Άσκηση 1

Δίνεται ένα σύστημα Linux που χρησιμοποιεί τον χρονοδρομολογητή O(1) και 4 διεργασίες Α,Β,Γ,Δ με αριθμούς προτεραιοτήτων 109, 110, 109 και 104 αντίστοιχα. Οι διεργασίες κατέφθασαν σε χρόνο t=0. Να δείξετε τον τρόπο χρονοδρομολόγησής τους αν καθεμία από αυτές υποτίθεται ότι θα εκτελεστεί για 1000ms, καθώς και την κατάσταση των ουρών active/expired στο τέλος της χρονοδρομολόγησης. Ο τρόπος υπολογισμού των κβάντων ακολουθεί την πολιτική του O(1) χρονοδρομολογητή και οι διεργασίες είναι σε κατάσταση sleep κατά 50%, 25%,25%, και 50%, αντίστοιχα, κάθε φορά που εκτελούνται. Στο ενδιάμεσο, θεωρήστε ότι δεν εισέρχονται νέες διεργασίες στο σύστημα

ΛΥΣΗ

O(1) φτάνουν σε t=0 και εκτελούνται για 1000ms

Διεργασία	Προτεραιότητα	Sleep
Α	109	50%
Β	110	25%
Γ	109	25%
Δ	104	50%

Διεργασία	Προτεραιότητα	Sleep Time
Δ	104	50% * 1000 = 500 ms
Α	109	50% * 1000 = 500 ms
Γ	109	25% * 1000 = 250 ms
Β	110	25% * 1000 = 250 ms

Διεργασία Δ: Sleep Time = 500 ms άρα Bonus = 5
Άρα DP = max[100, { min(104-5+5,139)}]=max[100, { min(104,139)}]=**104**
Θα πάει στην 104 της expired
Νέα κβάντα (140-104)*20 = 720 ms

Διεργασία Α: Sleep Time = 500 ms άρα Bonus = 5
Άρα DP = max[100, { min(109-5+5,139)}]=max[100, { min(109,139)}]=**109**
Θα πάει στην 109 της expired
Νέα κβάντα (140-109)*20 = 620 ms

Διεργασία Γ: Sleep Time = 250 ms άρα Bonus = 2
Άρα DP = max[100, { min(109-2+5,139)}]=max[100, { min(112,139)}]=**112**
Θα πάει στην 112 της expired
Νέα κβάντα (140-112)*20 = 560 ms

Διεργασία Β: Sleep Time = 250 ms άρα Bonus = 2
Άρα DP = max[100, { min(110-2+5,139)}]=max[100, { min(113,139)}]=**113**
Θα πάει στην 113 της expired
Νέα κβάντα (140-113)*20 = 540 ms

ΠΡΙΝ: BITMAP[100-115]: 0000100001100000
ΜΕΤΑ: BITMAP[100-115]: 0000100001001100

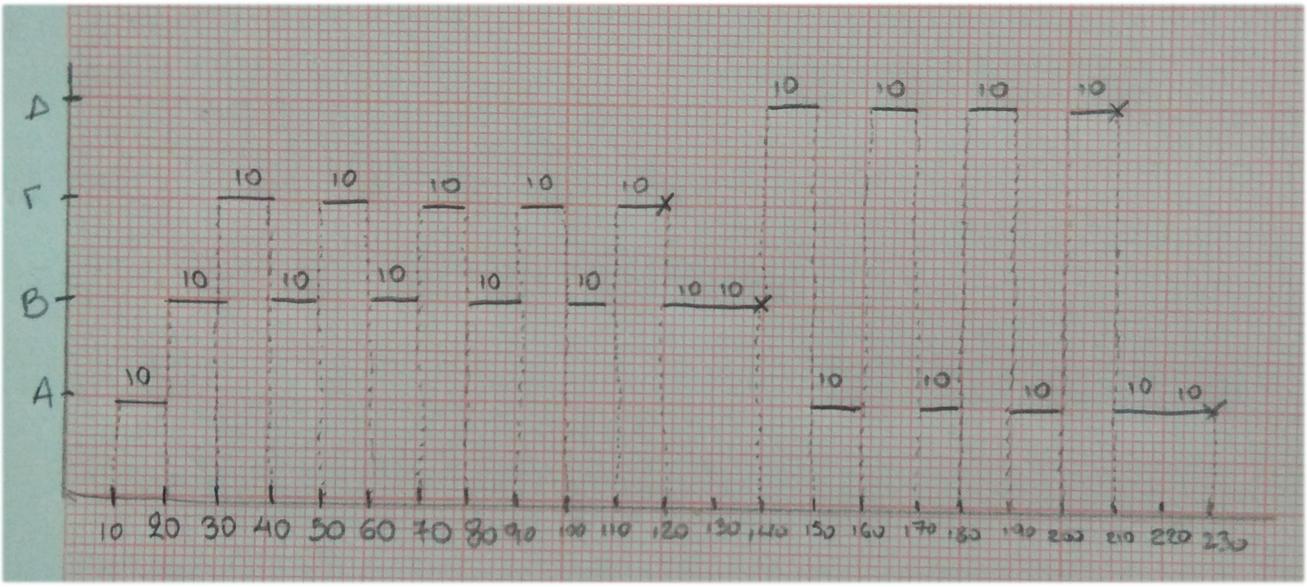
Άσκηση 2

Δίνονται 4 διεργασίες των οποίων οι χρόνοι άφιξης και εκτέλεσης δίνονται παρακάτω:

	Άφιξη	Εκτέλεση	Προτεραιότητα
Α	10	60	1
Β	20	70	0
Γ	20	50	0
Δ	30	40	1

Οι Β και Γ έχουν μεγαλύτερη προτεραιότητα από τις Α και Β (δηλαδή η προτεραιότητα 1 είναι μικρότερη της 0). Μεταξύ ίσων σε προτεραιότητα διεργασιών ακολουθείται ο αλγόριθμος **RR** με q=10. Να υπολογίσετε το μέσο χρόνο αναμονής και το μέσο χρόνο παραμονής των διεργασιών στο σύστημα καθώς και τους αντίστοιχους σταθμισμένους χρόνους

ΛΥΣΗ



- Τρέχει η Α για 10 (10-20).
- Μπαίνει η Β για 10 (20-30).
- Μπαίνει η Γ γιατί έχει μεγαλύτερη προτεραιότητα από την Δ (30-40)
- Συνεχίζει η Β για 10 (40-50)
- Γίνεται εναλλάξ για 10 η Γ και η Β μέχρι το **120** όπου η *Γ τελειώνει.*
- Η *Β* τρέχει για 10 + 10 (120-140) όπου και *τελειώνει* στο **140**
- Μπαίνει η Δ στο 140 και πάει εναλλάξ Δ - Α μέχρι να *τελειώσει η Δ* στο **210**
- Τρέχει η Α από το 210 άλλα 10 + 10 και *τελειώνει* στο **230**

TT = (χρόνος προηγούμενης διεργασίας + χρόνος εκτέλεσης της τρέχουσας διεργασίας) – Χρόνος άφιξης

$TT_A = 230 – 10 = 220$

$TT_B = 140 – 20 = 120$

$TT_{\Gamma} = 120 – 20 = 100$

$TT_{\Delta} = 210 – 30 = 180$

ATT = (220+120+100+180)/4 = 155

WT = Συνολικός χρόνος παραμονής – χρόνος εκτέλεσης (TT - RT)

