PROG2 15 06 22

Liste di adiacenza (codice)

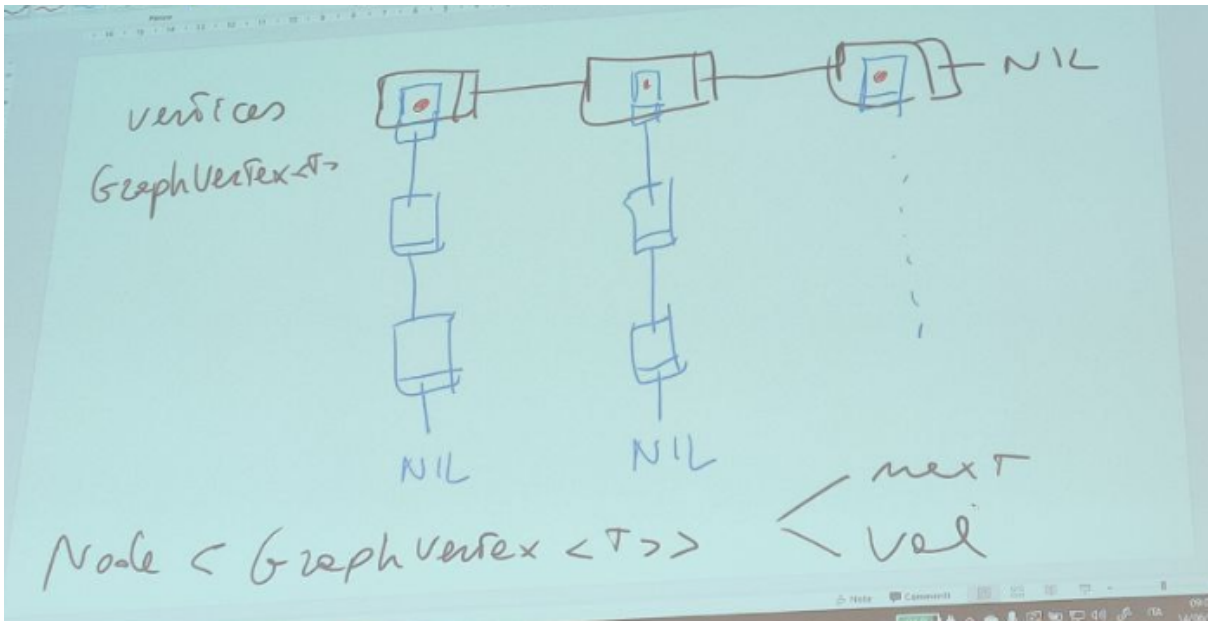
Se orientato, ho 2E archi (da i a j e da j a i)

Se non orientato, ho E archi (da i a j)

Usando una lista gia implementata, in testa ho la chiave del vertice e successivamente ci sono i nodi adiacenti

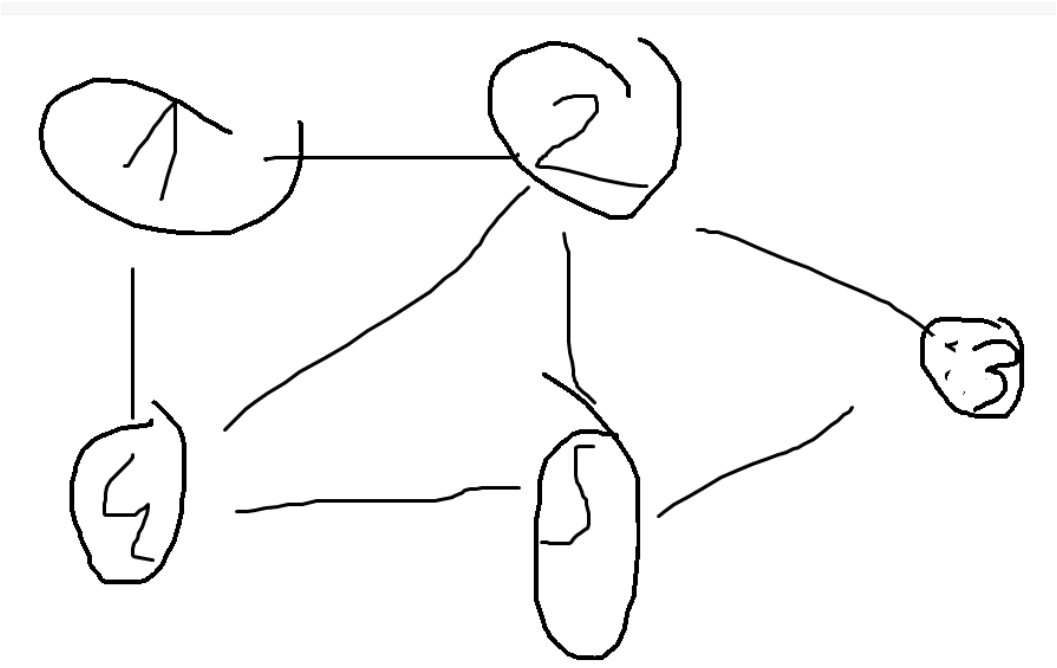
search()

Ho una lista fatta di nodi di tipo GraphVertex<T>. Essa è una lista di una lista. Quando si cerca un valore, ci interessa la head all'interno della lista contenuta nel nodo.



`if(key==ptr->val.getHead()->val)` vedo se la chiave è uguale alla testa della lista blu.

Prendiamo un grafo di esempio



Visita in ampiezza di un grafo (breadth-first search)BFS

può partire da un nodo qualunque. Da dove si parte? si parte da un nodo(casuale) chiamato sorgente

In ampiezza perchè la visita procede per distanza dal nodo sorgente. Quindi al primo passo visito tutti i nodi a distanza 1. Al passo k avrò scoperto nodi a distanza k dal nodo sorgente.

convenzione di colori:

Bianco=nodo non scoperto, non visitato

dopo essere stato scoperto può essere “scoperto ma non totalmente esplorato” e “Scoperto totalmente”

grigio nodo esplorato parzialmente

nero nodo esplorato

Inizialmente tutti i nodi sono bianchi e tutti i nodi hanno INFINITO al loro interno (inizialmente)

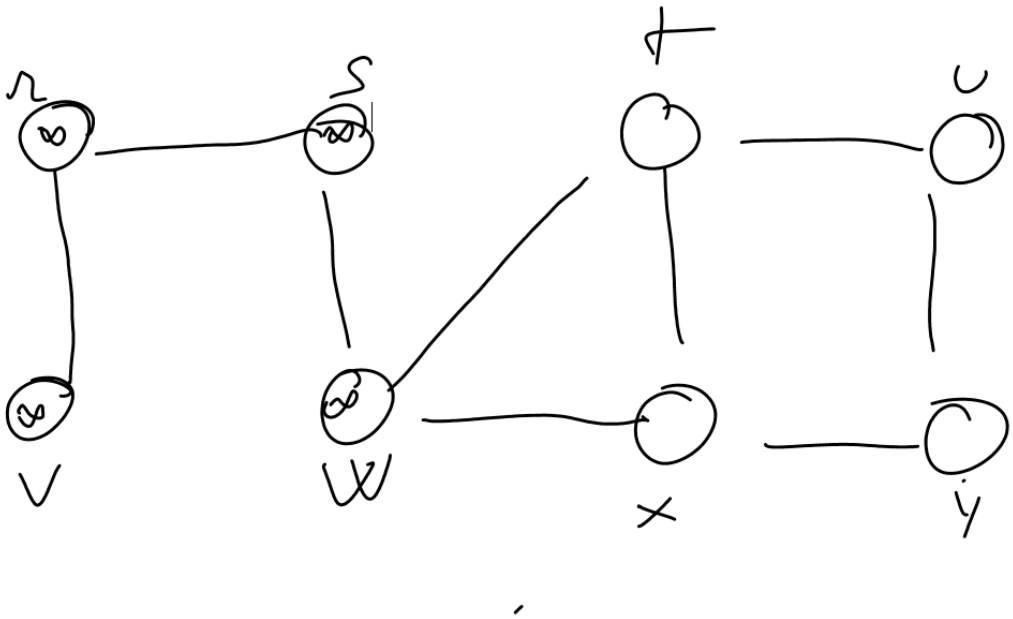
Array discovery “d” = contiene l’istante(int) che contiene quando il nodo è stato scoperto.

Array pigreco = array dei predecessori (tutti NIL all’inizio)

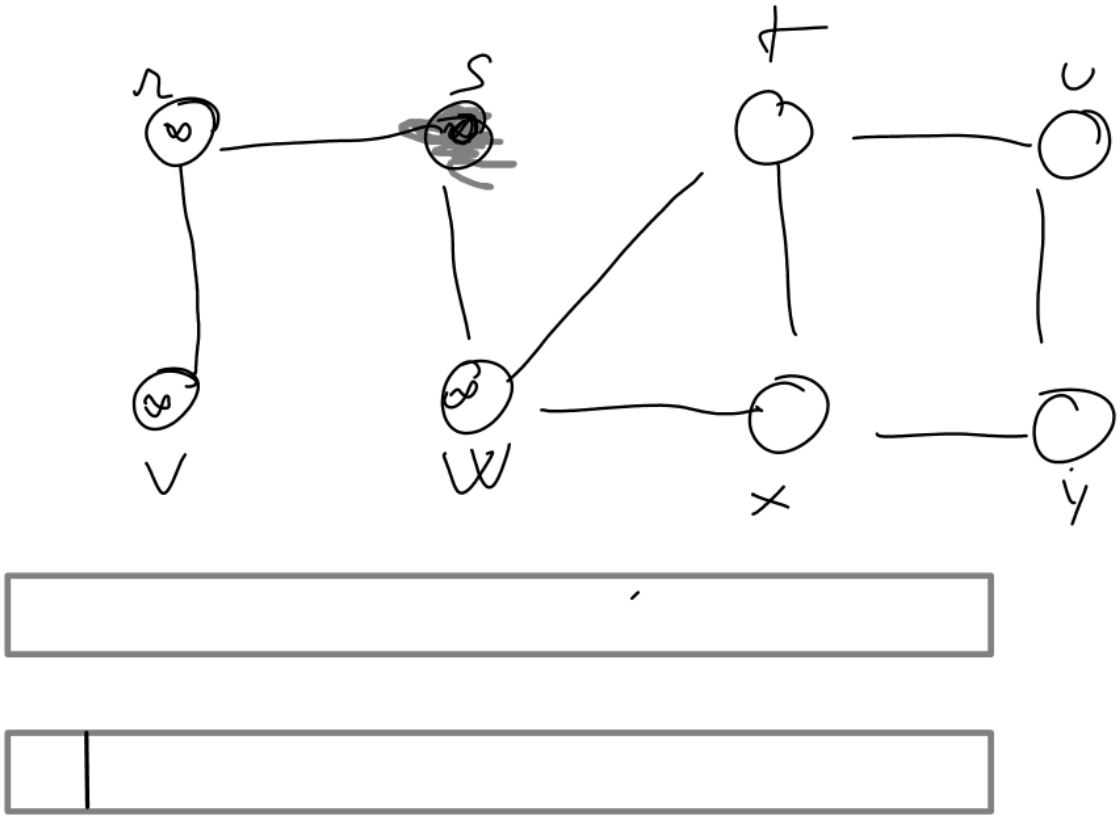
Dopo questa inizializzazione

Nodo sorgente?

Nodo sorgente = s



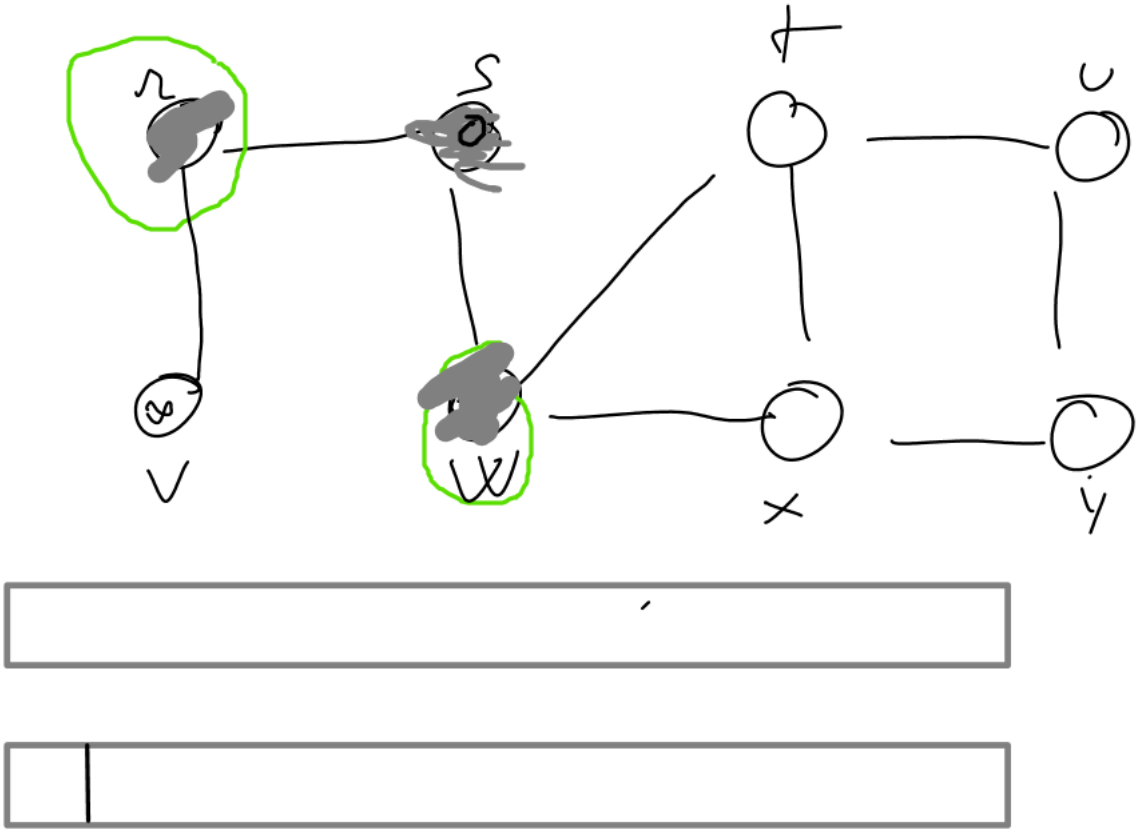
vuol dire che sto visitando S. e gli do istante 0



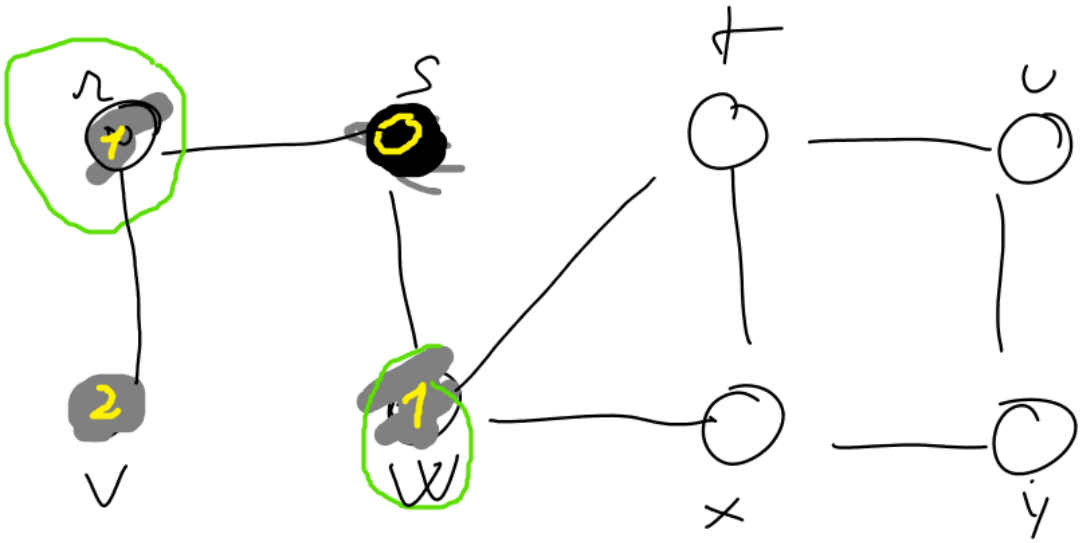
In UNA CODA (queue) Q (cioè la pigreco) avrò tutti i nodi grigi.Quindi S viene aggiunto alla queue Q.

