Presentazione algoritmo di ricerca bidirezionale di flusso senza ottimizzazione con stesso valore di nodi esplorati da source e da sink

Filippo Magi

February 19, 2022

1 strutture dati

1.1 BiEdge

L'arco BiEdge che collega due nodi, che ha le informazioni sulla capacità residua e sulla quantità di flusso inviata, oltre a un booleano per capire se durante l'invio del flusso deve inviarlo o ritirarlo.

1.2 Node

Il nodo contiene le informazioni sugli archi a lui collegati con una lista di BiEdge. Contiene un intero Label, che rappresenta la distanza tra quel nodo e s o t, a seconda da quale parte è stato esplorato. Tre booleani, il primo mi rappresentano se è stato esplorato da s o da t, il secondo se l'arco che lo collega con il nodo precedente (e quindi più vicino a s/t) ha capacità $= 0$ e l'ultimo mi indica se è stato esplorato o meno (nell'iterazione corrente, se fosse necessario resettarlo) Inoltre ci sono le informazioni di indirizzamento (nextNode e nextEdge per i nodi esplorati da t, previousNode e previousEdge per i nodi esplorati da s).

1.3 Graph

Graph è rappresentato da due insiemi, uno contenente i nodi esplorati da source, l'altro contenente i nodi esplorati da sink.

2 Descrizione

Effettuo la bfs, mi restituisce un nodo che, tramite le informazioni di indirizzamento, riesco a ottenere un percorso. Calcolo il flusso inviabile attraverso quel percorso. Invio il flusso nel percorso, salvando dove si creano i colli di bottiglia. Nel caso non mi venga restituito un nodo (null) o il flusso inviabile sia zero, termino l'algoritmo.

2.1 Bfs

Ricevo in input due booleani che mi indicano se devo esplorare solo la parte esplorata da source, da sink o se devo esplorarle entrambe. Cancello le informazioni in quella parte di grafo e accodo nell'apposita coda il nodo iniziale (cioè il nodo sorgente s o il nodo t). Finché entrambe le code non sono vuote, proseguo come segue

1. se la coda source non è vuota, mentre o la coda di archi di source è vuota oppure sia la coda di sink sia la coda di archi di sink sono vuote, proseguo, altrimenti passo al punto 4
2. seleziono il valore *elementSource* attraverso la dequeue della coda di source
3. per ogni arco di *elementSource*, seleziono quelli con nodi esplorabili e li inserisco nella coda degli archi di source
4. ripeto i precedenti tre punti per la coda di sink, salvandomi *elementSink*
5. finché entrambe le code di archi non sono vuote, esploro i nodi selezionando un arco alla volta, altrimenti torno al punto 1
6. nel caso in cui trovasi un nodo esplorato dall'altra parte, significa che ho trovato un percorso, aggiorno i dati e restituisco il valore.

Nel caso non trovi un percorso, restituisco null