Process Scheduling Table

a)

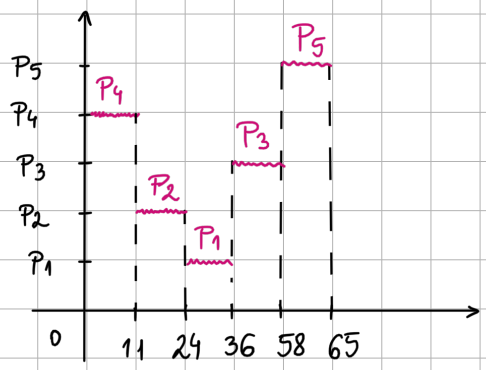
	Arrival	Burst	RR		SJF		SRTN		FCFS	
			Tw	TR	Tw	TR	Tw	TR	Tw	TR
P_1	5	12	36	48	13	25	23	35	19	31
P_2	1	13	37	50	29	42	29	42	10	23
P_3	5	22	38	60	38	60	38	60	31	53
P_4	0	11	37	48	0	11	7	18	0	11
P_5	7	7	33	40	4	11	3	10	51	58
Average values			36,2	49,2	16,8	29,8	20	33	22,2	35,2

FCFS

Gantt Chart

P_4	P_2	P_1	P_3	P_5

0 11 24 36 58 65



T_R = Completion time − Arrival time

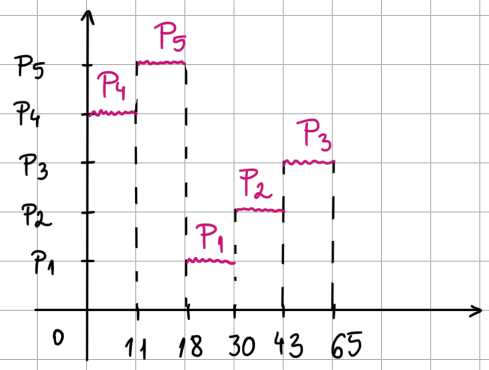
T_W = T_R − Burst time

SJF

Gantt Chart

P_4	P_5	P_1	P_2	P_3

0 11 18 30 43 65



Burst time average = $(12+13+22+11+7)/5 = 65/5 = 13$

$2 = 80\,\% \; 13 \simeq 10$

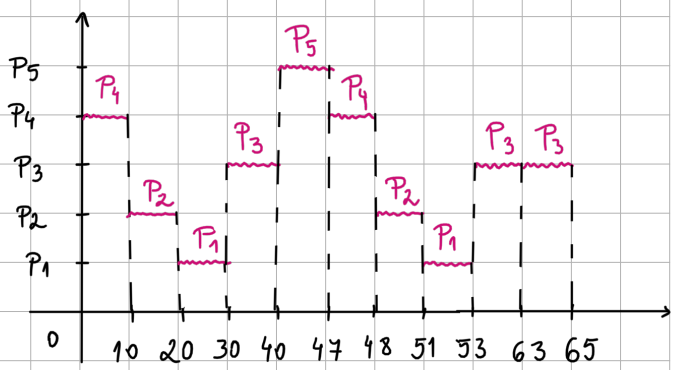
RR

Ready queue : $\{P_4-11, P_2-13, P_1-12, P_3-22, P_5-7, P_4-1, P_2-3, P_1-2, P_3-12, P_3-2\}$

Gantt Chart

P_4	P_2	P_1	P_3	P_5	P_4	P_2	P_1	P_3	P_3

0 10 20 30 40 47 48 51 53 63 65



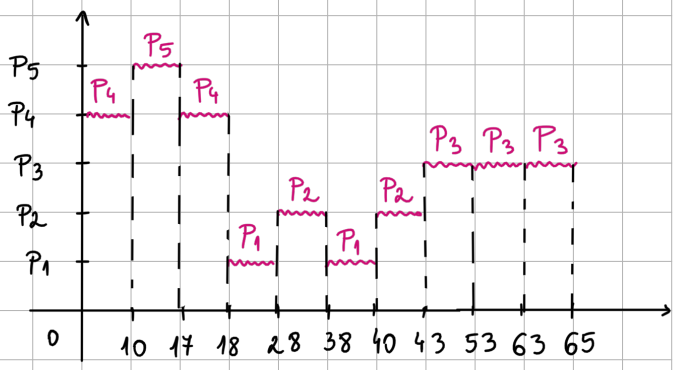
SRTN

Ready queue : { ~~P_4-11~~, ~~P_2-13~~, ~~P_1-12~~, ~~P_3-22~~, ~~P_5-7~~, ~~P_4-1~~, ~~P_1-2~~, ~~P_2-3~~, ~~P_3-12~~, ~~P_3-2~~ }

