta.

En colab

```
df_libros.loc[1, 'Autor']
```

```
'Pierce Brown'
```

Uso de Pandas

- Acceder a toda una columna, usando su etiqueta.

	Título	Año_publicacion	Autor
0	The Martian	2011	Andy Weir
1	Red Rising	2016	Pierce Brown
2	Mockingjay	2010	Suzanne Collins

En colab

```
df_libros['Título']
```

```
    Título
0   The Martian
1   Red Rising
2   Mockingjay
```

Uso de Pandas

- Filtrar información

	Título	Año_publicacion	Autor
0	The Martian	2011	Andy Weir
1	Red Rising	2016	Pierce Brown
2	Mockingjay	2010	Suzanne Collins

En colab

```
df_andy_weir = df_libros[df_libros['Autor'] == 'Andy Weir']
print(df_andy_weir)
```

```
      Título  Año_publicacion      Autor
0  The Martian             2011  Andy Weir
```

Uso de Pandas

- Agregar una nueva columna

	Título	Año_publicacion	Autor
0	The Martian	2011	Andy Weir
1	Red Rising	2016	Pierce Brown
2	Mockingjay	2010	Suzanne Collins

En colab

```
df_libros.loc[:, 'Actual'] = df_libros['Año_publicacion'] > 2000
print(df_libros)
```

```
       Título  Año_publicacion            Autor   Actual
0  The Martian             2011        Andy Weir    True
1   Red Rising             2016     Pierce Brown    True
2   Mockingjay             2010  Suzanne Collins    True
```

Uso de Pandas

- Ordenar datos según columnas

	Título	Año_publicacion	Autor
0	The Martian	2011	Andy Weir
1	Red Rising	2016	Pierce Brown
2	Mockingjay	2010	Suzanne Collins

En colab

```
df_libros = df_libros[['Autor', 'Actual', 'Año_publicacion', 'Título']]
print(df_libros)
```

```
            Autor  Actual  Año_publicacion       Título
0        Andy Weir    True             2011  The Martian
1     Pierce Brown    True             2016   Red Rising
2  Suzanne Collins    True             2010   Mockingjay
```

Pregunta Evaluación Escrita

Pregunta de Evaluación Escrita

Tema: Expresiones Regulares (Examen 2023-2)

4. En expresiones regulares usamos: `( )` para agrupar, `*` para indicar que el carácter o grupo está 0 o más veces y `|` para definir un *or*. Dado lo anterior, ¿**cuántos** *substrings* hacen *match* entre la Frase y la Expresión?

   Frase: `'¡Aaahhh! La cama de mi casa está ocupada por mi gata'`

   Expresión: `r'a(s|m)*a'`

   A) 3
   B) 5
   C) 0
   D) 4
   E) 2

Pregunta de Evaluación Escrita

Tema: Expresiones Regulares (Examen 2023-2)

4. En expresiones regulares usamos: `( )` para agrupar, `*` para indicar que el carácter o grupo está 0 o más veces y `|` para definir un *or*. Dado lo anterior, ¿**cuántos** *substrings* hacen *match* entre la Frase y la Expresión?

   Frase: `'¡Aaahhh! La cama de mi casa está ocupada por mi gata'`

   Expresión: `r'a(s|m)*a'`

   A) 3

   B) 5

   C) 0

   D) 4

   E) 2

Pregunta de Evaluación Escrita

Tema: Grafos (Examen 2023-2)

21. Dado el siguiente grafo representado como una matriz de adyacencia:

```
matriz = [
    [0, 1, 0, 0],
    [1, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0]
]
```

¿Cuál de las siguientes operaciones se puede lograr sin uso de iteraciones (for/*while*) o recursión para ejecutarse?

I. Eliminar un enlace del grafo.
II. Agregar un enlace al grafo.
III. Verificar conexión directa entre 2 nodos.
IV. Encontrar un camino entre dos nodos que no están directamente conectados.

A) Solo IV B) I y II C) III y IV D) I, II y III E) I, II, III y IV

Pregunta de Evaluación Escrita

Tema: Grafos (Examen 2023-2)

21. Dado el siguiente grafo representado como una matriz de adyacencia:

```
matriz = [
    [0, 1, 0, 0],
    [1, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0]
]
```

¿Cuál de las siguientes operaciones se puede lograr sin uso de iteraciones (for/*while*) o recursión para ejecutarse?

I. Eliminar un enlace del grafo.

II. Agregar un enlace al grafo.

III. Verificar conexión directa entre 2 nodos.

IV. Encontrar un camino entre dos nodos que no están directamente conectados.

A) Solo IV B) I y II C) III y IV **D) I, II y III** E) I, II, III y IV

Programación Avanzada
IIC2233 2025-2

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha