GRAFO IPOTESI

# Funzioni da implementare:

* Aggiungere Arco tra due nodi.
* Rimuovere Arco tra due nodi.
* Aggiungere un Nodo.
* Rimuovere un Nodo.

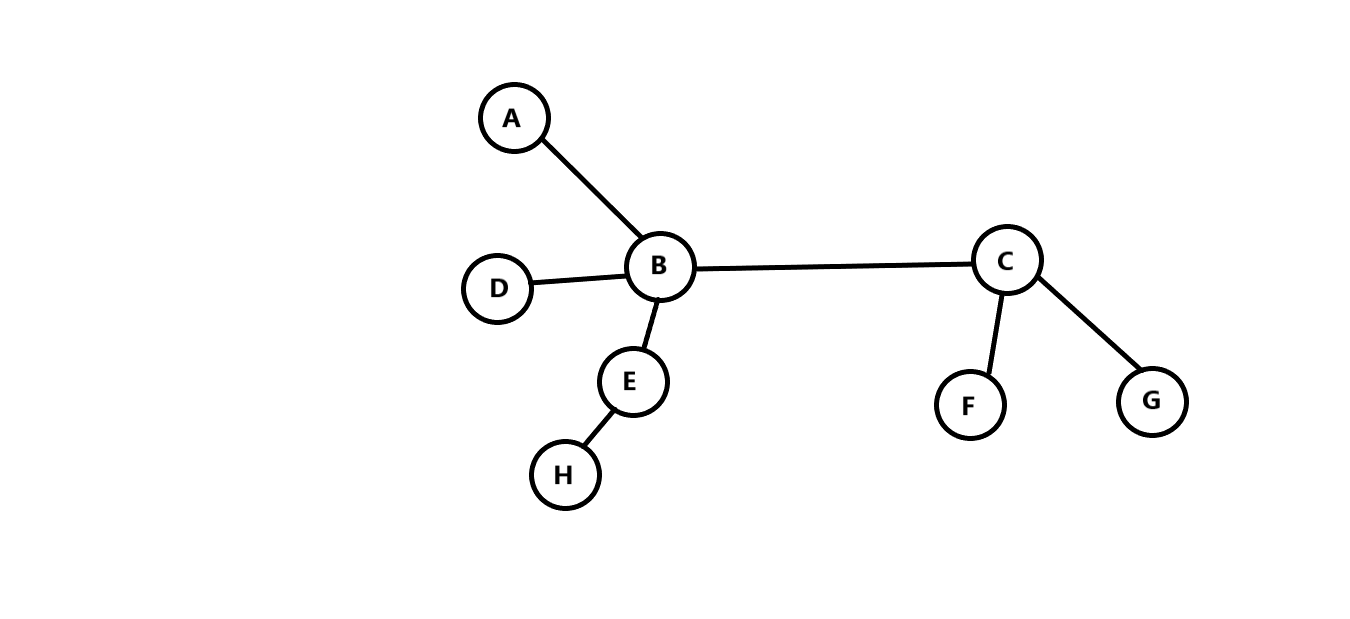
# ipotesi soluzione:

Ogni Nodo contiene un vettore di Nodi adiacenti, in questo modo dal momento che ogni nodo conosce la posizione di altri nodi adiacenti, si crea una soluzione ricorsiva che porta a conoscere il percorso per tutti i nodi.

I collegamenti o archi, si possono aggiungere scegliendo un Nodo esistente, anche non adiacente, che viene aggiunto al vettore dei nodi adiacenti, di conseguenze come prima, si conosceranno tutti i percorsi precedenti necessari.

Ogni nodo ha conoscenza in modo ricorsivo all’intero grafo e a tutti i percorsi.

## Esempio GRAFO:



## Ipotesi soluzione:

* Ogni nodo ha un vettore dei Nodi adiacenti (archi).
* Ogni nodo ha un Dato (Numero o lettera identificativo, può anche essere un qualsiasi oggetto/dato).
* Metodo che ritorna il nodo ricercato se trovato, o un valore bandiera se non esistente dopo la ricerca nei nodi adiacenti.
* Metodo inserisci arco, che aggiunge un nodo al vettore dei nodi adiacenti, fa questo a entrambi i nodi coinvolti.
* Metodo rimuovi arco, che rimuove un arco con un nodo, e viene effettuato su entrambi i nodi coinvolti, e successivamente si esegui un controllo per verificare se i nodi hanno altri nodi adiacenti o sono diventati “Zombie”.

### Ricerca

Il metodo ricerca incomincia con:

* Un ciclo for che controlla se tra i nodi adiacenti è presente il nodo da ricercare, se trovato lo ritorna.
* Un ciclo che per ogni nodo adiacente, richiama la ricerca (quindi ricorsivo), se uno dei nodi adiacenti trova il nodo da ricercare, ferma tutto e lo ritorna, se invece non lo trova e finiscono tutti i cicli for, si considera come introvabile il nodo richiesto.

### Aggiungere un arco:

Per aggiungere un arco, si seleziona il nodo a cui aggiungere l’arco e tramite un metodo “aggiungi”, si aggiunge il nodo richiesto al vettore dei nodi adiacente, e si fa lo stesso per il nodo appena aggiunto in modo che entrambi conoscano di essere adiacenti tra loro (se grafo non indirizzato, se indirizzato solamente uno dei due conoscerà la posizione dell’altro).

### Rimuovere un arco:

Per rimuovere un arco, si prende il vettore dei nodi adiacenti e (in caso di grafo non indirizzato) si rimuove al nodo interessato il nodo adiacente interessato, e si fa lo stesso sul nodo principale, nel caso sia indirizzato invece, si rimuove solamente dal nodo interessato l’arco/nodo adiacente dal vettore.

### Aggiungere un nodo:

Per aggiungere un Nodo si richiede a chi deve essere adiacente, in modo che non sia nullo a se, e si richiama quindi aggiungi al Nodo adiacente con un arco verso al nuovo Nodo appena creato.

### Rimuovere un Nodo:

Per rimuovere un nodo, si rimuove da tutti i nodi adiacenti l’arco presente nei rispettivi vettori dei nodi adiacenti, in modo che nessuno punti più al nodo rimosso, e il nodo stesso viene non ha più archi con nessuno.

Questa era la soluzione utilizzando oggetti e vettori di oggetti.

SI potrebbe anche implementare una soluzione tramite matrice delle adiacenze, che in automatico permette di creare i percorsi minimi tra i grafi e tutte le loro adiacenze, questa si ingrandisce molto ad ogni nodo aggiunto dal momento che tutti dovranno sapere se sono o meno adiacenti ad esso.

La matrice composta da un vettore a matrice a due dimensioni, contiene per ogni nodo una riga e una colonna, e all’aggiunta di ogni nodo incrementano le dimensioni totali dal momento che ogni nodo deve avere tra i suoi dati con chi è adiacente e viceversa.

La ricerca parte da un nodo, e successivamente leggendo la tabella e le sue adiacenze, crea un percorso per raggiungere la destinazione più veloce possibile.  
I percorsi minimi possono anche essere salvati in un secondo vettore e successivamente usati e modificati con l’aggiunta o rimozione di nodi.