$WT_A = 220 – 60 = 160$

$WT_B = 120 – 70 = 50$

$WT_{\Gamma} = 100 – 50 = 50$

$WT_{\Delta} = 180 – 40 = 140$

AWT = (160+50+50+140)/4 = 100

$WTT_A = 220/60 = 3{,}6$

$WTT_B = 120/70 = 1{,}7$

$WTT_{\Gamma} = 100/50 = 2$

$WTT_{\Delta} = 180/40 = 4{,}5$

$WWT_A = 160/60 = 2{,}6$

$WWT_B = 50/70 = 0{,}71$

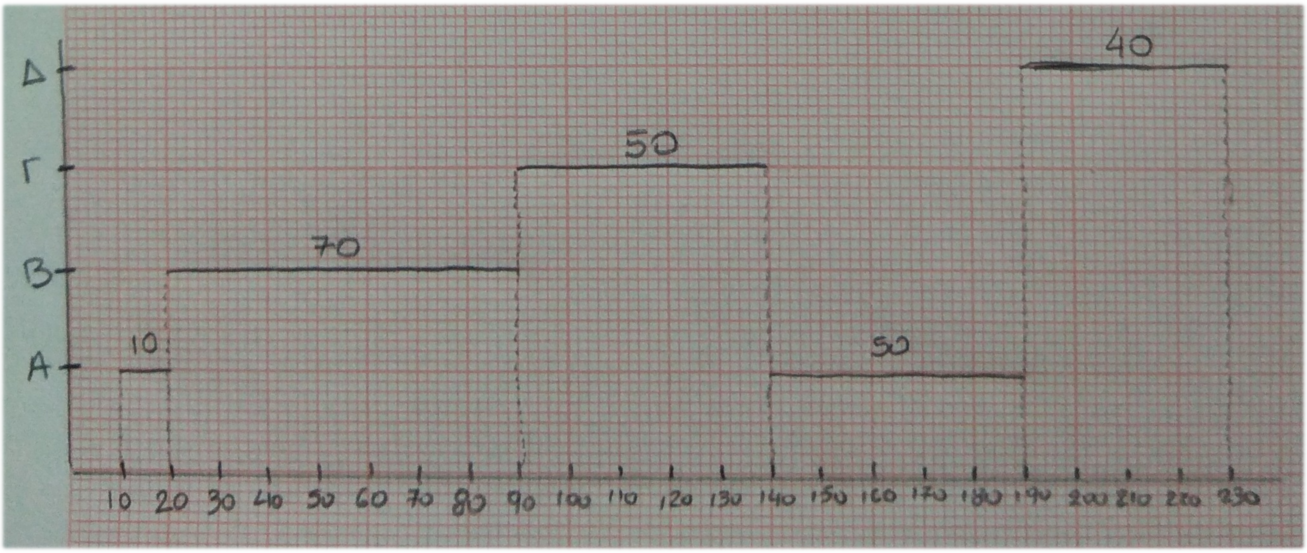
$WWT_{\Gamma} = 50/50 = 1$

$WWT_{\Delta} = 140/40 = 3{,}5$

Άσκηση 3

Να επαναλάβετε την Άσκηση 2, αλλά αυτή τη φορά να χρησιμοποιήσετε αλγόριθμο **FIFO** μεταξύ διεργασιών με ίση προτεραιότητα.

ΛΥΣΗ



- Έρχεται η Α και τρέχει για **10**. Σταματάει στο *20* γιατί έρχεται η Β με μεγαλύτερη προτεραιότητα.
- Έρχεται η Β και η Γ με μεγαλύτερη προτεραιότητα από την Α, οπότε θα τρέξουν πιο μπροστά της Α. Τρέχει πρώτα η Β για **70** και τελειώνει.
- Στη συνέχεια τρέχει η Γ για **50** και τελειώνει.
- Μπαίνει η Α και τρέχει άλλα 50 που μένουν και τελειώνει.
- Μπαίνει η Δ για 40 και τελειώνει (είχε έρθει στο 30 αλλά περίμενε την Β και Γ με μεγαλύτερη προτεραιότητα και την Α που ήδη είχε ξεκινήσει να τρέχει).

TT = (χρόνος προηγούμενης διεργασίας + χρόνος εκτέλεσης της τρέχουσας διεργασίας) – Χρόνος άφιξης

TT_A = 190 – 10 = 180

TT_B = 90 – 20 = 70

$TT_Γ$ = 140 – 20 = 120

$TT_Δ$ = 230 – 30 = 200

ATT = (180+70+120+200)/4 = 570/4 = 142,5

WT = Συνολικός χρόνος παραμονής – χρόνος εκτέλεσης (TT - RT)

