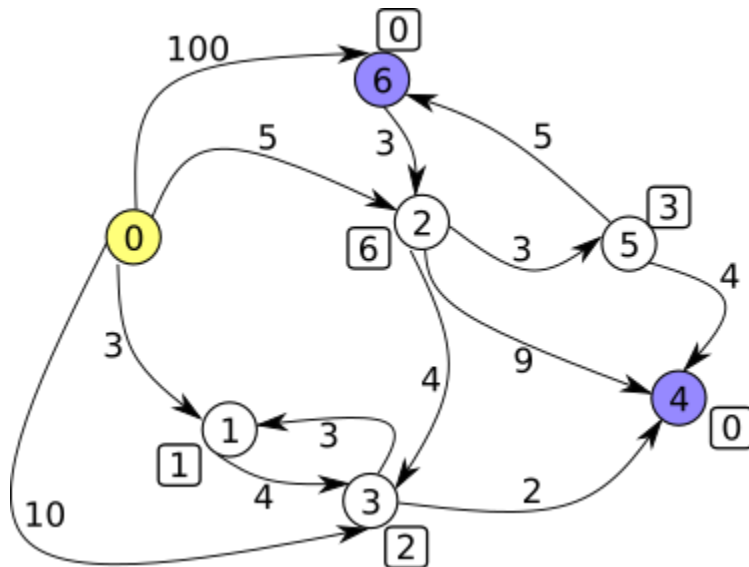


## Laborator 3. Taskuri.

### 1) Pregătirea contextului pentru algoritmul A\*.

- Preluați clasele Nod și Graf implementate în laboratoarele anterioare.
- Modificați clasa Nod astfel încât constructorul să primească parametrii  $g$  și  $h$  și să creeze proprietățile clasei cu aceeași valoare și același nume. Valorile implicite pentru  $g$  și  $h$  în antetul constructorului vor fi ambele 0. Constructorul va calcula și proprietatea  $f$  a instanței nod ca sumă a parametrilor  $g$  și  $h$ .
- Modificați clasa Graf astfel încât să primească o reprezentare a grafului cu muchii ponderate. Vom presupune că ponderile sunt toate nenule, iar 0 înseamnă că nu există muchie/arc. Puteți folosi una dintre variantele următoare:
  - Matrice de ponderi pe muchii/arce.
  - Listă de vecini cu costuri. Pentru fiecare nod  $n$  se va memora o listă de tuple în care prima valoare e informația vecinului și a doua valoare e costul de la nodul  $n$  la acel vecin.
  - Listă de noduri și muchii cu costuri. Dacă nodurile sunt numerotate de la 0 la  $N-1$  (unde  $N$  e numărul de noduri) atunci e suficientă lista de muchii. Fiecare muchie/arc va fi reprezentată printr-un tuplu:  $(n_1, n_2, \text{cost})$  unde  $n_1$  și  $n_2$  sunt cele două noduri pentru care am reprezentat legătura iar al treilea parametru e costul asociat legăturii.
- Modificați clasa Graf astfel încât să primească o listă de estimări. Fiecare valoare din listă va corespunde unui nod din graf și va estima (în mod admisibil) costul de la nodul corespunzător la scop (admisibil = valoarea să fie mai mică sau egală cu costul drumului cel mai ieftin de la nodul respectiv la oricare dintre nodurile scop).
- Scrieți datele grafului de mai jos folosind modul de reprezentare ales. Estimarea pentru fiecare nod se găsește în pătrățelul vecin cu nodul. Nodul start e galben și scopurile sunt mov.



- Modificați 3 valori în listă astfel încât estimarea să își păstreze proprietatea de admisibilitate. Scrieți într-un comentariu o altă listă în care una dintre estimări să nu fie admisibilă.
- Adăugați metoda `estimeaza_h(nod)` (sau `estimeaza_h(infoNod)`) care primește un nod (sau o informație de nod) și returnează estimarea corespunzătoare informației lui.
- Modificați funcția `succesori` din clasa `Graf` (implementată în primul laborator) astfel încât să calculeze pentru noii succesori și proprietățile `g`, `h`, `f`.