Estraggo un nodo dalla coda e prendo tutti gli elementi ad esso adiacenti.

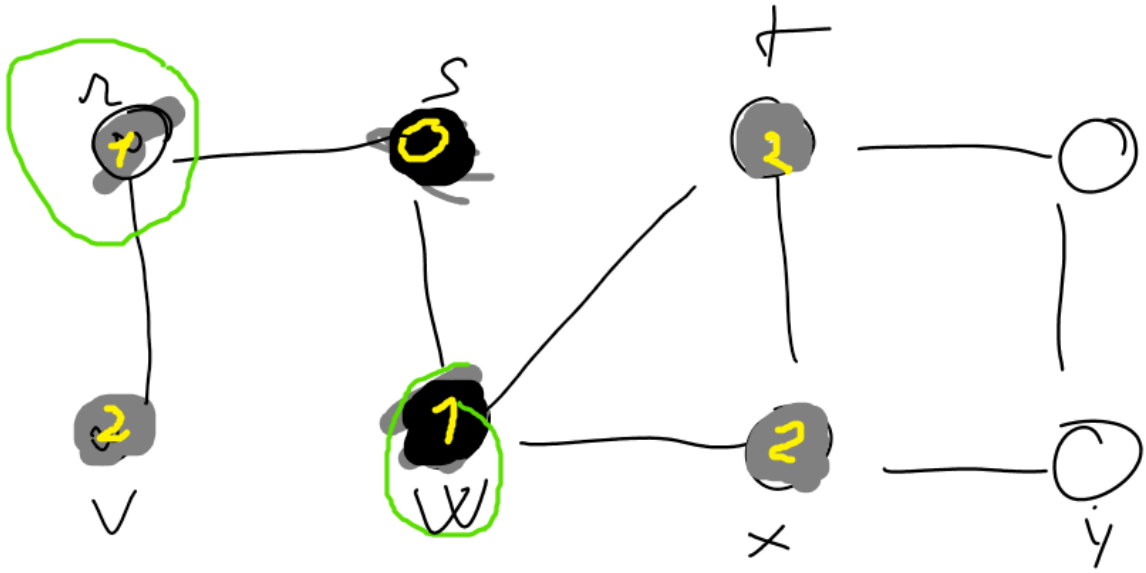
Quindi prendo S e prendo tutti i suoi adiacenti. Se il colore degli adiacenti è bianco, lo faccio diventare grigio, quindi li visito. e tolgo S dalla coda Quindi nel nodo il valore di discovery è 1.Nella coda allora poi metto r e w (non ho piu S)



Allora prendo r dalla coda, la rimuovo dalla coda, e prendo tutti i suoi adiacenti. Se sono bianchi, li metto grigi, quindi v diventa grigio. il valore di discovery di r è 2. Posso mettere v nella coda



Ho ancora elementi in coda? si. Ho w. w viene estratto dalla coda. vedo i suoi nodi adiacenti. sono t e x. t e x diventano grigi e il loro valore di discovery è 2 (w+1). il loro predecessore è w. Aggiungo t e x alla coda Q.



W diventa nero perchè non ho altri nodi adiacenti ad esso.

la v diventa nero.

vedo la coda. ho t e x. estraggo t e il suo nodo adiacente u diventa grigio, valore 3, e viene aggiunto alla coda. ci sono altri nodi adiacenti? no. allora t va a nero e adesso tocca alla x.

estraggo x. x ha un adiacente bianco y. y va a grigio e y ha valore 3. aggiungo y alla coda.

in coda ho y e u. marco nero x con valore 2.

devo visitare u e y. Non hanno nessuno vertice adiacente bianco. per cui posso marcarli come completi e inserire il loro valore di discovery

La visita BFS trova i cammini minimi fra sorgente e gli altri nodi

visita in ampiezza(scopro un nodo e mi fermo per scoprire i vicini)

visita in profondita (scopro nodo, esploro tutto e torno indietro)