Explicación del Código de Clasificación de Pares

Benjamín Campos, Javier Morales, Benjamín Cruz

Universidad San Sebastián

Introducción

Este documento explica un código Python que clasifica pares de números en distintas categorías para análisis de graphlets para el profesor Carlos Hernandez Ulloa.

Descripción del Código

El código Python tiene como objetivo procesar un archivo de pares de números y clasificar cada par según la frecuencia con que aparece, considerando también la inversión de los pares. Esta clasificación se traduce directamente en la **definición de los tipos de arco en un grafo**, útil para analizar graphlets en bioinformática.

- **Lectura del archivo:** El programa abre un archivo de texto donde cada línea contiene exactamente dos números separados por un espacio. Cada par de números (a, b) representa un arco del nodo a al nodo b en un grafo.
- **Clasificación de pares:** Se usan dos listas principales:
  - `lista1`: almacena pares que aparecen por primera vez, es decir, que aún no han sido registrados en ninguna lista. Estos representan arcos **unidireccionales iniciales** en el grafo.
  - `lista2`: almacena pares que aparecen por segunda vez, lo que indica que tanto el par original como su inverso existen en los datos. Estos se consideran arcos **bidireccionales** en el grafo.
- **Revisión de duplicados:** Por cada par (a, b) leído:
  1. Se genera su inverso (b, a).
  2. Si el par o su inverso ya está en `lista1`, significa que ya se había registrado previamente como un arco unidireccional. En este caso:
     - Se remueve de `lista1` (el original o su inverso, según corresponda).
     - Se agrega a `lista2`, indicando que ahora se tiene un arco bidireccional entre a y b.
  3. Si el par no estaba en ninguna lista, se agrega a `lista1`, considerando que es la primera aparición de este arco.
- **Asignación de tipo de arco:** Una vez finalizada la clasificación, el código imprime cada par con un tipo que indica cómo se representa el arco en el grafo:
  - **Tipo 1:** Aplica a pares en `lista1` (primer aparición). Representa un arco **unidireccional** del primer nodo al segundo.
  - **Tipo 2:** Aplica al inverso del par en `lista1`. Representa un arco **unidireccional invertido** del segundo nodo al primero.
  - **Tipo 3:** Aplica a pares en `lista2`, tanto al original como al inverso. Representa un arco **bidireccional**, es decir, ambos nodos apuntan entre sí.
- **Interpretación en el grafo:**
  - Por ejemplo, si existe el par $(1, 2)$ en `lista1`, significa que el nodo 1 apunta a nodo 2 (tipo de arco 1).
  - Si luego aparece $(2, 1)$, ambos pasan a `lista2`, y ahora $(1, 2)$ y $(2, 1)$ son de tipo 3, representando un arco bidireccional entre los nodos 1 y 2.
- **Flujo general del código:**
  1. Abrir archivo y leer línea por línea.
  2. Validar que cada línea tenga exactamente dos números.
  3. Convertir los números en enteros y formar el par y su inverso.
  4. Clasificar el par según su aparición en las listas.
  5. Finalmente, imprimir todos los pares con el tipo de arco correspondiente.

De esta manera, el código permite construir un grafo dirigido donde los tipos de arco reflejan la dirección y la bidireccionalidad de las relaciones entre nodos, todo a partir de un archivo de pares de números.

Código Python

```python
def procesar_pares(nombre_archivo):
    lista1 = []  # pares que aparecen una vez
    lista2 = []  # pares que aparecen dos veces
    with open(nombre_archivo, "r") as f:
        for linea in f:
            partes = linea.strip().split()
            if len(partes) != 2:
                continue
            a, b = map(int, partes)
            par = (a, b)
            par_inv = (b, a)
            if par in lista1 or par_inv in lista1:
                if par in lista1:
                    lista1.remove(par)
                else:
                    lista1.remove(par_inv)
                lista2.append(par)
            elif par not in lista2 and par_inv not in lista2:
                lista1.append(par)
    for par in lista1:
        a, b = par
        print(a, b, 1)
        print(b, a, 2)
    for par in lista2:
        a, b = par
        print(a, b, 3)
        print(b, a, 3)
```

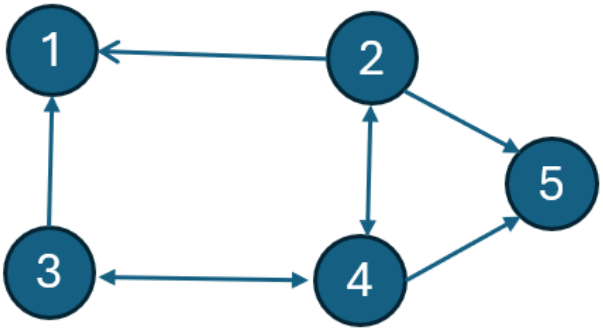
Ejemplo de Entrada

```
2 1
3 1
2 4
4 2
3 4
4 3
2 5
4 5
```

Resultados Esperados

```
2 1 1
1 2 2
3 1 1
1 3 2
2 4 3
4 2 3
3 4 3
4 3 3
2 5 1
5 2 2
4 5 1
5 4 2
```

Representación Gráfica



Conclusión

El código identifica automáticamente pares únicos y duplicados (considerando inversos), facilitando el análisis de graphlets en redes biológicas, con resultados claramente categorizados.