Presentazioni dell'algoritmo di ricerca del flusso massimo monodirezionale con propagazione della malattia

Filippo Magi

February 17, 2022

1 Strutture dati

1.1 BiEdge

I nodi vengono collegati tra di loro da archi BiEdge, che contiene le informazioni da quale nodo esce e in quale nodo entra. Ovviamente conserva in memoria la quantità di flusso che passa e la sua capacità residua, oltre a un booleano (reversed) che mi indica che quel arco, durante l'invio del flusso, dovrà inviarlo o ritirarlo.

1.2 Node

Node contiene al suo interno informazioni sugli archi a lui collegati (Lista di BiEdge), un booleano che indica se il nodo è valido o meno, le indicazioni sul percorso che deve fare, quindi sia previousNode e previousEdge, la label, cioè la distanza tra lui e il nodo sorgente s, oltre a un booleano che mi indica se è stato visitato o meno.

1.3 Graph

Contiene un insieme per i nodi non validi e una lista di insiemi, tale che ogni insieme contiene i nodi con una certa label.

2 Descrizione

il funzionamento è molto simile a quello di LastLevelOpt, con una differenza in fase di "inizializzazione". Provo a riparare tutti i nodi malati di **noCaps**, se riesco a ripararli tutti, procedo dichiarando di aver trovato il percorso, nel caso in cui un nodo non sia riparabile, lo inserisco in una coda di **malati**. Per ogni nodo $\in$ **malati**, procedo a sottoporlo a SickPropagation, nel caso riesco a trovare un percorso, restituisco il valore del flusso inviabile, altrimenti procedo

con il nuovo nodo. Se nessuna propagazione della malattia mi ha portato a un percorso, controllo se t è valido, nel caso lo sia, ho automaticamente trovato un percorso. Nel caso in cui t non sia più valido cancello tutte le informazioni dai nodi con label pari o maggiore a quella del primo nodo **first** esplorato dalla coda di **malati** . Nel caso in cui SickPropagation non abbia già inserito dei nodi in coda, vi inserisco i nodi $n|n.label = first.label - 1$, poi procedo con la ricerca di t normalmente, come in LastLevelOpt.

2.1 SickPropagation

Datomi il nodo n, lo inserisco in una coda *malati*. Provo a ripararlo, nel caso in cui riesca, lo inserisco nella coda, che verrà poi utilizzata per la ricerca del cammino, a meno che non sia il nodo destinazione t, in tal caso restituisco il valore del flusso inviabile attraverso il cammino descritto dai previousNode. Nel caso non sia possibile ripararlo,inserisco tutti i nodi adiacenti che sono stati precedentemente esplorati da lui, cioè con $malati.enqueue(x \in A(n)|x.previousNode = n)$. Ripeto finché o riparo il nodo t, o si esaurisce la coda di *malati*.