Gantt Chart

P_4	P_5	P_4	P_1	P_2	P_1	P_2	P_3	P_3	P_3

0 10 17 18 28 38 40 43 53 63 65



b) The algorithm that performed the best, considering the average values, is SJF in terms of waiting time and turn around time. The fact that the shortest process was prioritised, led to a shorter waiting time and turn around time.

c) FCFS → 4 context switches RR → 8 context switches
SJF → 4 context switches SRTN → 7 context switches

The number of context switches is the lowest in the non-preemptive algorithms (FCFS, SJF), but in their case, once begun, a process is taken to its end. Context switches occur for preemptive algorithms and in our case, for a q computed as 80% of the average burst time, SRTN has fewer context switches than RR.

d) If the value of q would have been 6 in the RR algorithm, which is smaller than the computed q of value 10, then the number of context switches would increase, as well as the waiting time and the turn around time, making the process interrupt more often. So a lower value for q decreases the performance of the algorithm.

for $q = 6$; Ready queue : { ~~P_4-11~~, ~~P_2-13~~, ~~P_1-12~~, ~~P_3-22~~, ~~P_4-5~~, ~~P_5-7~~, ~~P_2-7~~, ~~P_1-6~~, ~~P_3-16~~, ~~P_5-1~~, ~~P_2-1~~, ~~P_3-10~~, ~~P_3-4~~ }

1 2 3 4 5 6 7 8 9 10 11

P_4	P_2	P_1	P_3	P_4	P_5	P_2	P_1	P_3	P_5	P_6	P_3	P_3

0 6 12 18 24 29 35 41 47 53 54 55 61 65

We now have a total of 11 context switches, compared to the 8 context switches that we had when q was 10.

e) The process queue at moment of time 30 in the SRTF algorithm looks like this :
Ready queue : { ~~P_4-11~~, ~~P_2-13~~, ~~P_1-12~~, P_3-22, ~~P_5-7~~, ~~P_4-1~~, P_1-2 ... }
10 17 18 28 38 ⟹ Q : { P_3 P_1 }

f) In the SJF algorithm, besides the first process to execute (P_4), P_5 is the process that has the smallest response time.
Response time = Start time - Arrival time

$P_1 \rightarrow 18-5 = 13$ $P_3 \rightarrow 43-5 = 38$
$P_2 \rightarrow 30-1 = 29$ $P_4 \rightarrow 0-0 = 0$ (smallest)
$P_5 \rightarrow 11-7 = 4$ (next one)

Process Scheduling Table

a)

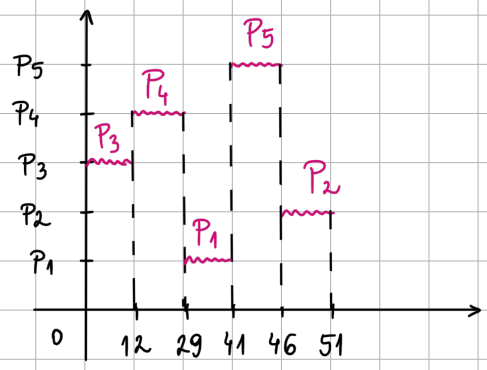
	Arrival	Burst	RR		SJF		SRTN		FCFS	
			T_W	T_R	T_W	T_R	T_W	T_R	T_W	T_R
P_1	2	12	36	48	20	32	28	40	27	39
P_2	5	5	24	29	12	17	12	17	41	46
P_3	0	12	26	38	0	12	5	17	0	12
P_4	0	17	34	51	34	51	34	51	12	29
P_5	2	5	22	27	10	15	6	11	39	44
Average values			28,4	38,6	23,8	34	17	27,2	15,2	25,4