WT_A = 180 – 60 = 120

WT_B = 70 – 70 = 0

$WT_Γ$ = 120 – 50 = 70

$WT_Δ$ = 200 – 40 = 160

AWT = (120+0+70+160)/4 = 350/4 = 87,5

WTT_A = 180/60 = 3

WTT_B = 70/70 = 1

$WTT_Γ$ = 120/50 = 2,4

$WTT_Δ$ = 200/40 = 5

WWT_A = 120/60 = 2

WWT_B = 0/70 = 0

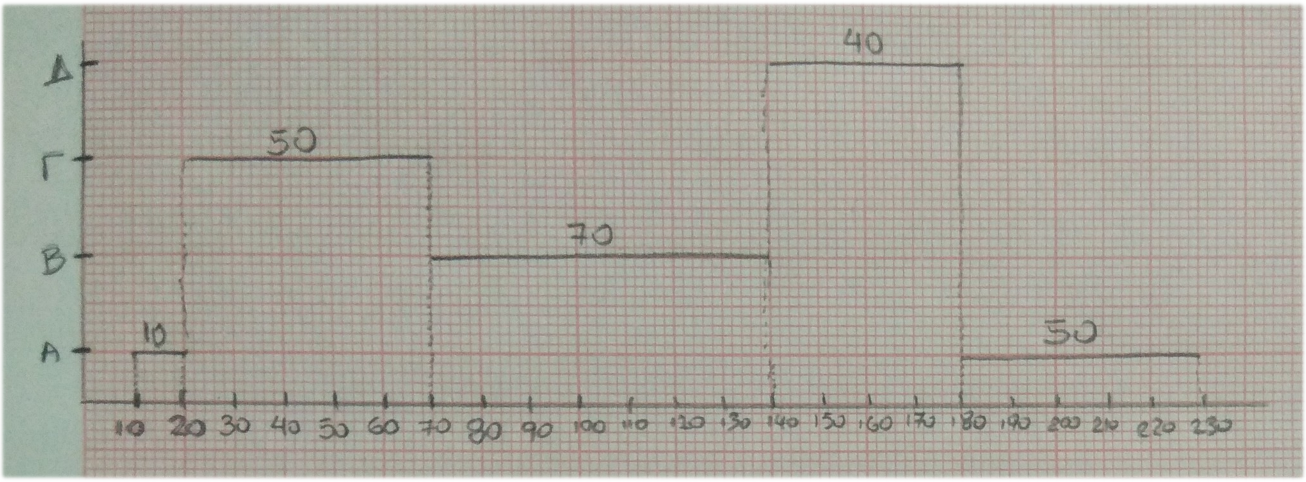
$WWT_Γ$ = 70/50 = 1,4

$WWT_Δ$ = 160/40 = 4

Άσκηση 4

Να επαναλάβετε την Άσκηση 2, αλλά αυτή τη φορά να χρησιμοποιήσετε αλγόριθμο **SRTN** μεταξύ διεργασιών με ίση προτεραιότητα.

ΛΥΣΗ



- Μπαίνει η Α και τρέχει μέχρι το **20**, όπου έρχονται οι Β και Γ. Στην Α απομένουν 60-10 = 50
- Οι Β και Γ έρχονται στο ίδιο χρονικό σημείο και έχουν ίδια προτεραιότητα, αλλά η Γ έχει λιγότερο εναπομείναντα χρόνο (50) έναντι της Β (70), επομένως θα τρέξει πρώτη. Θα τρέξει για 50 και θα τελειώσει στο **70**.
- Στην συνέχεια θα τρέξει η Β για 70 μέχρι να τελειώσει στο **140**.
- Στο 30 έχει έρθει και η Δ και πρέπει να τρέξει για 40. Η Α αν και έχει ίδια προτεραιότητα με την Δ, έχει περισσότερο εναπομείναντα χρόνο, δηλαδή 50. Άρα θα τρέξει η Δ από το 140 για 40 και θα τελειώσει στο **180**.
- Η Α θα ξεκινήσει στο 180 και θα τρέξει για άλλα 50, όπου θα τελειώσει στο **230**.

