Fakultatív feladat: többszintű ütemező megvalósítása

Beadható: 2023. április 20. (csütörtök)

Becsült programozási idő: >5 óra
A feladatbeküldés általános tudnivalói

Készítsen egy programot Java vagy Python nyelven, amely egy összetett ütemező működését szimulálja!

A globálisan preemptív, statikus prioritásos ütemező az alábbi ütemezési algoritmusokat futtatja az egyes szinteken az előadáson ismertetett módon:

- 1. magas prioritású szint (prioritás = 1) RR ütemező, időszelet: 2
- 2. alacsony prioritású szint (prioritás = 0) SRTF ütemező

Bemenet (standard input, stdin)

Soronként egy (max. 10) taszk adatai. Egy sor felépítése (vesszővel elválasztva):

- a taszk betűjele (A, B, C...)
- a taszk prioritása (0 vagy 1)
- a taszk indítási ideje (egész szám >= 0), a következő időszeletben már futhat (0: az ütemező indításakor már létezik), azonos ütemben beérkező új taszkok esetén az ABC-sorrend dönt
- a taszk CPU-löketideje (egész szám >= 1)

Példa:

```
A,0,0,6
B,0,1,5
C,1,5,2
D,1,10,1
```

A bemenet végét EOF jelzi (előtte soremelés biztosan van, és üres sor is előfordulhat).

Kimenet (standard output, stdout)

A kimenet első sorában a taszkok futási sorrendje betűjeleikkel (csak betűk, szóközök nélkül).
A második sorban a teljes várakozási idő taszkonként, **érkezésük** (nem feltétlenül abc-) sorrendjében, az alábbi formában (vesszővel elválasztva, szóközök nélkül):

1. taszk betűjel**:**várakozási idő,2. betűjel**:**várakozási idő, ...

Példa (a fenti bemenetre adott válasz):

```
ACABDB
A:2,B:8,C:0,D:0
```

Értékelés

Összesen 3 pont jár, ha minden teszten átmegy a megoldásuk. Arányosan kevesebb pont jár, ha nem minden esetben működik helyesen a programjuk.

Technikai információk

A programot a **HF portálon** kell leadni a megadott határidőig **egyetlen ZIP fájlba csomagolva**. A feltöltött fájlok (a ZIP és a tartalmának) neve ékezetes betűt és szóközt ne tartalmazzon!

A portál automatikusan kiértékeli a programot, és visszajelzést küld a működéséről. A kiértékelésre nincs időbeli garancia. A program bármikor újra beküldhető a határidőig. A beadási határidő leteltekor a legutolsó feltöltött program beadottá válik, annak értékelése adja a végleges pontszámot.

A beküldött **Java 8** programnak tartalmaznia kell egy "**Main**" nevű osztályt, melynek része a feladatot megoldó "main" függvény. A program tetszőleges számú forrásfájlból állhat. A program nem használhat a standard inputon és outputon kívül semmilyen más erőforrást, így nem végezhet fájlműveleteket és nem nyithat hálózati kapcsolatokat.

Python megoldás beküldése esetén a Python 3 verzióval kompatibilis megoldásokat fogadunk el, csak standard függvények használhatók (külső könyvtár, pl. NumPy nem), és a feltöltött ZIP fájlban egy db .py fájl lehet, más nem. Itt sem végezhetnek fájlműveleteket, nem nyithatnak meg hálózati kapcsolatokat.

Tesztadatok

Az első beküldés előtt érdemes az alábbi egyszerű tesztekkel megpróbálkozni.

```
A,1,2,7
B,1,2,3
ABABA
A:3,B:4

Q,0,5,8
P,1,7,2
QPQ
Q:2,P:0

A,0,0,5
B,0,0,4
C,0,1,3
D,0,2,1
BDBCA
A:8,B:1,C:4,D:0
A,0,0,3
B,1,0,2
C,0,3,3
D,1,4,1
BADAC
A:3,B:0,C:3,D:0
```

Egy ráadás, kicsit fogósabb:

```
A,0,0,5
B,0,1,3
C,1,1,1
D,0,4,1
E,1,3,2
ACBEDBA
A:7,B:4,C:0,E:0,D:1
```

Végül egy vitatható eset:

```
A,1,3,5
D,1,6,1
ADA
A:1,D:1
```

Vitatható, hogy ha A időszelete lejár, akkor egy teljesen újat kezdhet (és emiatt D-nek várnia kell egy ütemet), vagy folytathatja a futást, de ha közben jön egy másik taszk (D), akkor azonnal átütemezés van. A megoldásban járjunk el úgy, hogy új időszeletet kezdhet, azaz 4 időegységig fut, majd jön D egy időegysége, és végül A maradék egy lépése!