FCFS

Gantt Chart

P_3	P_4	P_1	P_5	P_2

0 12 29 41 46 51



T_R = Completion time - Arrival time

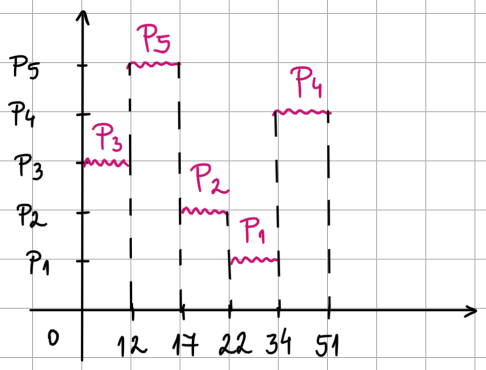
T_W = T_R - Burst time

SJF

Gantt Chart

P_3	P_5	P_2	P_1	P_4

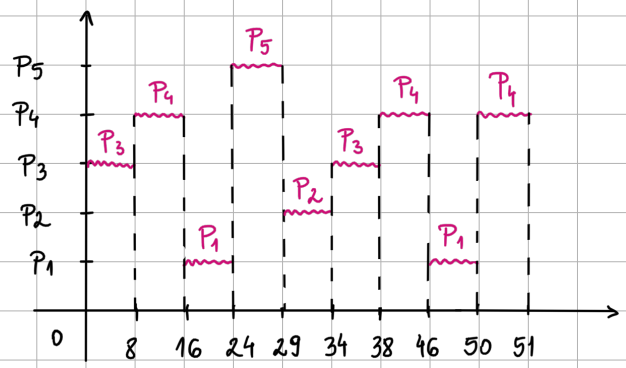
0 12 17 22 34 51



Average burst : $(12+5+12+17+5)/5 = 10,2$

$Q = 80\% \ 10,2 \simeq 8$

RR



Gantt Chart

P_3	P_4	P_1	P_5	P_2	P_3	P_4	P_1	P_4

0 8 16 24 29 34 38 46 50 51

Ready queue : { ~~P_3-12~~, ~~P_4-17~~, ~~P_1-12~~, ~~P_5-5~~, ~~P_2-5~~, ~~P_3-4~~, ~~P_4-9~~, ~~P_1-4~~, ~~P_4-1~~ }

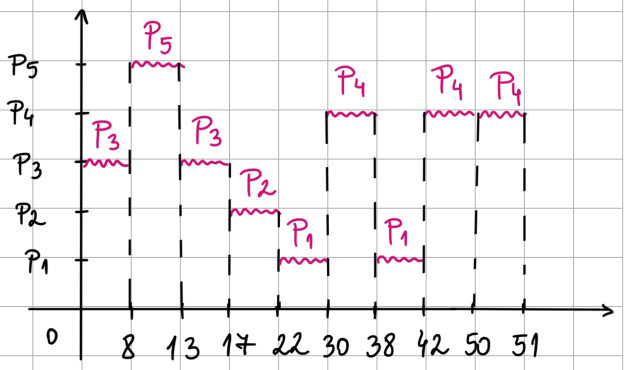
SRTN

Ready queue : { ~~$P_3 - 12$~~, ~~$P_4 - 17$~~, ~~$P_1 - 12$~~, ~~$P_5 - 5$~~, ~~$P_2 - 5$~~, ~~$P_3 - 4$~~, ~~$P_1 - 4$~~, ~~$P_4 - 9$~~, ~~$P_4 - 1$~~ }

Gantt Chart

P_3	P_5	P_3	P_2	P_1	P_4	P_1	P_4	P_4

0 8 13 17 22 30 38 42 50 51



b) The algorithm that performed the best, considering the average values, is FCFS in terms of waiting time and turn around time. The fact that the shortest process was prioritised, led to a shorter waiting time and turn around time.

c) FCFS → 4 context switches RR → 8 context switches
SJF → 4 context switches SRTN → 7 context switches

The number of context switches is the lowest in the non-preemptive algorithms (FCFS, SJF), but in their case, once begun, a process is taken to its end. Context switches occur for preemptive algorithms and in our case, for a q computed as 80% of the average burst time, SRTN has fewer context switches than RR.

d) If the value of q would have been 6 in the RR algorithm, which is smaller than the computed q of value 8, then the number of context switches would increase, as well as the waiting time and the turn around time, making the process interrupt more often, but since the values are so close, we are not able to spot the difference, since the algorithm performs the same.

for $q = 6$; Ready queue : { ~~$P_3 - 12$~~, ~~$P_4 - 17$~~, ~~$P_1 - 12$~~, ~~$P_5 - 5$~~, ~~$P_2 - 5$~~, ~~$P_3 - 6$~~, ~~$P_4 - 11$~~, ~~$P_1 - 6$~~, ~~$P_4 - 5$~~ }

1 2 3 4 5 6 7 8

P_3	P_4	P_1	P_5	P_2	P_3	P_4	P_1	P_4

0 6 12 18 23 28 34 40 46 51

We now have a total of 8 context switches, which is the same as when q was 8.

e) The process queue at moment of time 30 in the SRTF algorithm looks like this :
Ready queue : { ~~$P_3 - 12$~~, $P_4 - 17$, ~~$P_1 - 12$~~, ~~$P_5 - 5$~~, ~~$P_2 - 5$~~, ~~$P_3 - 4$~~,
8 13 17 22 30 ⟹ Q : { P_4 }

f) In the SJF algorithm, besides the first process to execute (P_3), P_5 is the process that has the smallest response time.
Response time = Start time - Arrival time

$P_1 \rightarrow 22 - 2 = 20$ $P_3 \rightarrow 0 - 0 = 0$ (smallest)
$P_2 \rightarrow 17 - 5 = 12$ $P_4 \rightarrow 34 - 0 = 34$
$P_5 \rightarrow 12 - 2 = 10$ (next one)

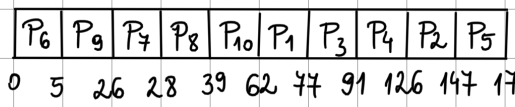
Process Scheduling Table

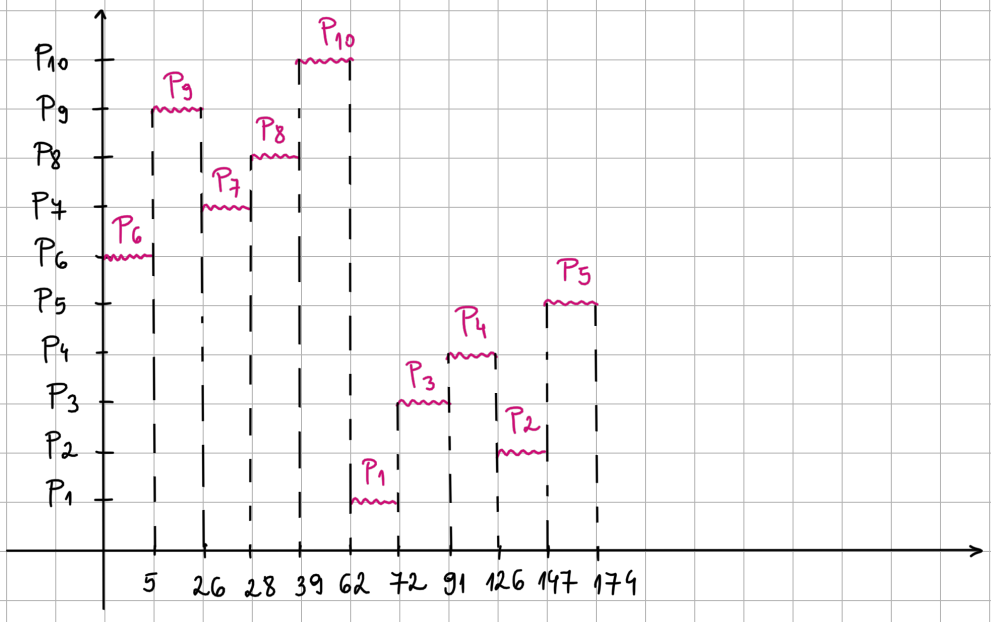
a)

			RR		SJF		SRTN		FCFS	
	Arrival	Burst	Tw	Tr	Tw	Tr	Tw	Tr	Tw	Tr
P_1	9	15	109	124	23	38	37	52	53	68
P_2	15	21	118	139	53	74	67	88	111	132
P_3	12	14	48	62	6	20	6	20	65	79
P_4	13	35	126	161	126	161	126	161	78	113
P_5	30	27	110	137	82	109	96	123	117	144
P_6	0	5	0	5	0	5	0	5	0	5
P_7	3	2	16	18	2	4	2	4	23	25
P_8	3	11	18	29	4	15	4	15	25	36
P_9	0	21	88	109	47	68	47	68	5	26
P_{10}	3	23	106	129	86	109	86	109	36	59
Average values			73,9	91,3	42,9	60,3	47,1	64,5	51,3	68,7

FCFS

Gantt Chart





f) In the SJF algorithm, besides the first process to execute (P_6), P_7 is the process that has the smallest response time.