TT = (χρόνος προηγούμενης διεργασίας + χρόνος εκτέλεσης της τρέχουσας διεργασίας) – Χρόνος άφιξης

TT_A = 230 – 10 = 220
TT_B = 140 – 20 = 120
TT_Γ = 70 – 20 = 50
TT_Δ = 180 – 30 = 150
ATT = (220+120+50+150)/4 = 540/4 = 135

WT = Συνολικός χρόνος παραμονής – χρόνος εκτέλεσης (TT - RT)

WT_A = 220 – 60 = 160
WT_B = 120 – 70 = 50
WT_Γ = 50 – 50 = 0
WT_Δ = 150 – 40 = 110
AWT = (160+50+0+110)/4 = 320/4 = 80

WTT_A = 220/60 = 3,6
WTT_B = 120/70 = 1,7
WTT_Γ = 50/50 = 1
WTT_Δ = 150/40 = 3,75

WWT_A = 160/60 = 2,6
WWT_B = 50/70 = 0,71
WWT_Γ = 0/50 = 0
WWT_Δ = 110/40 = 2,75

Άσκηση 5

Δίνονται 6 διεργασίες P0-P5 που ξεκινούν από την ουρά με προτεραιότητα 139. Όταν τρέξουν για πρώτη φορά, οι τρεις κάνουν I/O 50% του χρόνου ενώ οι άλλες τρεις 80%. Στη συνέχεια, εμφανίζονται τρεις διεργασίες, οι οποίες έχουν προτεραιότητα 100. Αν υποθέσουμε ότι ο χρόνος είναι αρχικά μηδενικός, πότε θα τρέξουν για $2^{η}$ φορά οι διεργασίες της ουράς 139;

ΛΥΣΗ

P_0, P_1, P_2 }50% I/O

P_3, P_4, P_5 }80% I/O

Δίνεται χρόνος 1000ms σε κάθε διεργασία για να καθορίσει το σύστημα ποιο θα είναι το αρχικό Bonus.

- Για τις P0, P1, P3 έχουμε Sleep Time 500ms άρα το Bonus = 5 (ουδέτερη συμπεριφορά διότι DP=SP) και παραμένουν στην ίδια ουρά.
- Για τις P3, P4, P5 έχουμε Sleep Time 800ms, άρα το Bonus = 8. Η νέα προτεραιότητα θα είναι: DP = max[100,{min(139 – 8 + 5, 139)}] = max[100,{min(136, 139)}] => **DP = 136**
- Έστω ότι έρχονται 3 νέες διεργασίες P6, P7, P8 και τρέχουν για 1sec η κάθε μία.
- Υπολογίζουμε πόσο θα τρέξουν οι διεργασίες με προτεραιότητα 136 ως εξής:

(140 – 136) * 5 = 20ms η κάθε μία

Άρα οι διεργασίες της ουράς 139 θα τρέξουν για 2η φορά σε χρόνο:

(6 * 1000) + (3 * 1000) + (3 * 20) = 9060 ms ή 9,06 sec

Άσκηση 6

Δίνονται 6 διεργασίες P0-P5 έρχονται με τη σειρά με αρχική τι μη vruntime=1. Υποθέτουμε ότι το MG=4 ms. Επίσης, το TL= 24 ms (αυξάνεται σε σχέση με το default για να τρέξουν όλες οι διεργασίες από 4ms). Οι τιμές nice είναι

P0=-10, P1=-5, P2=0, P3=1, P4=4, P5=5,

A) Να τοποθετήσετε αυτές τις διεργασίες στο RB-TREE όταν αυτές εκτελεστούν για vruntime.
B) Να βρείτε τα νέα vruntime και τα κβάντα που θα πάρουν
C) Έστω ότι μπαίνουν τρεις νέες διεργασίες P6-P8 με τιμές nice -19, -18, -17 . Να τις τοποθετήσετε στο δέντρο αν έχουν αρχική τιμή vruntime =1.
D) Οι νέες διεργασίες θα τρέξουν για 4 ms, ενώ οι άλλες για τον αριθμό κβάντων που τους έχει δοθεί. Να δείξετε την κατάσταση του RB-Tree μετά από αυτή την εκτέλεση.

