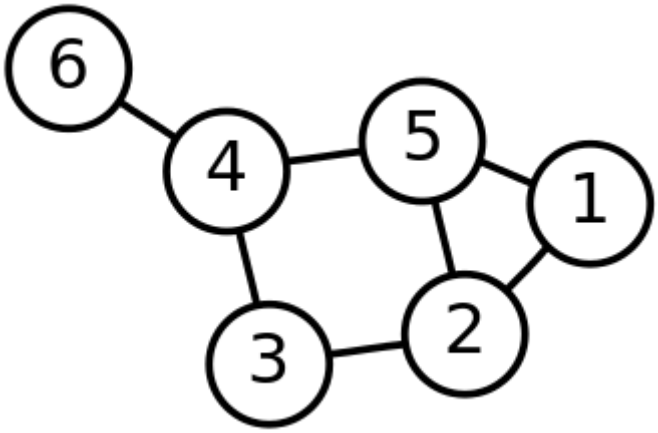
Parcurgerea în adâncime și în lățime

1 Obiective

În lucrarea de față vom prezenta cum se pot parcurge grafurile în Prolog. Grafurile le vom reprezenta prin predicate edge.

2 Considerații teoretice

Exemplu de graf neorientat:



În Prolog:

```
edge(1,5).
edge(1,2). %etc
```

2.1 Parcurgere în adâncime (DFS)

Ne vom folosi de un predicat auxiliar pentru a stoca nodurile deja vizitate.

```
:- dynamic nod_vizitat/1.

% d_search(Source, Path)
d_search(X,_):-df_search(X,_). % parcurgerea nodurilor
d_search(_,L):-!, collect_reverse([],L). % colectarea rezultatelor

df_search(X,L):-
      asserta(nod_vizitat(X)),
      edge(X,Y),
      not(nod_vizitat(Y)),
      df_search(Y,L).

% colectarea se face în ordine inversa
collect_reverse(L,P):-
      retract(nod_vizitat(X)), !,
      collect_reverse([X|L],P).
collect_reverse(L,L).
```

Urmărește execuția la:

```
?- d_search(1,R).
```

2.2 Parcurgerea în lățime (BFS)

În cazul parcurgerii în lățime avem nevoie de un predicat care să ne spună care este următorul nod de expandat. În predicatul „nod_vizitat” vom stoca elementele în ordine parcurgerii lor.

```
% b_search(Source, Path)
b_search(X,_):- % parcurgerea nodurilor
      assertz(nod_vizitat(X)),
      assertz(nod_de_expandat(X)),
      bf_search.
b_search(_,R):-!, collect_reverse([],R). % colectarea rezultatelor

bf_search:-
      retract(nod_de_expandat(X)),
      expand(X),!,
      bf_search.

expand(X):-
      edge(X,Y),
      not(nod_vizitat(Y)),
      asserta(nod_vizitat(Y)),
      assertz(nod_de_expandat(Y)),
      fail.
expand(_).
```

Urmărește execuția la:

```
?- b_search(1,R).
```

3 Exerciții

1. Modificați predicatul DFS astfel încât să caute noduri numai până la o anumită adâncime (DLS – Depth-Limited Search).