Presentazione algoritmo di ricerca bidirezionale di flusso con divisione per label

Filippo Magi

February 18, 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. Inoltre salva la propria distanza dal nodo sorgente s (il nodo destinazione t ha il valore massimo consentito (int.MaxValue), quindi c'è un gap tra i due dati, andrebbe, in fase finale, andrebbero corrette le label), le informazioni per l'indirizzamento (previousEdge e previousNode per la parte esplorata da s, nextEdge e nextNode per la parte esplorata da t). Infine, ci sono due booleani, uni che mi indica se il nodo è stato precedentemente esplorato o meno, l'altro se è stato esplorato da s o da t.

1.3 Graph

Il grafo è rappresentato da due insieme di nodi, in uno sono presenti quelli esplorati da source, nel secondo quelli esplorati da sink.

2 descrizione

Effettuo una bfs che sia basa sulla label (descritta in seguito), dicendogli se deve lavorare sulla parte di source o su quella di sink (o su entrambe), per poi capire la quantità di flusso inviabile e, se diversa da 0, inviarlo, salvandomi dove ci sono archi con capacità residua pari a zero (quindi se devo esplorare la parte source o la parte sink). Se non trovo il percorso o il flusso inviabile è pari a 0, termino l'algoritmo.

2.1 bfs

Riceve in input due booleani, *sourceSide* e *sinkSide* per indicare quale parte del grafo devo esplorare. Per ogni nodo di quella parte,cancello i dati di indirizzamento e indico di non averlo caricato, inoltre carico nella coda apposita s e/o t. Esploro tutti i nodi presenti nella coda apposita, caricando i nodi esplorati in una coda buffer (onde evitare di esplorarli in quella esecuzione). Dopo aver "esplorato" tutti gli elementi presenti nella coda, faccio uno swap (a livello di puntatore) tra buffer (con i nodi interessati) e la coda usata (vuota), per poi procedere come la coda successiva, che sarà,se presente, della parte opposta. Per l'esplorazione, consideriamo di ottenere dalla coda un nodo n, se un arco è uscente da n e ha capacità residua positiva, controllo il nodo a lui collegato:

- nel caso in cui stato esplorato dalla stessa parte, analizzo l'arco successivo
- nel caso in cui sia stato esplorato dalla parte opposta, vuol dire che ho trovato un percorso
- altrimenti aggiorno i dati del nodo trovato e lo inserisco nella coda $buffer$

Svolgo la stessa cosa per gli archi entranti in n con flusso positivo