Utolsó módosítás: 2023. April 25., Tuesday, 11:52

Fakultatív feladat: lapcsere-algoritmus megvalósítása

Beadható: 2023. május 9. (kedd)

Becsült programozási idő: ~5 óra
A feladatbeküldés általános tudnivalói

Készítsen egy programot Java vagy Python nyelven, amely egy lapcsere-rendszer működését szimulálja!

A program bemeneteként memóriaműveletek során hivatkozott lapok azonosítóit kapja a hivatkozásuk sorrendjében. Kimeneteként a végrehajtott lapcserék eredményeképpen lefoglalt fizikai memóriakeretek azonosítóit és a laphibák számát adja vissza.

A rendszerben 3 memóriakeret található, amelyek kezdetben mind üresek. Az induláskor a lapok a cserehelyen találhatók.

A lapokat számok (1-99), a kereteket betűk (A,B és C) jelölik.

Bemenet (standard input, stdin)

Egyetlen sorban a lapokra történő hivatkozások egymástól vesszővel elválasztva. Például:

```
1,2,3,-1,5,-1
```

A negatív számok írási műveleteket jeleznek a megadott lapon; ekkor a keretek "dirty" jelzést kapnak (ez nem minden algoritmus esetén releváns információ, de a bemeneten előfordulhat).

A bemenet végét EOF jelzi (előtte soremelés, üres sor lehet). Ekkor kell a kimenetre kiírni az eredményt.

Kimenet (standard output, stdout)

A kimeneten az első sorban a bemeneti memóriahivatkozások kiszolgálásához lefoglalt memóriakeretek betűjelei szerepelnek a megfelelő sorrendben, szóközök nélkül, egybeírva, majd a következő sorban a laphibák száma. A kiírt eredmények előtt, után üres karakterek, további sorok ne legyenek!
Amennyiben egy memóriahivatkozáshoz nem kellett új keretet foglalni (már a memóriában volt a lap), a kimeneten az adott pozícióban "-" jel jelenik meg.
Ha egy memóriafoglalás nem teljesíthető (nincs szabad keret és egyetlen keret sem szabadítható fel), akkor a kimeneten "*" karakter jelenik meg (a műveletet nem ismétli meg az algoritmus). Ez utóbbi eset értelemszerűen nem minden algoritmusnál fordulhat elő.

Megvalósítandó algoritmus

Újabb esély (SC) lapcsere maximum 3 lépésig tartó tárba fagyasztással

A program írja ki az algoritmus szerinti memóriafoglalásokat és a laphibák számát!
Pl. a fenti bemenetre adott válasz:

```
ABC-AB
5
```

Értékelés

Összesen 3 pont jár, ha minden teszten átmegy a megoldás. Arányosan kevesebb pont szerezhető, ha nem minden esetben működik helyesen a beküldött program.

Technikai információk

A programot a **HF portálon** kell leadni a megadott határidőig **egyetlen ZIP fájlba csomagolva**. A feltöltött fájlok (a ZIP és a tartalmának) neve ékezetes betűt és szóközt ne tartalmazzon!

A portál automatikusan kiértékeli a programot, és visszajelzést küld a működéséről. A kiértékelésre nincs időbeli garancia. A program bármikor újra beküldhető a határidőig. A beadási határidő leteltekor a legutolsó feltöltött program beadottá válik, annak értékelése adja a végleges pontszámot.

