Presentazione algoritmo di ricerca bidirezionale di flusso con divisione per label

Filippo Magi

February 18, 2022

1 strutture dati

1.1 BiEdge

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 node contiene tutti gli archi a lui collegato come una lista di BiEdge, inoltre contiene informazioni per quanto riguarda il proprio indirizzamento per l'invio del flusso, tramite gli attributi previousNode e previousEdge per quanto riguarda la parte esplorata dal nodo sorgente s, e nextNode e nextEdge per quanto riguarda la parte esplorata dal nodo destinazione t, e un booleano sourceSide, per capire da chi è stato esplorato. Inoltre tiene conto della label, con la distanza da s (t ha il valore massimo consentito, e esplorando faccio diminuire il valore, i valori andrebbero corretti).

Infine contiene un attributo per tenere traccia del flusso passante per quel nodo, cioè InFlow (usato anche per vedere se è stato esplorato o meno).

1.3 Graph

Contiene due insiemi di nodi, uno contenente i nodi esplorati da s e uno quelli esplorati da t.

2 descrizione

L'algoritmo svolge una BfsBidirezionale nella ricerca che un nodo sia in comune (descritto in seguito), dopo aver trovato il nodo dove si incontrano, proseguo a inviare il flusso, nel percorso descritto da nextNode e previousNode, salvando dove in quale delle due parti si almeno un arco la capacità è diventata pari a 0, rimuovendo anche il valore di InFlow (il valore del nodo dove si incontrano

le due ricerche viene modificato a dovere). Andrò ad analizzare solo quella parte nella seguente iterazione dell'algoritmo (se è presente in entrambe le parti naturalmente le faccio entrambe). Finché riesco a trovare un percorso proseguo ripeto la Bfs e l'invio del flusso.

2.1 Ricerca del percorso

Analizzo quale delle due parti devo analizzare, faccio un reset (elimino informazioni di indirizzamento e InFlow) di quella parte (o di entrambe se necessario) e aggiungo il nodo iniziale alla coda (s, t o entrambi). Finché entrambe le code non sono vuote, procedo ad analizzare tutti i nodi,ancora da esplorare, collegati al nodo che ho ottenuto dalla dequeue della coda interessata (per poi accodarli alla coda usata per ottenere quel valore), per poi passare all'altra coda, se possibile. Quando un nodo incontra un nodo esplorato "dall'altra parte", restituisco il valore del nodo (aggiornato) appartenente ai nodi esplorati da Sink (scelta arbitraria fatta).