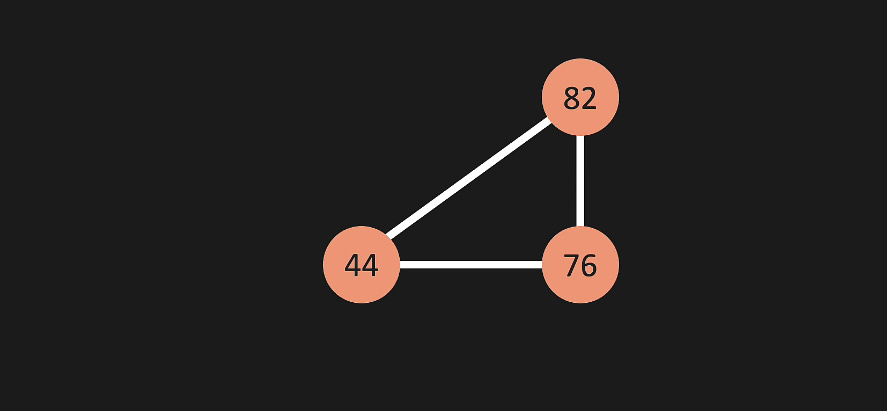
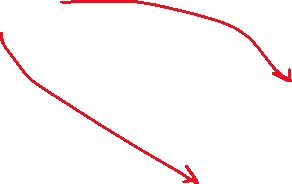
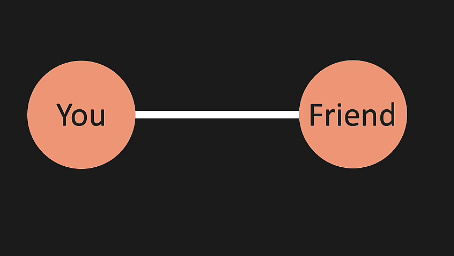
* Vertex = node
* edge = conexion (muchie)

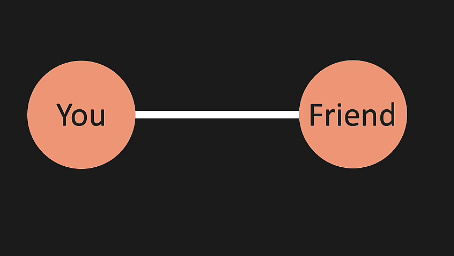




* Daca intre 2 noduri nu e pusa sageata la muchie, inseamna ca ele sunt conectate intre ele:

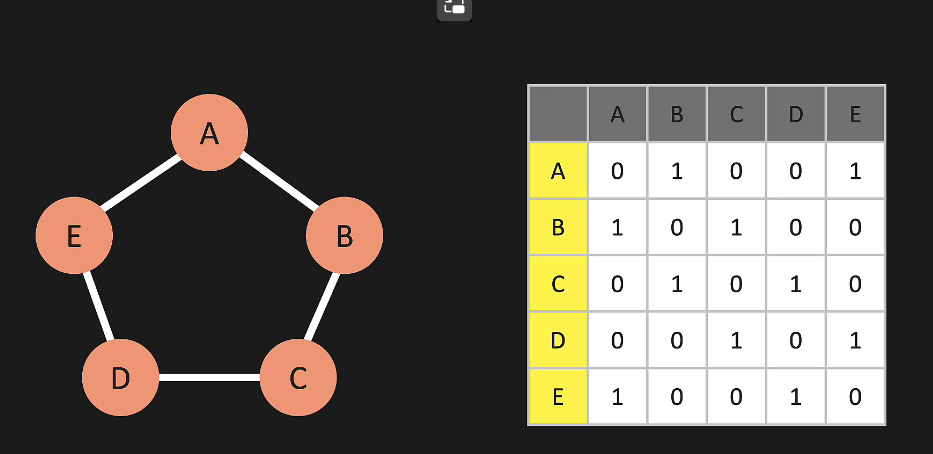


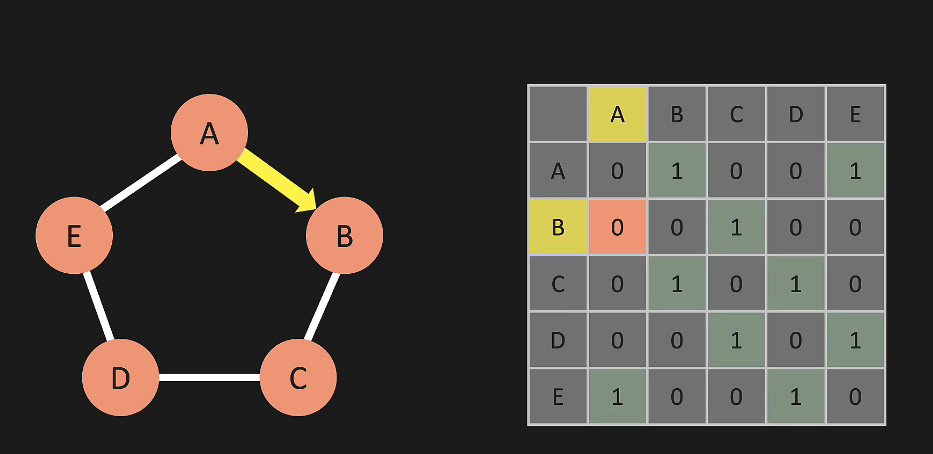
e echivalent cu

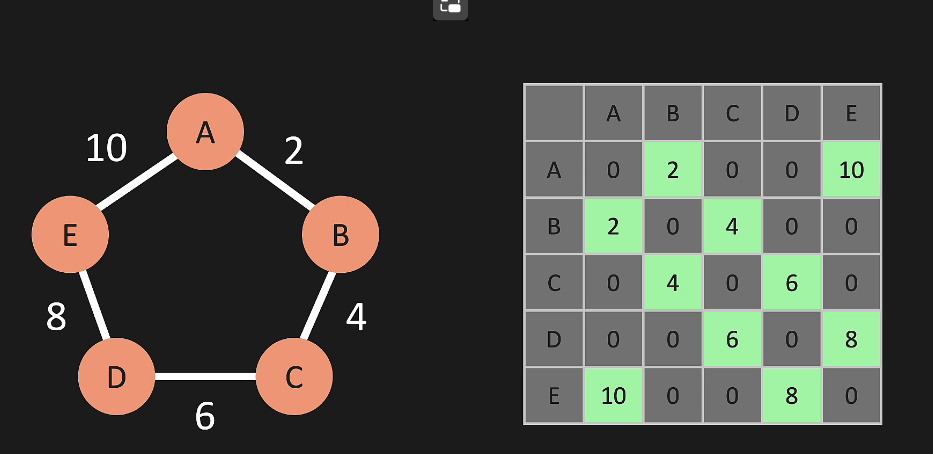




**Matricea de adiacenta**



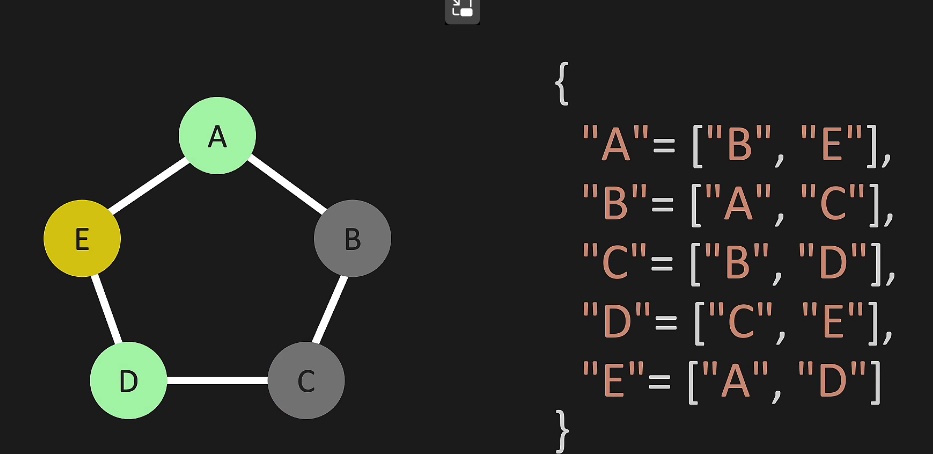




**Lista de adiacenta**

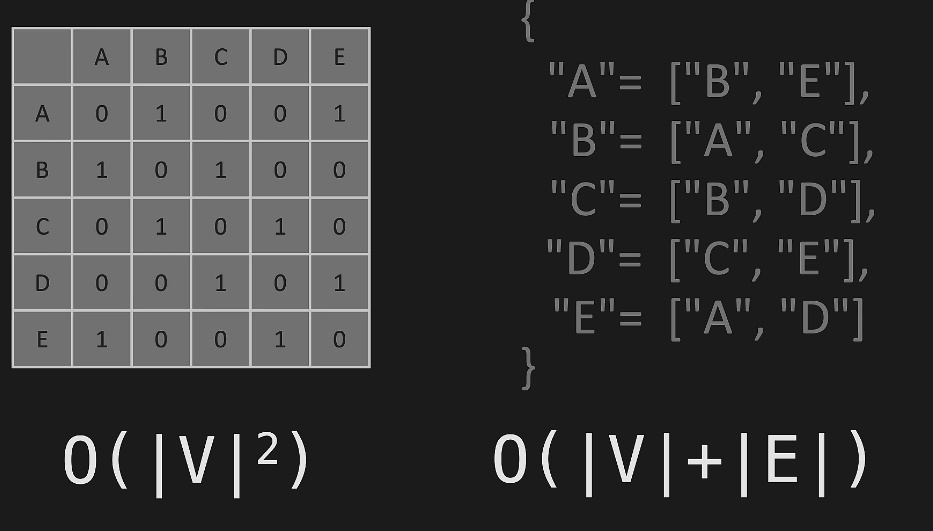
O facem cu o **HashMap**

Key va fi nodul si valoarea va fi un array cu nodurile conectate(ArrayList sau LinkedList)



**Big O**

Space complexity:



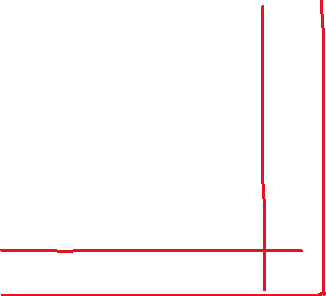
V – noduri

E – muchii

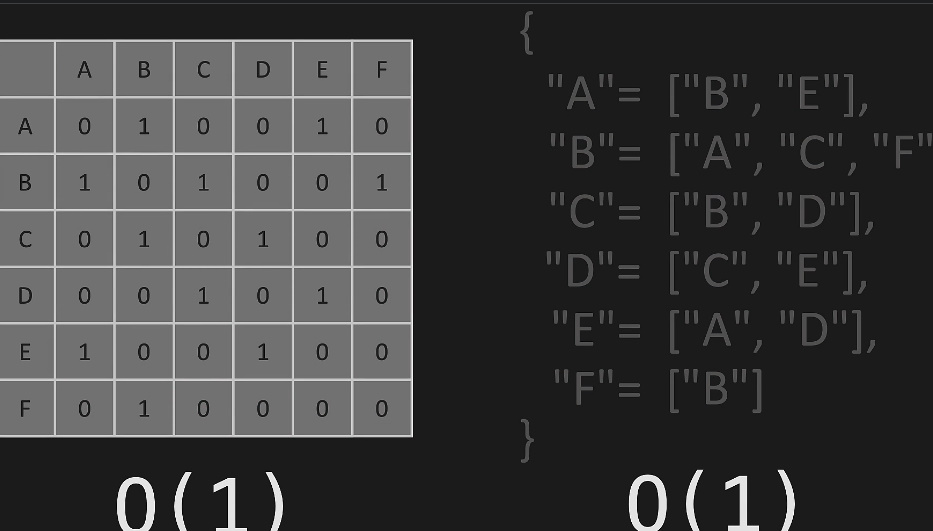
