Computing Betweeness Centrality on Shortest Paths between Two Nodes in an Unweighted and Undirected Graph

Descrizione del problema:

Siano v1 e v2 due vertici di un grafo non orientato e non pesato, bisogna determinare e stampare a video, i nodi del grafo per i quali passano il maggior numero di cammini minimi che vanno da v1 a v2, ovvero i nodi con la betweenness score massima (se sono più di uno, avranno la stessa betweenness score massima).

Il grafo deve essere formato da nodi numerati partendo da 0 e deve essere costruito leggendo un file di testo strutturato nel seguente modo:
- nella prima riga deve esserci il numero totale di nodi
- nelle righe successive devono esserci i numeri di due nodi (separati da uno spazio), che dovranno essere collegati fra loro da un arco.

Il programma deve quindi ricevere 3 parametri:
- il nome del file di testo dal quale deve costruire il grafo;
- il numero del nodo v1;
- il numero del nodo v2.

Ricevuti i parametri, prima ancora di costruire il grafo, viene eseguito un controllo che verifica se il numero del nodo v1 e quello del nodo v2 sono uguali, e in tal caso il programma terminerà per ovvi motivi.
Nel caso contrario, il programma procede con la creazione del grafo utilizzando una lista di adiacenza. Di conseguenza lo spazio di memoria impiegato sarà $\Theta(|V|+|E|)$ e il costo computazione dell'algoritmo che la inizializza $\Theta(|V|+|E|)$.

Dopo aver creato il grafo è possibile controllare se v1 e v2 sono adiacenti, perché in tal caso il programma dovrà ovviamente terminare (prima di perdere tempo col cercare i nodi con betweenness score massima).

Per trovare i nodi con betweenness score massima, ho utilizzato due BFS leggermente modificate. La prima parte da v1, mentre la seconda parte da v2 e sfrutta le distanze di ogni nodo dalla sorgente, già calcolate dalla prima, per visitare solo i nodi che fanno parte dei cammini minimi fra v1 e v2. Inoltre, entrambe le BFS, calcolano (e salvano) per ogni nodo, il numero di cammini (fra v1 e v2), che passano per da esso. Dopo le due BFS, sarà sufficiente moltiplicare, per ogni nodo, il numero dei cammini calcolati con la prima BFS, per quelli calcolati con la seconda. Successivamente basta stampare a video il nodo o i nodi con betweeness score massima.
In sostanza, per contare i betweeness score di ogni nodo appartenente ai cammini minimi fra v1 e v2, ho prima calcolato il numero di cammini minimi che passano per ogni nodo, da v1 a v2, e li ho moltiplicati per quelli che vanno da v2 a v1, servendomi di due BFS.

Le due BFS modificate costano comunque $O(|V|+|E|)$, mentre l'algoritmo per calcolare i betwenness score moltiplicando i cammini di ogni nodo, trovare il massimo/i massimi, e stamparlo/i, costa $2|V|$, quindi $\Theta(|V|)$.

Pseudo-codice:

Algoritmo pre-BFS:

```
Foreach v ϵ G do
   v.dist  = ∞
   v.path1 = 0
   v.path2 = 0
   v.color = white
```

BFS1(G, v1, vett)

```
v1.dist = 0
Q = {v1}
while Q ≠ ∅ do
   u = head[Q]
   foreach v ϵ adj[u] do
      if v.dist == ∞ then
           v.dist = u.dist + 1
           Enqueue(Q, v)
      if u == v1 then
           vett[v]. path1 = 1
      else if (v.dist > u.dist) || (v.dist == ∞) then
                v.path1 = v.path1 + u.path1
   Dequeue(Q)
```

BFS2(G, v2, vett)

```
v2.color = black
Q = {v2}
while Q ≠ ∅ do
   u = head[Q]
   foreach v ϵ adj[u] do
      if v.dist == u.dist – 1 then
          if v.color == white then
              v.color = black
              Enqueue(Q, v)
          if u == v2 then
              v.path2 = 1
              else v.path2 = v.path2 + u.path2
   u.color = black
   Dequeue(Q)
```

Algoritmo per calcolare e stampare la betweenness score massima

```
bsMAX = 0
Foreach v ϵ G do
   v.path1 = v.path1 * v.path2
   if v.path1 > bsMAX then
   bsMAX = v.path1
Foreach v ϵ G do
   if v.path1 == bsMAX then
      print(v)
```