Tehnici de căutare neinformată

Se folosesc pentru probleme care pot fi abstractizate la un graf (orientat sau neorientat). Presupun existența unui nod de început (nodul start) și unul sau mai multe noduri scop (la care vrem să ajungem din nodul start).

BreadthFirst

Pași:

1. Se pune nodul start într-o coadă.
2. Repetitiv:
    a. dacă nu este coada vidă, se extrage primul nod din coadă și se verifică dacă este scop, caz în care se returnează o soluție. Dacă am ajuns la numărul de soluții dorit, ne oprim.
    b. expandăm nodul și adăugăm succesorii săi în coadă, cu condiția ca succesorii să nu fi fost deja vizitați pe ramurile din arbore corespunzătoare lor (atenție e posibil să fi fost vizitați pe o altă ramură din arbore, dar acest lucru nu contează, vizitarea nu se consideră la nivel global ci doar pe drumul curent al acelui succesor).

DepthFirst

1. Se pune nodul start într-o stivă.
2. Repetitiv:
    a. Dacă vârful stivei este nod scop, afișăm o soluție. Dacă am ajuns la numărul de soluții dorit, ne oprim.
    b. Pentru vârful stivei dacă nu i s-au generat succesorii deja, adăugăm succesorii săi nevizitați **pe drumul curent** din stivă și verificăm dacă e nod scop, caz în care se returnează o soluție. Dacă am ajuns la numărul de soluții dorit, ne oprim.
    c. Dacă vârful stivei nu are succesori sau au fost deja generați și procesați, este eliminat din stivă.

Exerciții laborator 1

Atenție! Acestea sunt niște exerciții generale care servesc drept ghid și pot fi schimbate sau înlocuite de profesorul de laborator. Este necesar să verificați ce variantă de exerciții ați primit în oră la semigrupa la care ați participat.

1. Definiți o clasa NodArbore, care va reprezenta un nod dintr-un **arbore de cautare (deci nu nod al grafului inițial)**, cu câmpurile: informatie, parinte (care e un obiect de tip NodArbore). In constructor, vom defini pentru parinte ca valoare implicită None. Clasa NodArbore va avea următoarele metode:
    a. drumRadacina(self) care va returna o listă cu toate nodurile ca obiecte (nu informatia nodurilor) de la rădăcină până la nodul curent
    b. inDrum(self,infonod) care verifică dacă informatia nodului dat ca parametru a fost vizitata în drumul nodului curent (deci nu în tot arborele) caz în care returnează True, altfel (dacă nu a fost vizitat) False. Reformulat: dacă informația se mai găsește în istoricul (drumul) nodului curent, returnăm True.
    c. funcția __str__ care va returna un string doar cu informația nodului
    d. funcția __repr__ care va returna un string continand informatia nodului, urmata de un spatiu, urmat de o paranteză în care se află tot drumul de la rădăcină până la acel nod). De exemplu "c (a->b->c)" unde *c* e informația nodului curent și *a,b,c* sunt informațiile nodurilor din drumul de la rădăcină(*a*) către el.

2. Definiți o clasă Graf, în care se va memora un graf (alegeți voi dacă prin listă de vecini, matrice de adiacența, sau listă de noduri și muchii), inclusiv informația nodului start și nodurile scop (date ca lista de informații scop). Veți defini pentru ea:
    a. un constructor prin care se transmit informațiile grafului.
    b. o metoda scop(self, informatieNod) care primește o informație de nod și verifică dacă e nod scop
    c. o metoda (care va fi folosită în algoritmi pentru generarea arborelui de căutare), numită **succesori(self, *nod*)** care primește un nod al arborelui de cautare și parcurge nodurile adiacente din graf, returnand o lista de obiecte de clasa Nod ce reprezinta succesori direcți ai nodului (care vor fi fii în arborele de căutare), care nu au fost vizitati pe ramura curentă. Toți succesorii vor avea, evident, parintele egal cu *nod*

3. Implementați tehnica de căutare Breadthfirst, folosind clasele Nod și Graf definite mai sus. Se va citi de la tastatură (sau se da intr-o variabila) un număr NSOL de soluții. Se vor afișa primele NSOL soluții.
4. Implementați tehnica de căutare DepthFirst, folosind clasele Nod și Graf definite mai sus, în mod recursiv. Se va citi de la tastatură un număr NSOL de soluții. se vor afișa primele NSOL soluții.
5. Implementați tehnica de căutare DepthFirst, folosind , folosind clasele Nod și Graf definite mai sus, în mod nerecursiv. Se va citi de la tastatură un număr NSOL de soluții. se vor afișa primele NSOL soluții.

6. Modificați implementarea algoritmului BreadthFirst din laborator astfel încât să fie afișată soulția imediat ce a fost adăugată în coada (și nu când ajunge prima în coadă și este eliminată).

Anexă

```
m = [
    [0, 1, 0, 1, 1, 0, 0, 0, 0, 0],
    [1, 0, 1, 0, 0, 1, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 1, 0, 1, 0, 0],
    [1, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 1, 0, 1, 0, 0, 0, 1, 1],
    [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
]

start = 0
scopuri = [5, 9]
```