Documentatie

***Apelare program:***

Exemplu 1:

python main.py -in D:\facultate\IA\proiectIA\Input -out D:\facultate\IA\proiectIA\Output -nrsol 2 -time 10

Exemplu 2:

python main.py -in D:\facultate\IA\proiectIA\Input -out D:\facultate\IA\proiectIA\Output -nrsol 1 -time 5

***Validari:***

* **Verificarea corectitudinii datelor de intrare**: in metoda *citire* din clasa *Problema*se valideaza inputul:
* Trebuie sa existe stare initiala si finala
* Nu trebuie sa apara culori fara cost atribuit in starea initiala sau finala
* Trebuie sa am 3 valori pentru fiecare vas din starea initiala si 2 valori pentru fiecare vas din starea finala
* Trebuie sa am pe primele randuri cate un cost pentru fiecare culoare
* Trebuie sa am un numar mai mare sau egal de vase in configuratia initiala decat in cea finala
* **Găsirea unui mod de a realiza din starea iniţială că problema nu are soluții**: in metoda *stare\_cu\_potential* din clasa *NodParcurgere*se verifica daca starea curenta mai merita sa fie extinsa sau nu. Pentru o stare care nu merita sa fie extinsa putem garanta ca nu are solutii, insa pentru cele cu potential nu putem garanta acest lucru. Se apeleaza aceasta metoda pentru starea intiala.

Pentru a spune ca o stare nu are potential calculam cate din culorile finale s-au obtinut si verificam daca se mai pot face combinatii (vasele nu sunt toate pline si exista combinatii de culori posibile dintre cele curente). Daca ne-am blocat in configuratia curenta (nu mai putem face combinatii sau vasele sunt toate pline) si nu am ajuns la toate culorile finale, atunci starea poate fi considerata fara solutii.

***Optimizari:***

* **Gasirea unui mod de reprezentare a starii, cat mai eficient**: reprezentare sub forma de obiect de tip NodParcurgere cu datele membre:
* *nod* => referinta catre un obiect de tip Nod care contine *info* (cum arata vasele la momentul curent -lista de tupluri-) si *h* (valoarea functiei h’ -costul estimat de la nodul curent la un nod scop- )
* *parinte =>* referinta catre un obiect de tip NodParcurgere pentru a tine evidenta drumului
* *g* =>valoarea functiei g (costul de la nodul start la nodul curent)
* *f*  => valoarea functiei f’ (costul estimat al unui drum) = g + h’
* **Gasirea unor conditii din care sa reiasa ca o stare nu are cum sa contina in subarborele de succesori o stare finala deci nu mai merita expandata (nu are cum sa se ajunga prin starea respectiva la o stare scop):** metoda *stare\_cu\_potential* din clasa *NodParcurgere* prezentata si mai sus.
* **Implementarea eficienta a algoritmilor cu care se ruleaza programul, folosind eventual module care ofera structuri de date performante**: au fost folosite cozi (din queue.Queue) si stive (din queue.LifoQueue)

***Euristici:***

1. **Banala**: presupune atribuirea unei valori boolene pentru fiecare stare.

Daca starea este finala, valoarea atribuita va fi 0 pentru a favoriza selectarea ei in alegerea pasului urmator. In caz contrar, valoarea atribuita va fi 1.

Aceasta euristica este admisibila deoarece la fiecare pas are loc o turnare, deci costul este minim 1, iar ambele valori din euristica, 0 si 1, sunt mai mici decat minimul unei turnari. In acelasi timp, aceasta euristica grabeste ajungerea la o stare finala, astfel incat alege intotdeauna starea finala in defavoarea celorlalte (dat fiind faptul ca are loc o sortare descrescatoare dupa f).

1. **Admisibila 1**: presupune atribuirea unei valori intregi pentru fiecare stare.

Pentru fiecare vas din starea curenta care nu se regaseste in starea finala, dar are o culoare din starea finala (altfel spus, culoarea este buna, dar cantitatea nu) se aduna la h’ costul unui litru din acea culoare. Se considera culorile din starea finala sub forma de multime, pentru a nu exista duplicate, iar din aceasta multime se elimina culorile atunci cand se adauga costul la h’ (pentru a nu avea o situatie de genul in care s-ar adauga de 2 ori costul culorii si ar genera o euristica inadmisibila: Stare curenta (4 3 mov), (6 5 mov); Stare finala (6 mov)). In acest caz, am presupus ca pentru a ajunge la cantitatea corecta din culoarea respectiva se toarna un litru de apa. In realitate, se va turna cel putin un litru de apa, fapt ce face ca h’ <= h.

1. **Admisibila 2**:
2. **Neadmisibila**: presupune atribuirea unei valori intregi pentru fiecare stare.

Pentru fiecare vas care nu se regaseste in configuratia finala se adauga la h’ costul culorii din acel vas (daca este o culoare valida). Aceasta euristica este inadmisibila deoarece nu ne intereseaza in starea finala ce culori si in ce cantitati sunt in celelalte vase. Cel mai simplu mod de a ilustra e de a lua o configuratie finala, precum (3 mov), si o stare curenta, de exemplu (5 3 mov), (2 2 rosu), (3 1 albastru). Valoarea h pentru starea curenta este 0 deoarece este stare finala, pe cand valoarea h’ este costul culorii albastru + costul culorii rosu. Deci h’ >= h. Se poate observa de asemenea ca in cazul *inputului scurt* valoarea minima a drumului este 4, iar aceasta euristica furnizeaza valoarea 5, facand 2 turnari in loc de una.