Response time = Start time - Arrival time

$P_1 \rightarrow 32 - 9 = 23$

$P_2 \rightarrow 68 - 15 = 53$

$P_3 \rightarrow 18 - 12 = 6$

$P_4 \rightarrow 126 - 13 = 113$

$P_5 \rightarrow 112 - 30 = 82$

$P_6 \rightarrow 0 - 0 = 0$

$P_7 \rightarrow 5 - 3 = 2$

$P_8 \rightarrow 7 - 3 = 4$

$P_9 \rightarrow 46 - 0 = 46$

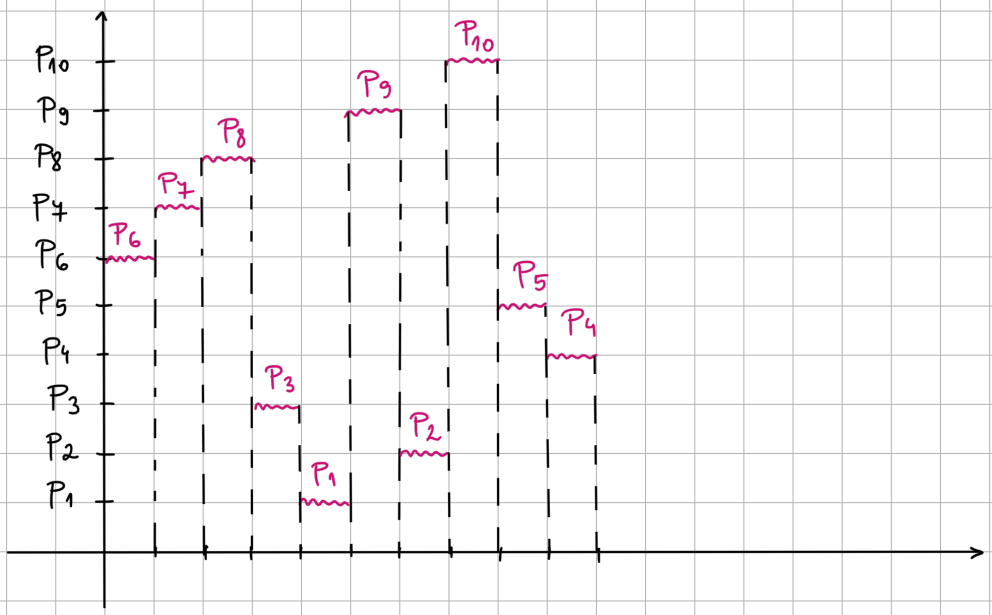
$P_{10} \rightarrow 82 - 3 = 79$

SJF

Gantt Chart

P_6	P_7	P_8	P_3	P_1	P_9	P_2	P_{10}	P_5	P_4

0 5 7 18 32 47 68 89 112 139 174



Burst time average : $(15+21+14+35+27+5+2+11+21+23)/10 = 17,4$

$2 = 80\,\%\ 17,4 \simeq 14$

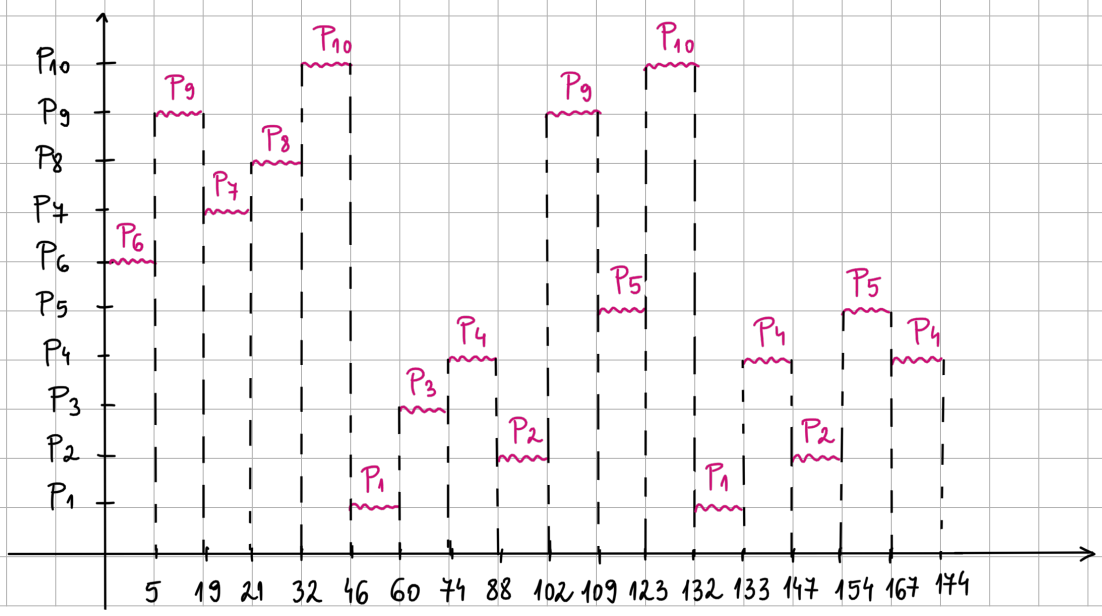
RR $(2 = 14)$

Gantt Chart

Ready queue : { ~~P_6-5~~, ~~P_9-21~~, ~~P_7-2~~, ~~P_8-11~~, ~~$P_{10}-23$~~, ~~P_1-15~~, ~~P_3-14~~, ~~P_4-35~~, ~~P_2-21~~, ~~P_9-7~~, ~~P_5-27~~, ~~$P_{10}-9$~~, ~~P_1-1~~, ~~P_4-21~~, ~~P_2-7~~, ~~P_5-13~~, ~~P_4-7~~ }

