Вештачка интелигенција-2021/2022/L

Quiz navigation

1 2

Finish attempt ...

Question **2**

Not complete

Marked out of 10.00

Flag question

Предложете соодветна репрезентација и напишете ги потребните функции во Python за да се реши следниот проблем за кој една можна почетна состојба е прикажана на Слика 1.

"На една лента составена од **L** полиња поставени се **N** дискови (**N** < **L**). Дисковите се меѓусебно различни и се нумерирани со целите броеви од 1 до **N.** На почетокот, дисковите се позиционирани во првите **N** полиња од лентата (гледајќи одлево - надесно), подредени во растечки редослед според нивните редни броеви (Слика 1 - почетна состојба за **N** = 3 и **L** = 7). Потребно е дисковите да се доведат на крајот на лентата (во последните **N** полиња од лентата, гледајќи одлево - надесно), при што ќе бидат подредени во опаѓачки редослед според нивните редни броеви (како пример, на Слика 2 е прикажана целната состојба која што соодветствува на почетната состојба прикажана на Слика 1). Во еден потег, еден диск може да се премести од полето во кое се наоѓа во соседно празно поле (лево или десно). Исто така, диск може да се премести и од полето во кое се наоѓа -> преку едно поле (во лево или десно), но само ако притоа „прескокнатото" поле содржи друг диск (на пример, може да се премести диск од првото во третото поле само ако третото поле е празно и второто поле содржи друг диск!). Не е дозволено дисковите да излегуваат од лентата. Потребно е проблемот да се реши во најмал број на потези."

За сите тест примери изгледот на лентата е ист како на примерот даден на сликите. За сите тест примери распоредот на дисковите на почетокот е ист (како што беше објаснето погоре). За секој тест пример се менува бројот на дискови, а исто така се менува и димензијата на лентата.

Од стандарден влез се вчитуваат влезните аргументи за секој тест пример. Најпрво е даден бројот на дискови (**N**), а потоа се чита димензијата на лентата (бројот на полиња од кои што е составена истата, **L**).

Движењата на дисковите потребно е да ги именувате на следниот начин:

- D1: Disk **i** - за преместување на дискот **i** надесно во соседно празно поле, **i** = 1, 2, ..., **N**
- D2: Disk **i** - за преместување на дискот **i** преку едно поле надесно, **i** = 1, 2, ..., **N**
- L1: Disk **i** - за преместување на дискот **i** налево во соседно празно поле, **i** = 1, 2, ..., **N**
- L2: Disk **i** - за преместување на дискот **i** преку едно поле налево, **i** = 1, 2, ..., **N**

Вашиот код треба да има само еден повик на функција за приказ на стандарден излез (print) со кој ќе ја вратите секвенцата на движења која треба да се направи за да може дисковите да се доведат на бараните позиции. Треба да применете неинформирано пребарување. Врз основа на тест примерите треба самите да определите кое пребарување ќе го користите.

НАПОМЕНА: Подреденоста на акциите во successor функција е важна кај неинформирано пребарување. Соодветно, за да се добие решението кое се очекува во изгенерираните излези, редоследот треба да биде D1, D2, L1, L2, за секое од полињата во лентата последователно, почнувајќи од почетокот. Доколку акциите не се подредени со истиот редослед, можно е да се најде исто оптимално решение со различна патека.

Слика 1:

1	2	3				

Слика 2:

				3	2	1

For example:

Input	Result
3 7	['D2: Disk 2', 'D1: Disk 1', 'D2: Disk 3', 'D1: Disk 1', 'D2: Disk 2', 'L1: Disk 3', 'D2: Disk 1', 'D2: Disk 1', 'D1: Disk 3']

Answer: (penalty regime: 0 %)

Reset answer

```
import bisect


"""
Дефинирање на класа за структурата на проблемот кој ќе го решаваме со пребарување.
Класата Problem е апстрактна класа од која правиме наследување за дефинирање на основните
карактеристики на секој проблем што сакаме да го решиме
"""


class Problem:
    def __init__(self, initial, goal=None):
        self.initial = initial
        self.goal = goal

    def successor(self, state):
        """За дадена состојба, врати речник од парови {акција : состојба}
        достапни од оваа состојба. Ако има многу следбеници, употребете
        итератор кој би ги генерирал следбениците еден по еден, наместо да
        ги генерирате сите одеднаш.
        :param state: дадена состојба
        :return:  речник од парови {акција : состојба} достапни од оваа
                  состојба
        :rtype: dict
        """
        raise NotImplementedError

    def actions(self, state):
        """За дадена состојба state, врати листа од сите акции што може да
        се применат над таа состојба
        :param state: дадена состојба
```

Check

Previous page

Finish attempt ...