ΛΥΣΗ

Δεδομένα:
6 διεργασίες P0-P5
MG=4 ms
TL= 24 ms (όλες τρέχουν από 4 ms)
nice: P0=-10, P1=-5, P2=0, P3=1, P4=4, P5=5

ΥΠΟΛΟΓΙΣΜΟΣ vruntime [VR = VR + (t *w) και $w = 1{,}25^{nice} * 1024$]

$P0 = 1 + [(4) * 1{,}25^{-10} *1024] = 440$

$P1 = 1 + [(4) * 1{,}25^{-5} *1024] = 1343$

$P2 = 1 + [(4) * 1{,}25^{0} *1024] = 4097$

$P3 = 1 + [(4) * 1{,}25^{1} *1024] = 5121$

$P4 = 1 + [(4) * 1{,}25^{4} *1024] = 10001$

$P5 = 1 + [(4) * 1{,}25^{5} *1024] = 12501$

Νέα κβάντα
($K=1024 / 1{,}25^{nice}$)

$K_{P0} = 1024 / 1{,}25^{-10} = 9536$

$K_{P1} = 1024 / 1{,}25^{-5} = 3125$

$K_{P2} = 1024 / 1{,}25^{0} = 1024$

$K_{P3} = 1024 / 1{,}25^{1} = 819$

$K_{P4} = 1024 / 1{,}25^{4} = 419$

$K_{P5} = 1024 / 1{,}25^{5} = 335$

ΣΥΝΟΛΟ M = 15258

Για κάθε διεργασία δίνουμε χρόνο από **TL(K/M)**

$P0 = (9536 / 15258) * 24 = 14{,}9$

$P1 = (3125 / 15258) * 24 = 4{,}9$

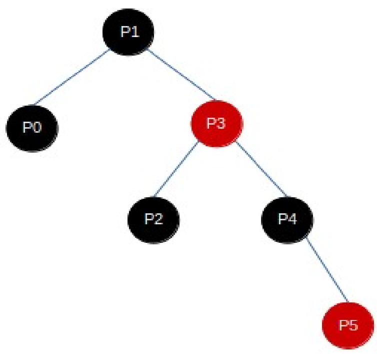
$P2 = (1024 / 15258) * 24 = 1{,}6$

$P3 = (819 / 15258) * 24 = 1{,}2$

$P4 = (419 / 15258) * 24 = 0{,}6$

$P5 = (335 / 15258) * 24 = 0{,}5$

Red Black Tree



Θα μπουν άλλες 3 διεργασίες P6-P8 και υπολογίζω το vruntime

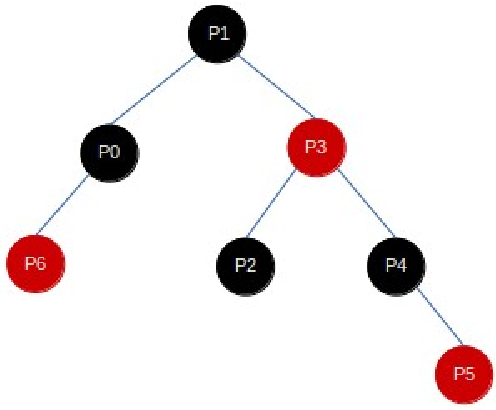
$P6 = 1 + [(4) * 1,25^{-19} * 1024] = 60$

$P7 = 1 + [(4) * 1,25^{-18} * 1024] = 74$

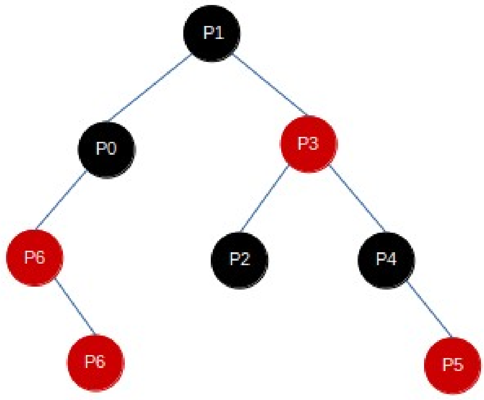
$P8 = 1 + [(4) * 1,25^{-17} * 1024] = 93$

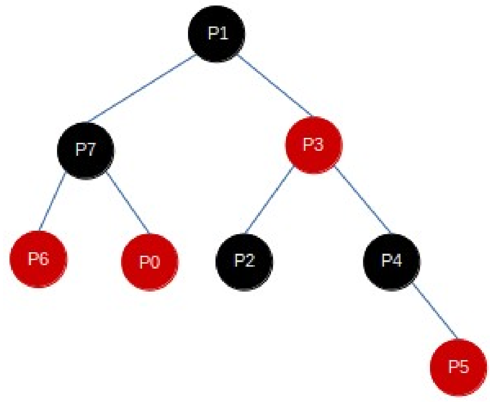
Εισαγωγή:

- Η P6 έχει μικρότερο vruntime από P0 άρα πάει αριστερά.

