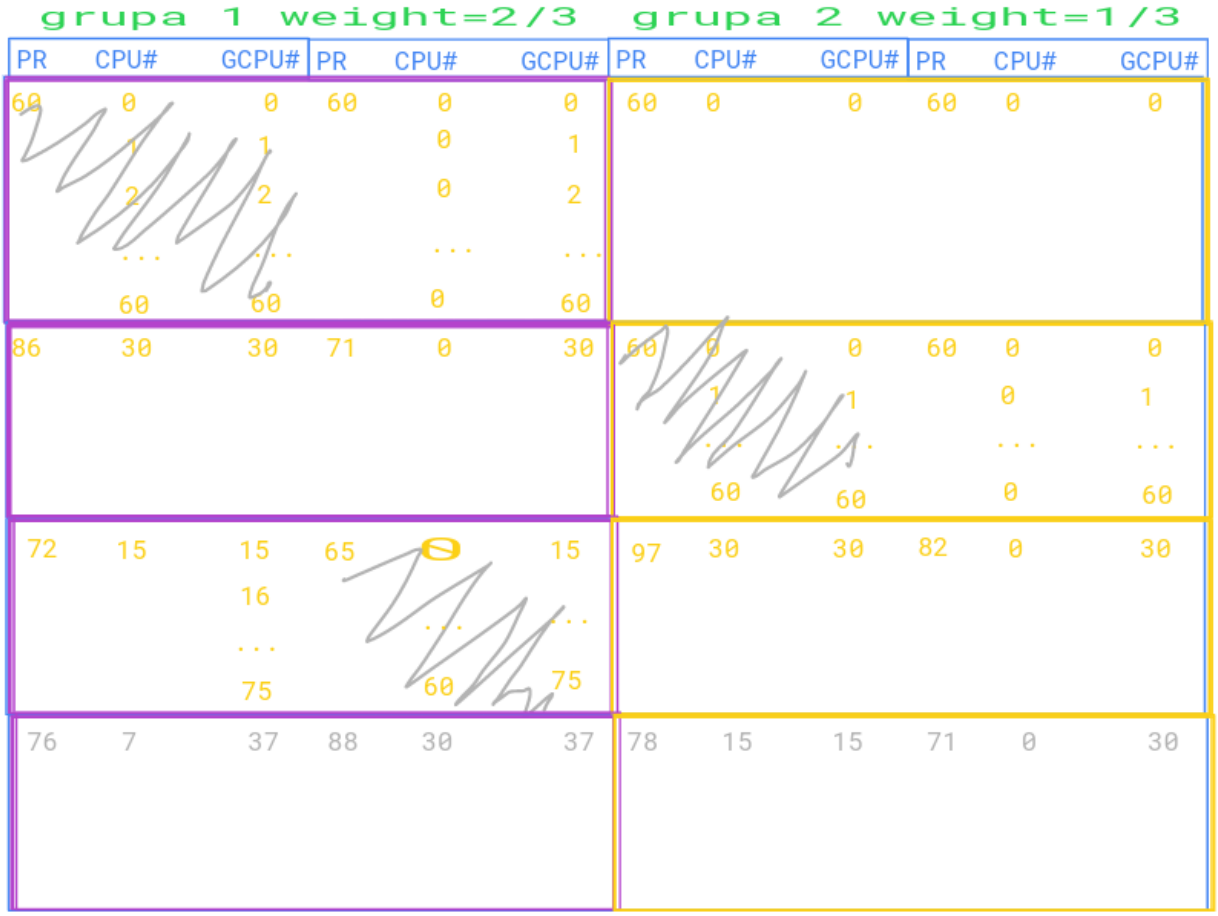
1. **Naloga**

   Na sistemu zaganjamo dve aplikaciji, vsaka ima dve niti. Prekinitve za namen štetja izvajanj niti in aplikacije se pojavljajo 60-krat v sekundi. Preračun prioritet se zgodi enkrat na sekundo. Osnovna prioriteta ob zagonu je 60. Prva aplikacija naj bo za izvajanje izbrana v 2/3 primerov. Sledite (skicirajte) izvajanju algoritma Pravično razporejanje (angl. *Fair Share Scheduling*) od časa 0 sekund do časa 3 sekund. Na koncu poračunajte še vrednosti v vrstici takoj za časom t = 3 sekunde.




```
GCPUi# = GCPU(i-1)#/2
CPUi# = CPU(i-1)#/2
PR = PRB + CPUi#/2 + GCPUi#/4*weight
```

prvič

```
PR = 60 + 0 + 30/(4*(1/3)) = 71
PR = 60 + 15 + 30/(4*(2/3)) = 86
```

drugič

```
PR = 60 + 0 + 15/(4*(1/3)) = 65
PR = 60 + 7 + 15/(4*(2/3)) = 72
```

tretjič

```
PR = 60 + 7 + 37/(4*(2/3)) = 88
PR = 60 + 3 + 37/(4*(2/3)) = 76
PR = 60 + 0 + 30/(4*(1/3)) = 71
PR = 60 + 7 + 15/(4*(1/3)) = 78
```

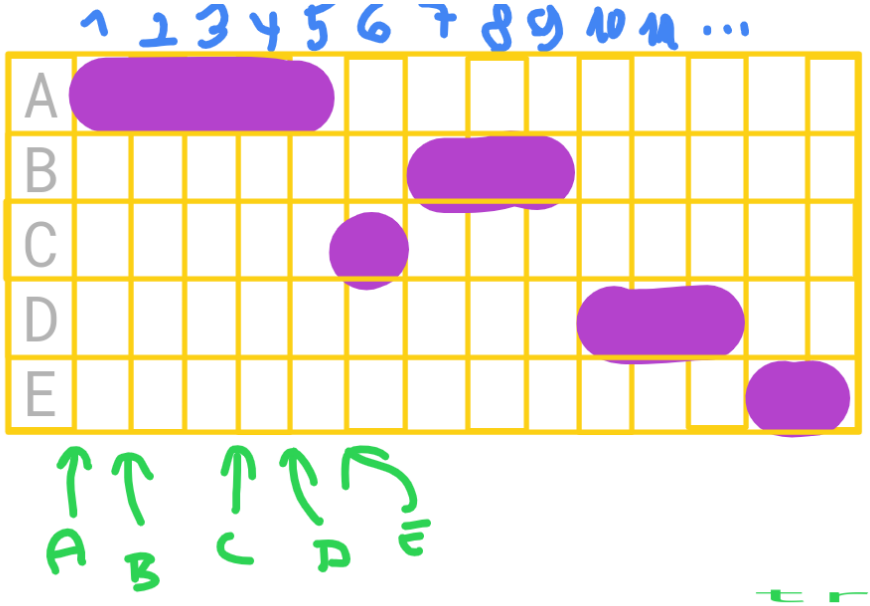
2. **Naloga**

Imamo en procesor in več procesov, ki jih želimo izvajati na procesorju. Čas prispetja in čas izvajanja posameznega procesa je sledeč:

proces	čas prispetja	čas izvajanja
A	0	5
B	1	3
C	3	1
D	4	3
E	5	2

Skicirajte kratkoročno razporejanje:

- po kriteriju najprej tisti z največjim odzivnim razmerjem (angl. *Highest Response Ratio Next*)



prvic

B=(4+3)/3=7/3 <- response rate
C=(2+1)/1=3
D=(1+3)/3=4/3
E=(0+2)/2=1

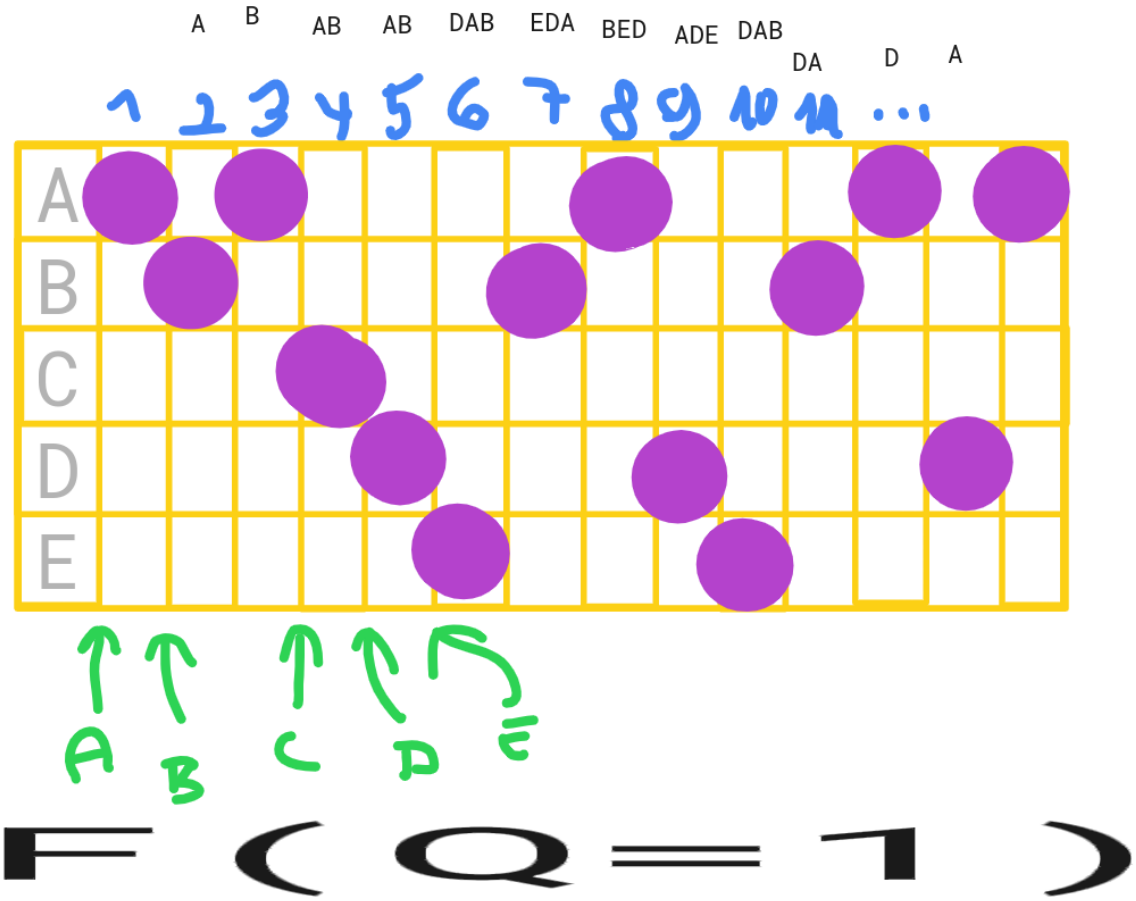
drugic

B=(5+3)/3=8/3
D=(2+3)/3=5/3
E=(1+2)/2=3/2

tretjic

D=(5+3)/3=8/3
E=(4+2)/2=3

- **po kriteriju povratnega odgovora (s kvantumom 1 in dvema vrstama) (angl. Feedback).**



- **Primerjajte rezultate obeh algoritmov – izračunajte in primerjajte povprečen normaliziran obračalni čas. Kateri algoritem je bil v danem primeru boljši?**
  1. → (1 + 8/3 + 3 + 10/3 + 3)/5 = 2.6
  2. → (14/5 + 11/3 + 1 + 3 + 5/2)/5 = 2.53

     Boljši je Feedback.

3. **Naloga**

V sistemu imamo opravka s štirimi periodičnimi procesi. Prvi ima čas izvajanja 20 ms in periodo 100 ms. Drugi ima čas izvajanja 25 ms in periodo 100 ms. Tretji ima čas izvajanja 40 ms in periodo 200 ms. Četrti pa ima čas izvajanja 15 ms in periodo 150 ms. Ali sistem ulovi vse roke? Argumentirajte z izračunom! Poimenujete algoritem! (Pomoč, da ne potrebujemo kalkulatorjev: $\sqrt[4]{2} \approx 1,19$.)

Ime algoritma je monotono razporejanje.

```
i = 1, 2, 3, 4

i = Ci/Pi => 20/100, 25/100, 40/200, 15/150
```

