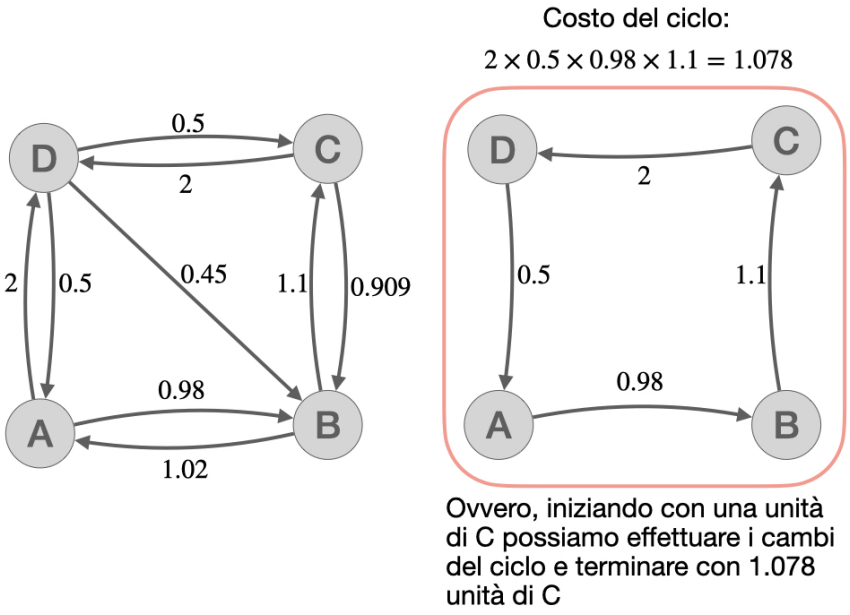
Progetto “Cambio valute”

Appello del 24 settembre 2021

Descrizione del progetto

Lo scopo del progetto è quello di realizzare un algoritmo che, dato un grafo diretto pesato rappresentate i tassi di cambio tra diverse valute, sia in grado di verificare se esistono dei cicli per i quali sia possibile guadagnare semplicemente effettuando dei cambi di valuta.

Un esempio di grafo in cui è presente un ciclo di questo tipo è il seguente:



Si noti come, rappresentando cambi di valuta, il “peso” di un percorso non sia dato dalla somma dei pesi degli archi che lo compongono ma dal prodotto. In particolare, ci si chiede se esiste un ciclo il cui prodotto degli archi che lo compongono è maggiore di uno.

Alcune considerazioni di cui tener conto sono le seguenti:

- Non effettuare nessun cambio di valuta ha peso 1 e non 0, ovvero la “distanza” tra il nodo i e sé stesso è 1 e non 0;
- Si ricorda che $\log(a \times b) = \log(a) + \log(b)$, quindi è possibile trasformare il problema di moltiplicare i pesi degli archi nel problema di *sommare* il logaritmo dei pesi degli archi;

- Dato che siamo interessati a massimizzare il prodotto, non useremo direttamente la somma dei logaritmi dei pesi, ma il loro negato;
- L'esistenza di cicli negativi implica l'esistenza di una sequenza di scambi di valute da cui è possibile guadagnare.

Si richiede di creare una classe `Grafo` che abbia i seguenti metodi:

- Creazione di un oggetto di tipo `Grafo` dati due argomenti:
  - Il numero di nodi del grafo (un intero). I nodi saranno indicati dagli interi da `1` a `n_nodes` (non da zero);
  - Una lista Python di triple di (i, j, w) dove i e j sono gli indici delle due estremità dell'arco e w è il suo peso.
- Un metodo `guadagno_possibile` che prende come argomento un intero tra `1` e `n_nodes` e ritorna un valore Booleano che indica se esiste un modo di guadagnare partendo da quella valuta.

Codice

Viene fornito uno scheletro del codice che deve essere implementato, con i metodi che devono essere scritti:

```
class Grafo:

    def __init__(self, n_nodes, edges):
        #...

    def guadagno_possibile(selg, nodo);
        #...
```

Nel progetto è consentito avere funzioni aggiuntive che testano il buon funzionamento delle funzionalità richieste. È anche consentito avere metodi aggiuntivi.

Si ricorda di commentare adeguatamente il codice, approfittandone per spiegare le scelte implementative effettuate.

Esempi d'uso

Il seguente frammento di codice chiama tutti i metodi la cui implementazione è richiesta dal progetto. Come commento è indicato il valore atteso in una implementazione funzionante:

```
def test():
    n_nodes = 4
    edges = [(1, 2, 0.98), (1, 4, 2), (2, 1, 1.02), (2, 3, 1.1),
             (3, 2, 0.909), (3, 4, 2), (4, 2, 0.45), (4, 3, 0.5), (4, 1, 0.5)]
    g = Grafo(n_nodes, edges)
    print(g.guadagno_possibile(1))  # True
    print(g.guadagno_possibile(2))  # True
    print(g.guadagno_possibile(3))  # True
```

```
print(g.guadagno_possibile(4))  # True
edges2 = [(1, 2, 0.25), (2, 1, 3), (3, 4, 1), (4, 3, 1.1)]
g2 = Grafo(n_nodes, edges2)
print(g2.guadagno_possibile(1))  # False
print(g2.guadagno_possibile(2))  # False
print(g2.guadagno_possibile(3))  # True
print(g2.guadagno_possibile(4))  # True
```

Indicazioni

Il progetto deve essere svolto **individualmente**. La consegna dovrà avvenire entro le ore 23:59 del giorno 19/09/2021 secondo le seguenti modalità:

- Invio di una email a `lmanzoni@units.it` dal vostro account email istituzionale con oggetto *[informatica] consegna progetto appello del 24/09/2021*.
- L'email deve avere come allegato il progetto in un singolo file in codice sorgente Python versione 3, dal nome `Nome_Cognome_matricola.py`, quindi, per esempio Mario Rossi di matricola 12345 consegnerà un file dal nome `Mario_Rossi_12345.py`.
- Il file deve contenere sotto forma di commento le seguenti linee indicanti nome, cognome e numero di matricola: `python # Nome: Mario # Cognome: Rossi # Matricola: 12345`