P_6	P_9	P_7	P_8	P_{10}	P_1	P_3	P_4	P_2	P_9	P_5	P_{10}	P_1	P_4	P_2	P_5	P_4

0 5 19 21 32 46 60 74 88 102 109 123 132 133 147 154 167 174



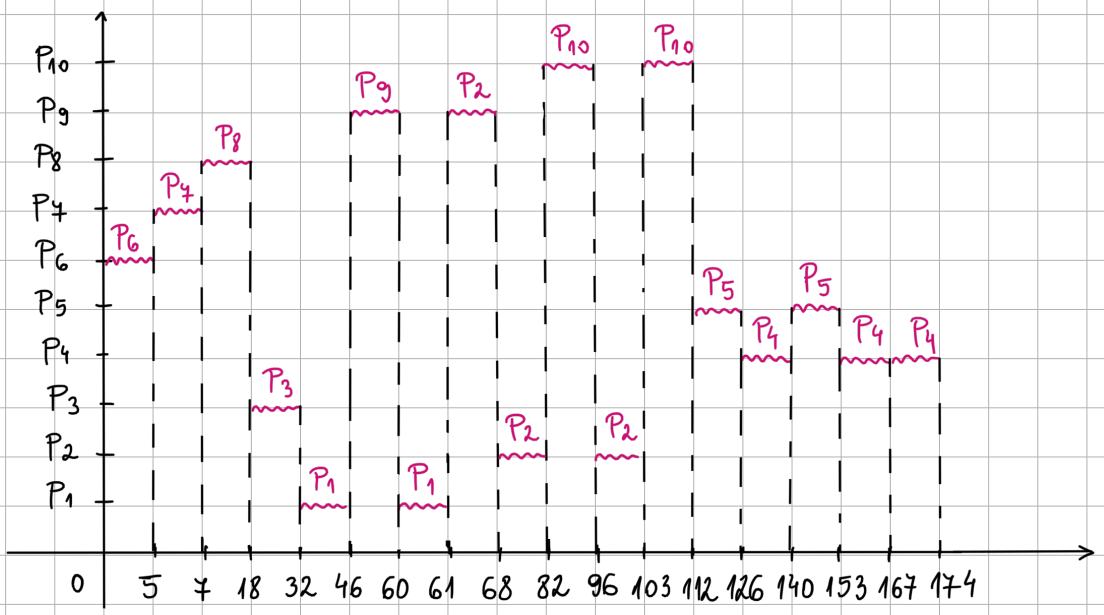
SRTN ($q = 14$)

Gantt Chart

Ready queue : {P_6-5, P_7-2, P_8-11, P_3-14, P_1-15, P_9-21, P_1-1, P_9-7, P_2-21, $P_{10}-23$, P_2-7, $P_{10}-9$, P_5-27, P_4-35, P_5-13, P_4-21, P_4-7}

P_6	P_7	P_8	P_3	P_1	P_9	P_1	P_9	P_2	P_{10}	P_2	P_{10}	P_5	P_4	P_5	P_4	P_4

0 5 7 18 32 46 60 61 68 82 96 103 112 126 140 153 167 174



b) The algorithms that performed the best, considering the average values, are SJF and SRTN. SJF seems to be performing better because of its non-preemptive nature which allows a low number of context switches.

c) FCFS → 9 context switches RR → 16 context switches
SJF → 9 context switches SRTN → 15 context switches

The number of context switches is the lowest in the non-preemptive algorithms (FCFS, SJF), but in their case, once begun, a process is taken to its end. Context switches occur for preemptive algorithms and in our case, for a q computed as 80% of the average burst time, SRTN has fewer context switches than RR.

d) By increasing the quantum value to 50, which is greater than the initially computed 14, the efficiency of the RR algorithm will increase as the number of context switches will decrease. In this particular case, increasing the quantum value to 50 will make the RR algorithm to work as FCFS.

e) The process queue at moment of time 30 in the SRTF algorithm looks like this :
Q : P_9 P_{10} P_1 P_4 P_2 P_5

Memory Management

a) 16 - bit 1, 5, 6, 6, 5, 5, 5, 4, 4, 2, 1, 6, 7, 0, 1, 3, 4, 1, 2

8,192 bytes - page size

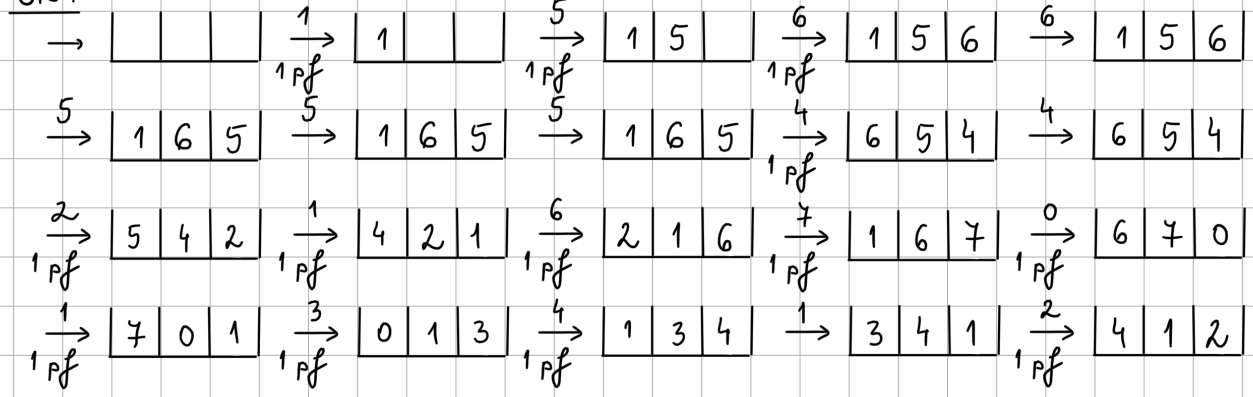
24 KB RAM memory $\Rightarrow \frac{24\ KB}{8,192\ b} = \frac{24 \cdot 2^{10}}{8 \cdot 2^{10}} = 3$ page frames

