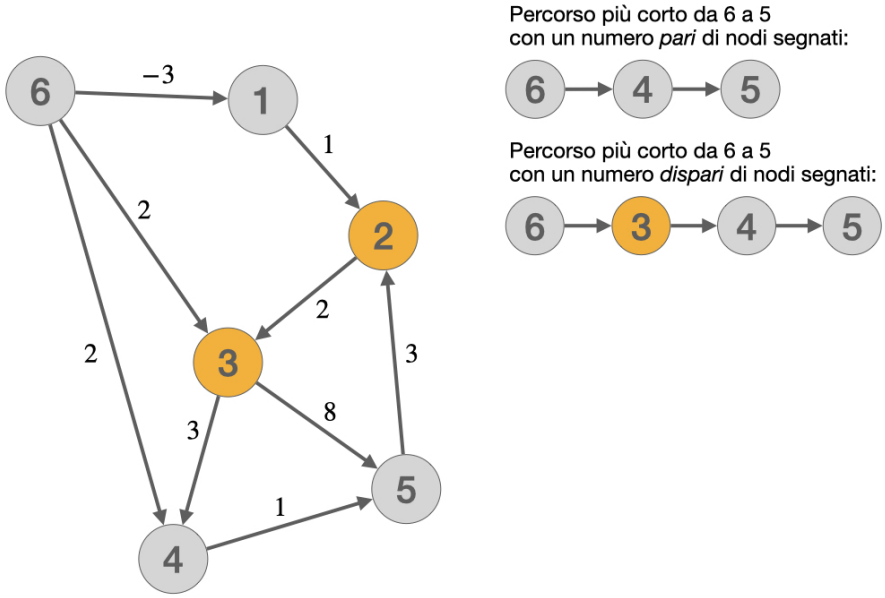
Progetto “percorsi con numero pari o dispari di nodi segnati”

Appello del 10 settembre 2021

Descrizione del progetto

Lo scopo del progetto è quello di realizzare un algoritmo su un grafo diretto pesato (con pesi che possono essere negativi) in cui alcuni nodi sono “segnati”. Si vuole trovare il percorso semplice (i.e., non contenente cicli) più corto contenente un numero pari di nodi segnati.

Un esempio di grafo in cui sono presenti due nodi segnati è il seguente:



Si noti come il percorso di peso minimo cambi a seconda che si richieda la presenza di un numero pari o dispari di nodi segnati.

Per calcolare questo percorso si possono fare alcune osservazioni. Denominiamo con $d_{i,j}^{k,\text{even}}$ il peso del percorso di peso minimo tra i nodi i e j contenente un numero pari di nodi segnati e passante per nodi intermedi di indice minore o uguale a k e con $d_{i,j}^{k,\text{odd}}$ l'equivalente per i percorsi con un numero *dispari* di nodi segnati. Allora abbiamo che:

- $d_{i,j}^{0,\text{even}} = 0$ solo se $i = j$ e il nodo i non è segnato, oppure se $i \neq j$ e o nessuno tra i e j è segnato oppure sia i che j sono segnati.

- $d_{i,j}^{0,\text{odd}} = 0$ solo se $i = j$ e il nodo i è segnato, oppure se $i \neq j$ e solo uno tra i e j è segnato.

Così come per l'algoritmo di Floyd–Warshall, è possibile ricostruite la distanza passando per i nodi fino a indice k usando le distanze per i nodi fino a indice $k-1$. Mostriamo qui il caso in cui il nodo k *non* sia segnato, il caso per k segnato è simile ma non uguale (bisogna tener conto del fatto che il nodo k non deve essere contato due volte per stabilire la parità del percorso):

- $d_{i,j}^{k,\text{even}} = \min\{d_{i,j}^{k-1,\text{even}}, d_{i,k}^{k-1,\text{even}} + d_{k,j}^{k-1,\text{even}}, d_{i,k}^{k-1,\text{odd}} + d_{k,j}^{k-1,\text{odd}}\}$.
- $d_{i,j}^{k,\text{odd}} = \min\{d_{i,j}^{k-1,\text{odd}}, d_{i,k}^{k-1,\text{odd}} + d_{k,j}^{k-1,\text{even}}, d_{i,k}^{k-1,\text{even}} + d_{k,j}^{k-1,\text{odd}}\}$.

Ovvero il percorso più corto (nel caso di percorso con numero pari di nodi segnati) potendo passare per i nodi da 1 a k si ottiene in uno di tre modi:

- Senza passare per k, quindi usando il percorso con numero pari di nodi segnati passando per al più i primi $k-1$ nodi
- Oppure "incollando" due percorsi con un numero pari di nodi
- Oppure "incollando" due percorsi con un numero dispari di nodi

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

- Creazione di un oggetto di tipo `Grafo` dati tre 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;
  - Una lista Python di interi rappresentanti i nodi segnati.
- Un metodo `distanza_pari` che ritorni il peso del percorso semplice di peso minimo contenente un numero pari di nodi segnati tra i due nodi i e j passati come argomenti.
- Un metodo `distanza_dispari` che ritorni il peso del percorso semplice di peso minimo contenente un numero dispari di nodi segnati tra i due nodi i e j passati come argomenti.

Si impongono inoltre i seguenti vincoli:

- Il calcolo del percorso di peso minimo tra due nodi deve avere al più complessità $O(|V|^3)$.

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, marked_nodes):
        #
```

```
    def distanza_pari(self, i, j):
        #

    def distanza_dispari(self, i, j):
        #
```

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 = 6
    edges = [(1, 2, 1), (2, 3, 2), (3, 4, 3),  (3, 5, 8), (4, 5, 1),
             (6, 1, -3), (6, 3, 2), (6, 4, 2), (5, 2, 3)]
    marked = [2, 3]
    g = Grafo(n_nodes, edges, marked)
    print(g.distanza_dispari(6, 5))  # 6
    print(g.distanza_pari(6, 5))  # 3
    print(g.distanza_dispari(6, 4))  # 5
    print(g.distanza_pari(6, 4))  # 2
    print(g.distanza_dispari(6, 3))  # 2
    print(g.distanza_pari(6, 3))  # 0
    print(g.distanza_dispari(1, 3))  # inf
    print(g.distanza_pari(1, 3))  # 3
    print(g.distanza_dispari(1, 4))  # inf
    print(g.distanza_pari(1, 4))  # 6
```

Indicazioni

Il progetto deve essere svolto **individualmente**. La consegna dovrà avvenire entro le ore 23:59 del giorno 03/09/2021 secondo le seguenti modalità:

- Invio di una email a `lmanzoni@units.it` dal vostro account email istituzionale con oggetto *[informatica] consegna progetto appello del 10/09/2021*.
- 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`