Lista de adiacenta e totusi mai buna, deoarece nu trebuie sa stocam si muchiile ce nu exista;

* Adaugarea unui nod(fara muchii)





* In matricea de adiacenta, pentru a adauga un nod trebuie o noua linie si coloana, si asta ar insemna ca ar trebui sa modificam array din fiecare nod, adica sa parcurgem fiecare nod si inca array si am avea for in for
* cu lista de adiacenta, totul e mult mai simplu
* Adaugarea muchiei:

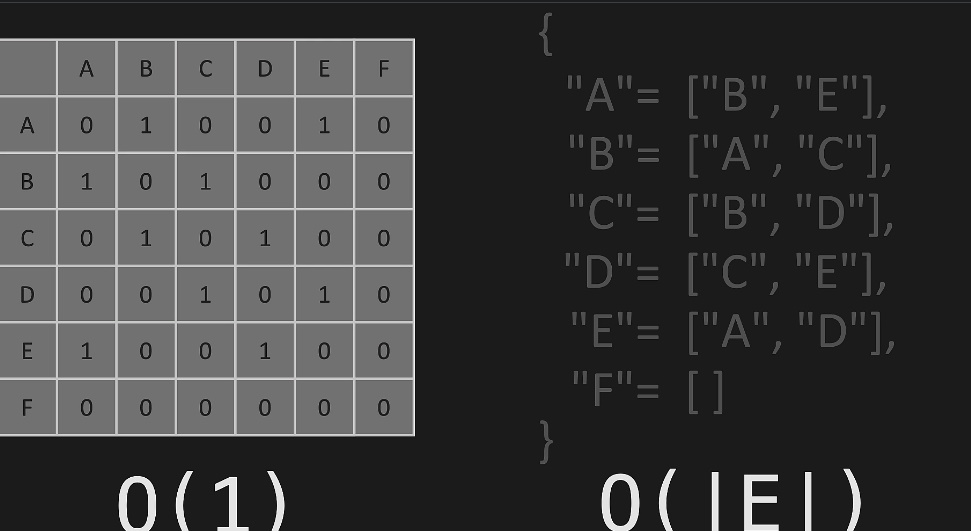




* **remove**() o muchie, de ex dintre F si B:

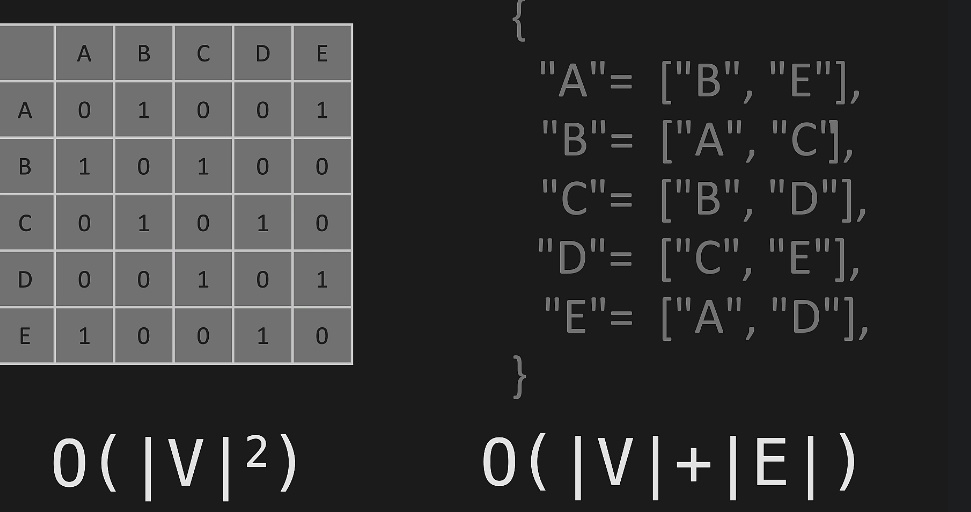
In lista de adiacenta, de ex, scoatem F din B si B din F, dar pentru asta trebuie sa parcurgem lista de noduri conectate a fiecarui din cele 2 noduri

In matricea de adiacenta e mai usor



La stergerea unei muchii, matricea de adiacenta e totusi mai buna

* **remove()** unui nod:
* Cu lista de adiacenta, putem sterge nodul F de la urma, impreuna cu array al sau, dar pe urma trebuie sa luam fiecare nod si sa iteram prin lista sa de muchii ca sa vedem daca nu cumva este F prin ele
* Cu matricea de adiacenta trebuie sa eliminam o coloana si linie si iar sa trecem deci prin fiecare array de muchii a fiecarui nod:

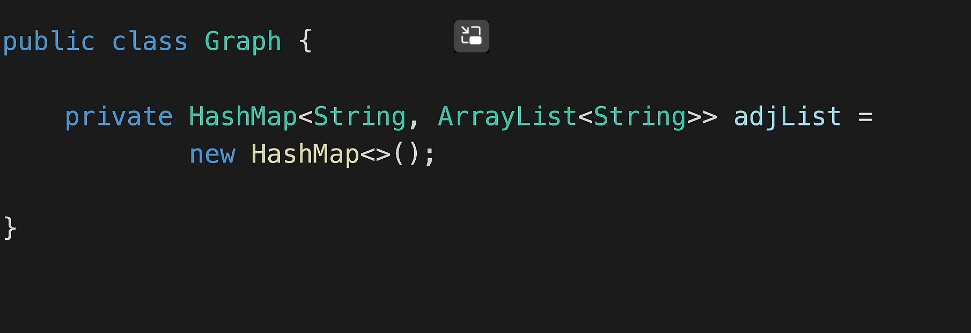


**Lista de adiacenta e mult mai buna si folosita. Nu trebuie sa stocam 0 cand avem milioane de noduri si nici milioane de randuri si noduri.**

* **De ex, facebook are miliarde de utilizatori, si acum de ex un user poate sa aiba doar un user, dar cu matricea de adiacenta, el tot va fi un tabel cu un miliard de linii si coloane plin de 0 practic, si de asta nu e deloc eficient**

**Methods**

* Clasa Graph:

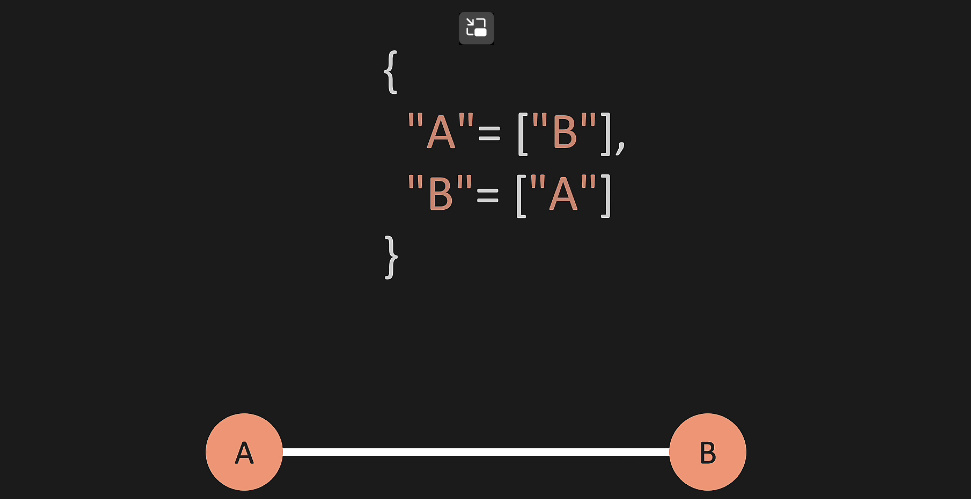


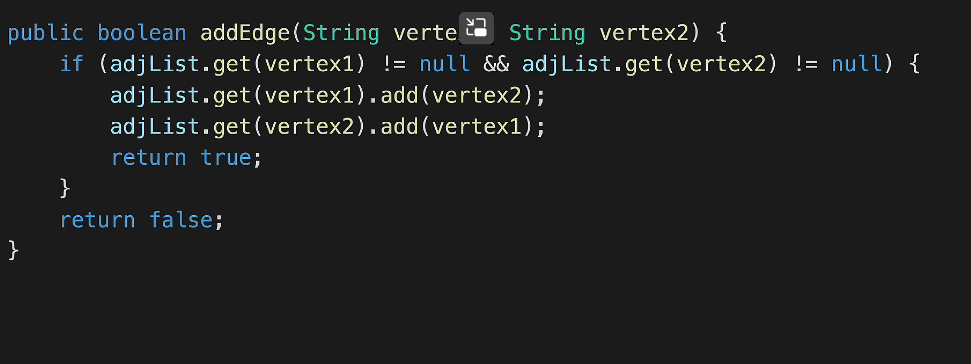
Deci, tot ce va avea clasa Graph e un HashMap, unde key va fi un nod, si ArrayList<String> va fi value, ce va stoca nodurile cu care acesta are legaturi

* **addVertex**()

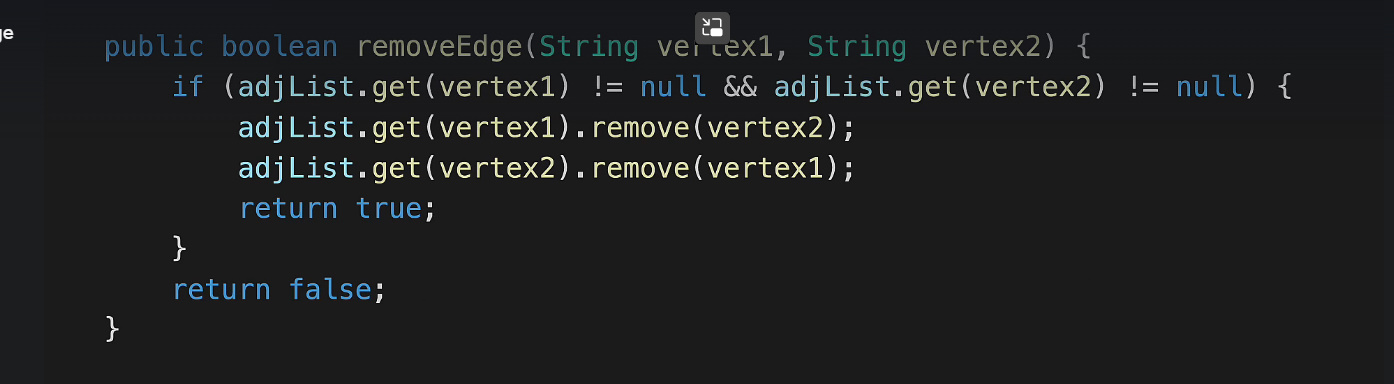


* **addEdge**(vertex1,vertex2)





* **removeEdge(vertex1,vertex2)**



* **removeVertex(vertex)**

