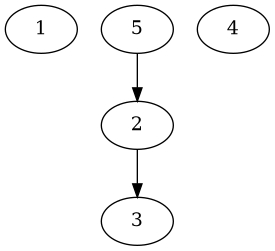
**Zadání zápočtového testu KI/APR2 30.5. 2025**

**1. příklad (15 bodů)**

Do třídy *Graph* přidejte metodu, která pro daný orientovaný graf vrací seznam všech izolovaných uzlů v grafu. Uzel je izolovaný, pokud z něho či do něj nevede žádná hrana.

*vyzkoušejte* na následujícím grafu (má dva izolované uzly)

g = Graph(GraphType.DIRECTED)

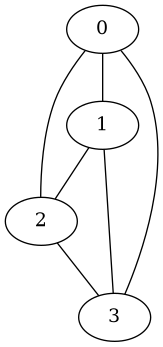
g.add\_node(1)

g.add\_edge(2, 3)

g.add\_node(4)

g.add\_edge(5, 2)

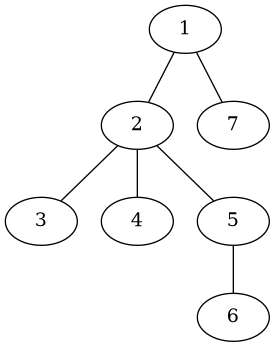
**2. příklad (15 bodů)**

Implementujte funkci *uplny\_graf*, která vytvoří úplný graf o *n* uzlech. Úplný graf je neorientovaný graf, jehož každé dva uzly jsou spojeny hranou. Navíc pro otestování vytvořte funkci *testuj\_uplnost*, která ověří, že graf je úplný a to podle věty, že stupeň každého uzlu v úplném grafu je roven *n*-1.

g = uplny\_graf(4) # vrací graf vpravo

testuj\_uplnost(g) # vrací True

**3. příklad (30 bodů)**

Vytvořte funkci, která otestuje zda je daný neorientovaný graf stromem. Graf je stromem, pokud **neobsahuje cyklus** (tj. posloupnost hran, po níž se dostaneme znovu do stejného uzlu) a je **souvislý** (má jen jednu komponentu).

vyzkoušejte na následujícím grafu (který je stromem):

tree = Graph(GraphType.UNDIRECTED)

tree.add\_edge(1, 2)

tree.add\_edge(1, 7)

tree.add\_edge(2, 3)

tree.add\_edge(2, 4)

tree.add\_edge(2, 5)

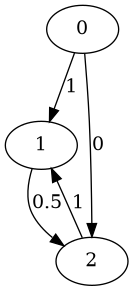
tree.add\_edge(5, 6)

Graf, který není stromem z něho uděláte přidáním hrany (4, 6) [vznikne cyklus] nebo odebráním hrany (2, 5) [graf se rozpadne na dvě komponenty].

**4. příklad (40 bodů)**

Vytvořte vlastní třídu *WeightedGraph* (s pomocí třídy *DictyonPhi.Graph*) representující jednoduchý **ohodnocený orientovaný graf**, jehož každá hrana má přiřazenu váhu (racionální číslo ≥ 0). Třída by měla mít následující metody:

1. *konstruktor*, který očekává matici vah (tj. čtvercovou matici, jejíž prvek na *i*-tém řádku a *j*-tém sloupci určuje váhu hrany vedoucího z uzlu *i* do uzlu *j*, pokud taková hrana neexistuje, pak má daný prvek hodnotu -1). Matice je representována seznamem řádků (což jsou opět seznamy).
2. metodu *get\_graph*, která vrací objekt třídy *DictyonPhi.Graph*
3. metodu *component\_count*, která vrací počet komponent v tomto grafu
4. metodu nebo vlastnost *get\_sorted\_weights,* která vrací iterátor přes všechna ohodnocení v grafu od nejmenší k největší.

**vyzkoušejte na grafu:**

wg = WeightedGraph([[-1, 1, 0],

[-1, -1, 0.5],

[-1, 1, -1]])