2) Implementați un algoritm bazat pe ideea algoritmului  $A^*$  care returnează soluții multiple, ordonate crescător după cost. Acest algoritm va returna primele `NSOL` soluții în ordinea costului, unde `NSOL` e citit de la tastatură. Acest algoritm diferă un pic față de algoritmul predat la curs, prin faptul că nu mai folosește lista `closed` iar în lista `open` pot exista mai multe noduri cu aceeași informație dar cu istoric diferit. **Taskuri de implementare:**

- Creați o funcție `aStarSolMultiple(graf, nsol)`, unde `graf` e o instanță a clasei `graf` și `nsol` e numărul de soluții de afișat. Pentru a ordona drumurile expandate după cost vom folosi o coadă de prioritate. **O vom simula printr-o listă** (pentru cine vrea să lucreze cu `PriorityQueue`, vezi exercițiul 3). Adăugăm nodul start în listă.

Repetitiv, cât timp coada nu e vidă facem următorii pași:

- Extragem primul nod din coadă. Vom numi acest nod nodul curent.
- Verificăm dacă nodul curent e scop, caz în care afișăm drumul soluție până la el și costul (total al) acestui drum. Actualizăm numărul de soluții rămase de afișat. Dacă nu mai avem soluții de afișat, terminăm.

- Calculăm succesorii nodului curent și îi adăugăm în coadă, astfel încât coada să rămână ordonată după cost (măsura  $f$ ). Pentru valori  $f$  egale, elemente din listă se ordonează descrescător după  $g$ . Având în vedere că elementele sunt în ordine crescătoare a costului drumului, aplicați un algoritm eficient (de exemplu, căutare binară) pentru introducerea succesorilor în coadă.
- Apelați metoda `aStarSolMultiple()` pe graful definit în exercițiul de mai sus, folosind clasele `Graf` și `Nod`.

3) Implementați algoritmul de mai sus, bazat pe  $A^*$ , dar cu soluții multiple, folosind o structură de tip `Priority Queue`. Atenție, e nevoie să implementați operatorii `__eq__` și `__le__` pentru clasa `Nod`. Măsurați cu `cProfile` sau cu funcția `time()` din modulul `time` timpul de execuție pentru implementarea cu coada de priorități ca listă și coada de priorități implementată cu [PriorityQueue](#). Scrieți într-un comentariu timpii și precizați care implementare e mai rapidă.

**4) Implementați algoritmul  $A^*$  pentru a obține drumul de cost minim pentru graful problemei din curs sau din laborator (folosit la exercitiul 2), folosind listele `open` și `closed` și urmând pseudocodul algoritmului. Algoritmul va fi implementat în funcția `a_star(graf)`**

5) Folosind implementarea problemei canibalilor și misionarilor de la laboratorul anterior, rezolvați problema folosind  $A^*$  urmând cerințele:

- Folosiți clasele `Nod` și `Graf` modificate astfel încât să trateze și costul
- Considerați costul pe orice mutare ca fiind 1
- Implementați funcția `estimeaza_h()` corespunzătoare problemei, funcția trebuind să returneze o estimatie admisibilă pentru fiecare nod.

6) Implementați algoritmul  $A^*$  folosind structuri eficiente:

- Folosiți `heapq` pentru coada `open`. Evităm să folosim clasa `PriorityQueue`, deoarece avem nevoie să ștergem elemente și după alte criterii decât prioritatea. Cu [heapq](#) avem acces la lista pe care se bazează heap-ul.
- Folosiți un dicționar în locul listei `closed`, în care cheile sunt informațiile nodurilor și valorile sunt chiar nodurile.
- Măsurați cu [cProfile](#) sau cu funcția `time()` din modulul `time` timpul de execuție pentru implementarea cu `open` și `closed` ca liste, respectiv `open` și `closed` ca structuri eficiente. Scrieți într-un comentariu timpii și precizați care implementare e mai rapidă.

7) Modificați algoritmul  $A^*$  cu soluții multiple (de la al 2-lea sau al 3-lea exercițiu) astfel încât în cazul unor drumuri de cost egal să returneze aceste drumuri în ordinea numărului lor de muchii.