A beküldött Java 8 programnak tartalmaznia kell egy "**Main**" nevű osztályt, melynek része a feladatot megoldó "main" függvény. A Java program tetszőleges számú forrásfájlból állhat. A program nem használhat a standard inputon és outputon kívül semmilyen más erőforrást, így nem végezhet fájlműveleteket és nem nyithat hálózati kapcsolatokat.

Python megoldás beküldése esetén a Python 3 verzióval kompatibilis megoldásokat fogadunk el, csak standard függvények használhatók (külső könyvtár, pl. NumPy nem), és a feltöltött ZIP fájlban egy db .py fájl lehet, más nem. Itt sem végezhetnek fájlműveleteket, nem nyithatnak meg hálózati kapcsolatokat.

Tesztadatok

Az első beküldés előtt érdemes az alábbi tesztekkel megpróbálkozni.

```
1,2,3,5,4,2
ABC*A-
5
1,2,3,2,4,3,2,1
ABC-A--B
5
1,2,3,3,4,5,2,1
ABC-ABC*
7
1,2,3,4,5,4,3,2,1
ABC*AB-CA
8
-5,2,5,3,2,1,-3
AB-C-A-
4
```

A fagyasztott keretek kezelése vitatható. Eljárhatnánk úgy, mintha újabb esélyt kapnának (azaz a FIFO végére kerülnek), de ez nem tükrözné az SC abbéli törekvését, hogy valahol a használati időt veszi figyelembe. Ezért az ilyen kereteket hagyjuk meg a helyükön a FIFO-ban (csak ne használjuk azokat)! Példa:

```
1,2,3,4,1,5,1,3,6,3
ABC*-B--CB
7
```

Utolsó módosítás: 2023. April 25., Tuesday, 11:49

Fakultatív feladat: holtpontelkerülés

Beadható: 2023. június 5. (hétfő)

Becsült programozási idő: ~5 óra
A feladatbeküldés általános tudnivalói

Készítsen egy programot Java vagy Python nyelven, amely erőforrás-allokációs gráf segítségével holtpontelkerülést valósít meg!

A szimulátor a standard input bemenetről fogadja taszkok nevét és utasításait (lásd lentebb), majd azok futtatásával szimulálja erőforrások allokációját. A program kimenete az elkerült holtpontok adatai (időrendben).

A szimuláció lépésekben történik. Minden lépésben minden taszk egy utasítást hajt végre (amennyiben nem várakozik egy erőforrásra). Az utasítás lehet erőforrás kérése, felszabadítása és "egyéb" művelet. Egy lépésen belül az utasítások végrehajtása a taszkok bemenetben meghatározott sorrendje szerint történik. Ha egy taszknak egy lépésben már nincs több végrehajtható művelete, akkor véget ér, és a lefoglalt állapotban maradt erőforrásai a lefoglalásaik sorrendjében felszabadulnak.

A szimulátor a taszkok műveletei alapján felépít egy erőforrás-allokációs gráfot, és annak segítségével minden foglalásnál ellenőrzi, hogy az holtponthoz vezet-e. Amennyiben igen, úgy a foglalást visszautasítja. Az így visszautasított foglalási műveletet a taszk nem ismétli meg később, és nem is blokkolódik miatta; a következő lépésben a következő utasítását hajtja majd végre. Amennyiben egy taszk általa nem lefoglalt erőforrást szabadítana fel (pl. egy korábban visszautasított foglalása miatt), akkor semmi sem történik (mintha ott "0" szerepelne a bemenetén).

A feladat során egypéldányos erőforrásokat foglalhatnak a taszkok, így a holtpontot az erőforrás-allokációs gráf elemzésével lehet detektálni. Minden foglalási igény kiszolgálása előtt ellenőrizni kell, hogy az az egy lépés holtpontot okoz-e (azaz a gráfban kialakul-e kör a lépés után). Egy taszk erőforráskérése háromféle kimenetet eredményezhet: az erőforrás szabad és nincs holtpont (normál visszatérés); az erőforrás szabad, de holtpont alakulna ki (a szimulátor visszautasítja a kérést); és az erőforrás foglalt (a szimulátor a taszkot várakozó állapotba helyezi).