$$\sum u_i = 0.75$$

$$n \times (\sqrt[n]{2} - 1) = 0.76$$

$$\sum u_i < n \times (\sqrt[n]{2} - 1)$$

$$0.75 < 0.76$$

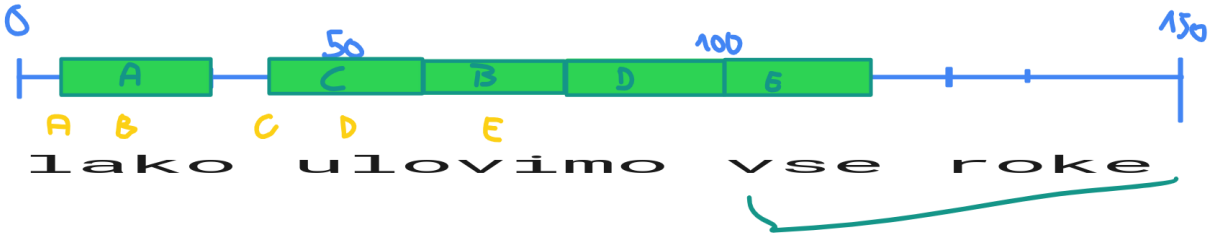
✓

4. **Naloga**

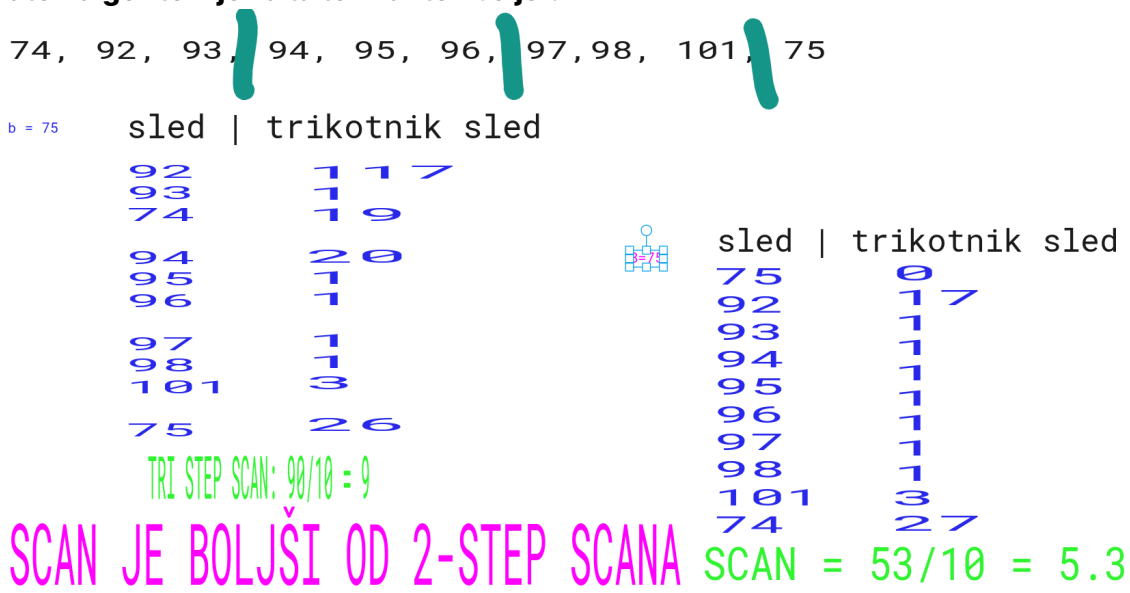
Po algoritmu realno časovnega razporejanja po principu prvega roka z neizsiljenim časom čakanja (angl. *earliest deadline with unforced idle times*) narišite časovnico izvajanja 20 ms procesov brez preklapljanja, iz katere so jasno razvidne dolžine in vrstni red izvajanj za naslednje procese:

proces	čas prispetja [ms]	rok pričetka izvajanja [ms]
A	10	30
B	20	120
C	40	40
D	50	90
E	70	110

Ali ulovimo vse roke?



5. **Naloga**
    1. **Kakšna je učinkovitost 3-koračnega skeniranja (angl. *3-step-scan*), če ima disk 150 sledi in je na začetku glava diska na sledi 75? Na začetku se glava premika proti večjim sledem, znotraj posameznega koraka pa ohranja na začetku smer prejšnjega koraka. Razporejevalnik je dobil zahteve po sledeh v naslednjem vrstnem redu: 74, 92, 93, 94, 95, 96, 97, 98, 101, 75.**
    2. **Kakšna pa je učinkovitost v primeru algoritma scan za ta tok zahtev? (Pri tem tudi pravilno poimenujte izračunano metriko učinkovitosti!)**
    3. **Kateri algoritem je za ta tok zahtev boljši?**



6. **Naloga**

Na preprostem vgradnem računalniku poganjamo tri procese P1–P3, ki potrebujejo vire R1–R6. Potrebe so podane v matriki C, trenutne alokacije virov pa v matriki A. Pri obeh matrikah so procesi podani po vrsticah, viri pa po stolpcih. Stevnost virov v tem računalniku je podana z vektorjem R.

$$C = \begin{bmatrix} 1 & 3 & 2 & 1 & 1 & 0 \\ 3 & 2 & 1 & 1 & 1 & 1 \\ 3 & 0 & 2 & 4 & 2 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 1 & 1 & 1 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 4 & 2 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} 8 & 6 & 3 & 5 & 4 & 1 \end{bmatrix}$$

Ali lahko poženemo tudi proces $P4, C(P4) = [110010]$ brez nevarnosti smrtnega objema? Odgovor argumentirajte!

Lahko, ker slika spodaj.