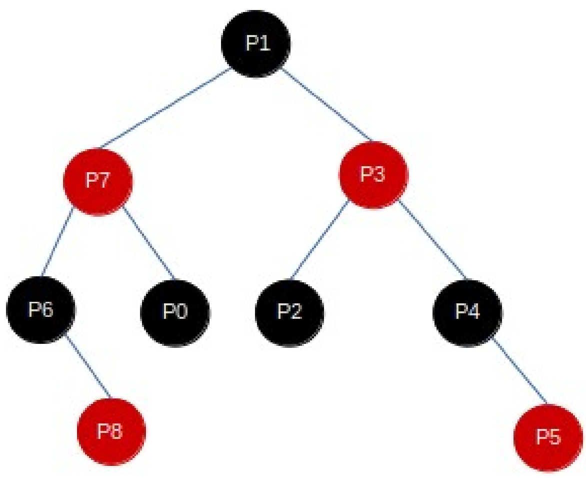


- Η P7 έχει μεγαλύτερο vruntime από P6 άρα πάει δεξιά. Ο θείος είναι black άρα περιστροφή.





- Τελικά εισάγεται και η P8 και έχουμε αλλαγές χρωμάτων.



P0 = 440 + [(14,9) * $1{,}25^{-10}$ *1024] = 2078

P1 =1343 + [(4,9) * $1{,}25^{-5}$ *1024] = 2987

P2 = 4097 + [(1,6) * $1{,}25^{0}$ *1024] = 5735.4

P3 = 5121 + [(1,2) * $1{,}25^{1}$ *1024] = 6657

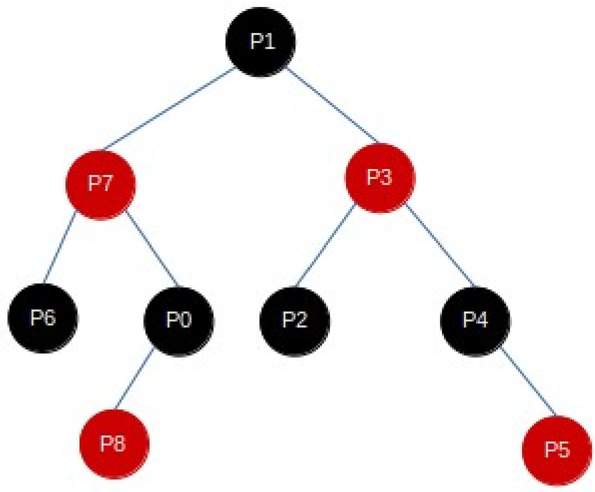
P4 = 10001 + [(0,6) * $1{,}25^{4}$ *1024] = 11501

P5 = 12497 + [(0,5) * $1{,}25^{5}$ *1024] = 14059.5

P6 = 60 + [(4) * $1{,}25^{-19}$ *1024] = 119

P7 = 74 + [(4) * $1{,}25^{-18}$ *1024] = 148

P8 = 93 + [(4) * $1{,}25^{-17}$ *1024] = 185



Άσκηση 7

Μία χρονική στιγμή, 7 διεργασίες Δ0-Δ6 έχουν vruntime 600, 200, 1200, 1000, 1400,1600 και 1800.

Α) Σχεδιάστε το RB tree την τρέχουσα στιγμή.

Β) Έστω ότι οι διεργασίες έχουν τιμές nice: 12, 13, 14, 15, 16, 17, και 18 αντίστοιχα. Να βρείτε τα νέα κβάντα τους αν TL = 28 και MG=4 και να δώσετε την κατάσταση του δέντρου μετά από την εκτέλεση.