Egy erőforrás felszabadítása során a szimulátor az arra várakozó taszkok közül a FIFO-elv szerint választ, azaz a legrégebben várakozó taszk kapja meg az erőforrást, és az a következő alkalommal (amikor a szimulátor ezt a taszkot futtatja) végrehajtja majd a következő utasítását.

Bemenet (standard input, stdin)

Soronként egy-egy taszk nevét és ütemenkénti utasításait tartalmazza a következők szerint:

- T1, T2, T3, ... taszkok nevei (egy szóköz nélküli karakterfüzér)
- R1, R2, R3, ... erőforrások nevei (egy szóköz nélküli karakterfüzér)
- "+R1": erőforrás-foglalási kérés, "-R1": erőforrás-felszabadítási utasítás, "0" egyéb művelet

pl.: a "T1,+R1,0,0,+R2,-R1,-R2" sorozat értelmezése: a T1 taszk elsőként szeretné lefoglalni R1-et, azután két ütemben más tevékenységet végez, majd foglalási kérést ad ki R2-re, azután felszabadítja R1-et, végül R2-t is.

Egy teljes bemeneti példa:

```
T1,+R1,0,0,+R2,-R1,-R2
T2,+R2,+R1,-R1,-R2
T3,0,0,0,+R3,+R3,-R3,-R3
```

A bemenet végén soremelés, üres sor lehet.

Kimenet (standard output, stdout)

A kimeneten soronként egy-egy elkerült holtpont részleteit kell megadni az alábbi formátumban: taszk neve (amelyik a holtpontot okozná), a taszk sorrendben hányadik műveletét utasította el a szimulátor, erőforrás neve (amelyiknek a lefoglalása holtpontot okozott volna). Például:

```
T1,4,R2
T3,5,R3
```

A fenti példa értelmezése: a szimulátor két utasításnál detektált holtpontot: T1 4. utasítása (R2 erőforrás foglalási kérelme) és T3 5. utasítása (R3 foglalási kérelme) során.

Értékelés

Összesen 3 pont jár, ha minden teszten átmegy a megoldás. Arányosan kevesebb pont szerezhető, ha nem minden esetben működik helyesen a beküldött program.

Technikai információk

A programot a **HF portálon** kell leadni a megadott határidőig **egyetlen ZIP fájlba csomagolva**. A feltöltött fájlok (a ZIP és a tartalmának) neve ékezetes betűt és szóközt ne tartalmazzon!

A portál automatikusan kiértékeli a programot, és visszajelzést küld a működéséről. A kiértékelésre nincs időbeli garancia. A program bármikor újra beküldhető a határidőig. A beadási határidő leteltekor a legutolsó feltöltött program beadottá válik, annak értékelése adja a végleges pontszámot.

A beküldött Java 8 programnak tartalmaznia kell egy "**Main**" nevű osztályt, melynek része a feladatot megoldó "main" függvény. A Java program tetszőleges számú forrásfájlból állhat. A program nem használhat a standard inputon és outputon kívül semmilyen más erőforrást, így nem végezhet fájlműveleteket és nem nyithat hálózati kapcsolatokat.

Python megoldás beküldése esetén a Python 3 verzióval kompatibilis megoldásokat fogadunk el, csak standard függvények használhatók (külső könyvtár, pl. NumPy nem), és a feltöltött ZIP fájlban egy darab .py fájl lehet, más nem. Itt sem végezhetnek fájlműveleteket, nem nyithatnak meg hálózati kapcsolatokat.

Tesztadatok

Az első beküldés előtt érdemes az alábbi tesztekkel megpróbálkozni (bemenet - kimenet párok).

```
T1,+R1,+R1
```

```
T1,2,R1
```

Magyarázat: a T1 még egy példányt kér az R1-ből a 2. lépésben, de csak egy példány van minden erőforrásból.

```
T1,+R1,+R2,-R1,-R2
T2,+R2,+R1,-R2,-R1

T2,2,R1

T1,+R1,-R1
T2,+R3,+R4,0,+R1,-R4,0
T3,+R3,0,+R4
T4,+R2,+R4
T5,0,+R3,+R4,0
T6,+R1,+R3,+R4,0,0

T2,4,R1
```