Blocuri cu număr limitat de transporturi

Se va implementa o problemă asemănătoare cu problema blocurilor, discutată în laborator. Puteți porni de la codul dat in laborator: https://repl.it/@IrinaCiocan/problemablocurilor#main.py

si pe care il puteti modifica, conform cerințelor de mai jos.

Optional puteti implementa problema de la zero, dacă doriți.

Imagini ajutătoare aveți la:

https://drive.google.com/drive/folders/1qRv5kq9Qv6KTj5es3vuNOXhfsRAGKlVF?usp=sharing

Avem un set de stive de blocuri. Considerăm că pentru fiecare bloc avem asociată o culoare (poate fi codificată cu o literă, de exemplu).

Mutarea blocurilor și trecerea de la o stare la alta se face în felul următor:

- Un bloc poate fi mutat de maxim K ori (deci se va păstra pentru fiecare bloc un contor cu numărul de mutări efectuate, sau rămase)
- atunci când plasăm un bloc de culoare c pe o stivă și în stânga sau în dreapta lui e un bloc cu aceeași culoare c, dacă există un alt bloc în stânga sau în dreapta lui de o culoare diferită, aceea se va transforma în c. (de exemplu dacă în stânga exista un bloc cu culoarea c și in dreapta un bloc cu culoarea b, blocul cu culoarea b își schimba culoarea în c. Idem pentru bloc de aceeași culoare în dreapta).

Costul mutarii unui bloc este dat de numarul de mutări efectuate până atunci de către bloc, înmulțit cu numărul de blocuri de aceeași culoare cu el, aflate pe stiva de pe care este luat.

O stare e considerata **scop** dacă în toata configurația (pe toate stivele) avem blocuri de aceeași culoare (nu există mai mult de o culoare în întreaga configurație).

Deci pot fi mai multe stari finale posibile pentru aceeasi configuratie initiala si **testarea atingerii scopului trebuie facuta prin verificarea conditiei**, **NU** prin enumerarea posibilitatilor de nod scop.

Fisierul de intrare va avea un format stabilit de voi, cu conditia sa cuprinda toate datele cerute de problema: K, starea initiala. In starea initiala pot fi si stive vide, puteti sa le codificati oricum (de exemplu cu un simbol care nu apare in sirurile din blocuri sau pur si simplu cu un spatiu/linie noua.

Cerințe:

1. Modificați modul de citire din fișier (formatul fișierului de intrare va fi decis de voi) pentru a citi datele proprii problemei. Nu mai este necesară citirea stării finale, deoarece în cazul problemei date orice configurație care îndeplinește condițiile din cerință poate fi stare finală (deci nu mai e o stare fixă). Explicați într-un comentariu cum ați tratat în fișier stivele vide; comentariul va fi scris lângă funcția care citește și parsează fișierul **(0.3 puncte)**
2. Creați un fișier de input cu o configurație corespunzătoare problemei (nu contează dacă are soluție sau nu). **(0.1 puncte)**
3. Modificați modul de generare a succesorilor ca să corespundă cu cerințele de mai sus. **(0.5 puncte)**
4. Calculați costul unei mutări **într-o variabila numită cost**, în funcția de generare a succesorilor. Folosiți apoi această variabilă în crearea unui succesor (în funcția de mai sus) astfel încât să aibă în mod corect setată proprietatea *g* (folosită de algoritmul A*) **(0.2 puncte)**
5. Modificați funcția testeaza_scop ca să verifice că o stare îndeplinește condițiile din enunț pentru a fi finală (atenție, nu se vor enumera stări scop, ci se va verifica dacă informația din nodul dat ca parametru îndeplinește condițiile). **(0.2 puncte)**
6. Modificați funcția calculeaza_h adaugand o euristica **care să se foloseasca de costul unei mutări așa cum e definit în enunț.** Explicați într-un comentariu cum este afectată euristica de costul mutărilor. **(0.3 puncte)**
7. Completați codul astfel încât să se afișeze timpul de găsire a unei soluții cu algoritmul A*. Timpul se va afișa în **milisecunde. (0.1 puncte)**