ΛΥΣΗ

Α) Αρχικό RB Tree

Δ0 = 600

Δ1 = 200

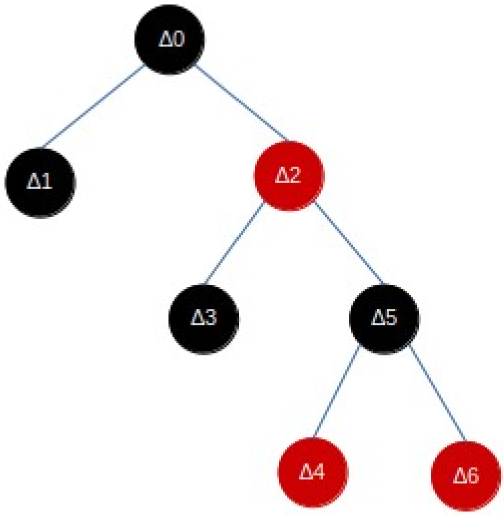
Δ2 = 1200

Δ3 = 1000

Δ4 = 1400

Δ5 = 1600

Δ6 = 1800



Β) Υπολογισμός νέων κβάντων

ΥΠΟΛΟΓΙΣΜΟΣ vruntime [$VR = VR + (t * w)$ και $w = 1{,}25^{nice} * 1024$]

Δ0 = 600 + [(4) * $1{,}25^{12}$ *1024] = 60204

Δ1 =200 + [(4) * $1{,}25^{13}$ *1024] = 74705

Δ2 = 1200 + [(4) * $1{,}25^{14}$ *1024] = 94332

Δ3 = 1000 + [(4) * $1{,}25^{15}$ *1024] = 117415

Δ4 = 1400 + [(4) * $1{,}25^{16}$ *1024] = 146919

Δ5 = 1600 + [(4) * $1{,}25^{17}$ *1024] = 183498

Δ6 = 1800 + [(4) * $1{,}25^{18}$ *1024] = 229173

Νέα κβάντα

(K=1024 / $1{,}25^{nice}$)

$K_{Δ0}$ = 1024 / $1{,}25^{12}$ = 70

$K_{Δ1}$ = 1024 / $1{,}25^{13}$ = 56

$K_{Δ2}$ = 1024 / $1{,}25^{14}$ = 45

$K_{Δ3}$ = 1024 / $1{,}25^{15}$ = 36

$K_{Δ4}$ = 1024 / $1{,}25^{16}$ = 28

$K_{Δ5}$ = 1024 / $1{,}25^{17}$ = 23

$K_{Δ6}$ = 1024 / $1{,}25^{17}$ = 18

ΣΥΝΟΛΟ Μ = 276

Για κάθε διεργασία δίνουμε χρόνο από **TL(K/M)**

Δ0 = (70 / 276) * 28 = 7,1

Δ1 = (56 / 276) * 28 = 5,6

Δ2 = (45 / 276) * 28 = 4,5

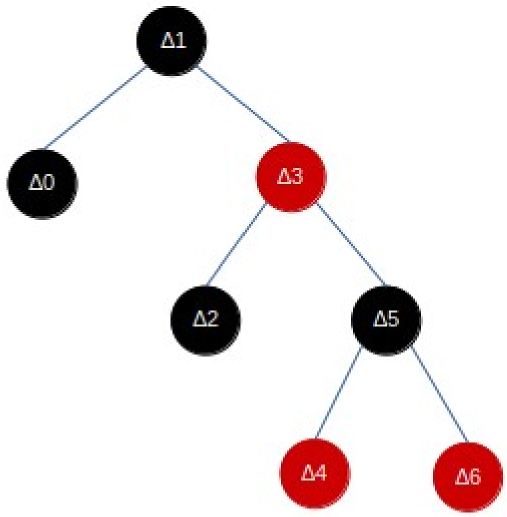
Δ3 = (36 / 276) * 28 = 3,6

Δ4 = (28 / 276) * 28 = 2,8

Δ5 = (23 / 276) * 28 = 2

Δ6 = (18 / 276) * 28 = 1,8

Οι θέσεις των Δ0, Δ1, Δ2, Δ3 θα αλλάξουν και το δέντρο θα γίνει όπως φαίνεται παρακάτω:



Άσκηση 8 **(Η άσκηση δεν αφορά άμεσα το μάθημα Λειτουργικά Συστήματα)**

Σχεδιάστε ένα RB-Tree για τις τιμές 3, 1, 5, 7, 6., 8, 9, και 10 που εισέρχονται με αυτή τη σειρά

ΛΥΣΗ