$\frac{2^{16}}{8 \cdot 2^{10}} = 8$ virtual pages

b) FIFO (when inserting a new element, if the element already exists we make sure it is on the last position resulting in no page faults, if the element doesn't exist, we remove the first element from the queue, we move the remaining elements up one position and we add the new element on the last position, resulting in a page fault)

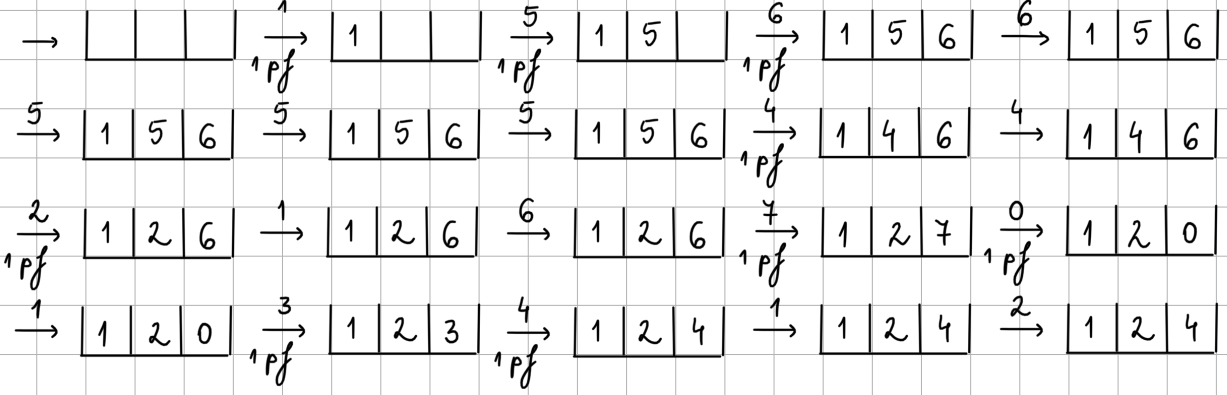
	1	5	6	6	5	5	5	4	4	2	1	6	7	0	1	3	4	1	2
F_1	1	1	1	1	1	1	1	6	6	5	4	2	1	6	7	0	1	3	4
F_2		5	5	5	6	6	6	5	5	4	2	1	6	7	0	1	3	4	1
F_3			6	6	5	5	5	4	4	2	1	6	7	0	1	3	4	1	2
p.f.	x	x	x	✓	✓	✓	✓	x	✓	x	x	x	x	x	x	x	x	✓	x

OR:



13 page faults at FIFO

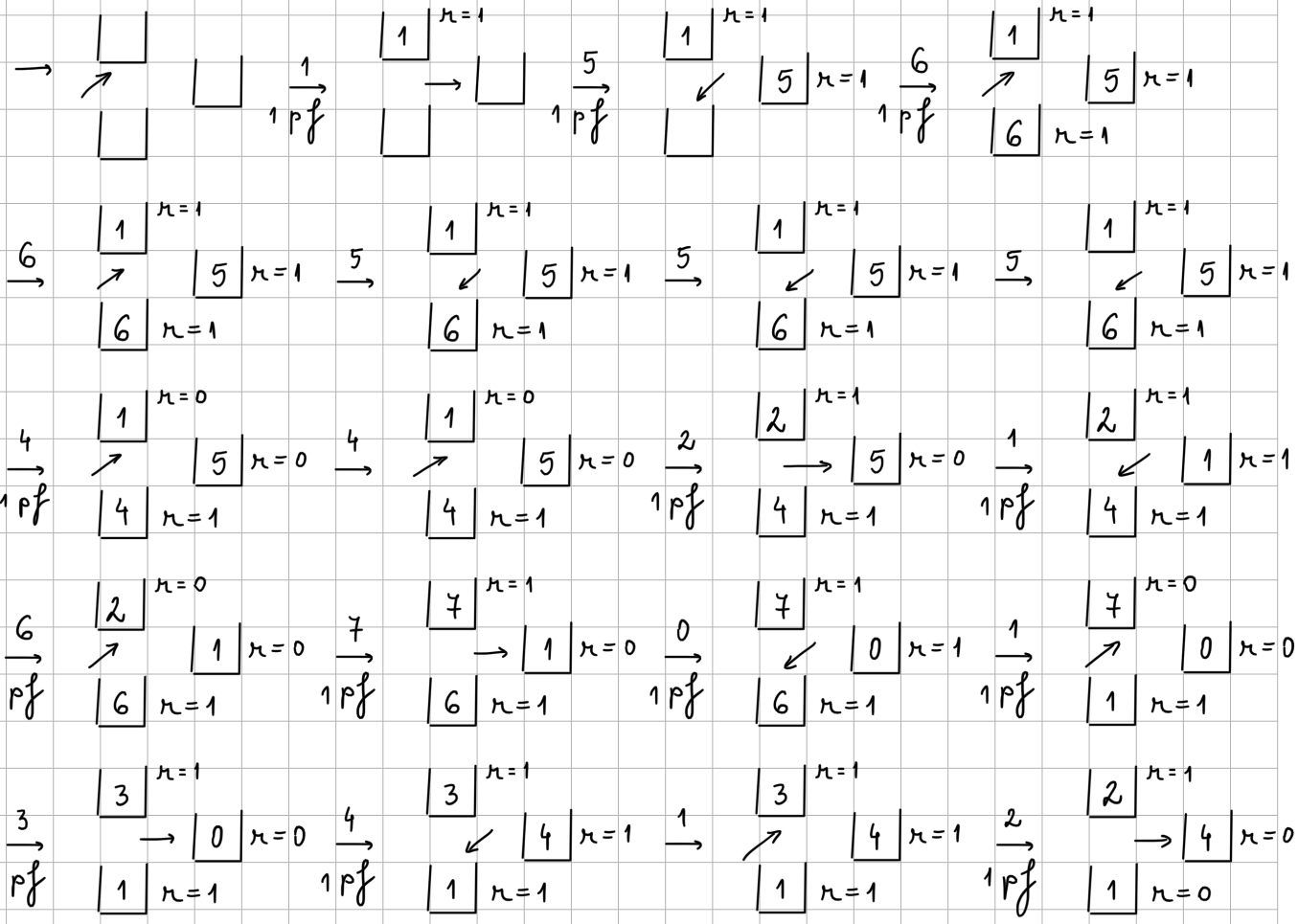
Optimal algorithm (when inserting an element that doesn't exist, we place that new element on the position of the element that will be used the last in the future, resulting in a page fault)



9 page faults at optimal algorithm

Clock 1, 5, 6, 6, 5, 5, 5, 4, 4, 2, 1, 6, 7, 0, 1, 3, 4, 1, 2

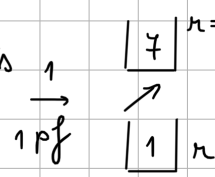
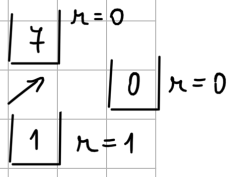
→ the arrow that points always moves to the next position
→ the r is set to 1 when inserting a new element
→ you cannot insert an element unless the r is 0
→ if the element already exists, now the arrow will point to the next position from where he finds that element
→ if the element doesn't exist, then if there is an element with r = 0, there is where we will place the new element, otherwise, if there is no element with r = 0, then we will set every element with r = 1 to r = 0 and we'll place the new element at the position where our arrow was previously pointing



13 page faults at Clock

c) At FIFO the moment 15 is $\xrightarrow[1\ pf]{1}$ | 7 | 0 | 1 |, so in memory we have | 7 | 0 | 1 |.

At optimal algorithm the moment 15 is $\xrightarrow{1}$ | 1 | 2 | 0 |, so in memory we have | 1 | 2 | 0 |.

At Clock the moment 15 is $\xrightarrow[1\ pf]{1}$  , so in memory we have 

Deadlocks

Process	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
MAX	35	60	90	95	90	45	30	30

a1)

Process	Has	Need	Max	
P_1	15	20	35	✓
P_2	5	55	60	✓
P_3	10	80	90	✓
P_4	0	95	95	✓
P_5	5	85	90	✓
P_6	0	45	45	✓
P_7	5	25	30	✓
P_8	5	25	30	✓

(initially the min val. from need)

→ $V_{min} = \cancel{20}\ \cancel{30}\ 50$

1. alloc. 20 → P_1 $\xrightarrow{P_1}$ 35 resources
   (35 − 25 = 10 remaining resources)
2. alloc. 25 → P_7 $\xrightarrow{P_7}$ 30 + 10 = 40 resources
   (40 − 25 = 15 remaining resources)
3. alloc. 25 → P_8 $\xrightarrow{P_8}$ 30 + 15 = 45 resources
   (45 − 45 = 0 remaining resources)
4. alloc. 45 → P_6 $\xrightarrow{P_6}$ 45 resources
   ⇒ deadlock. We need 10 more to execute process P_2 with 55 need

→ $V_{min} = 30$
(55 − 55 = 0 available resources)

5. alloc. 55 → P_2 $\xrightarrow{P_2}$ 60 resources
   ⇒ deadlock. We need 20 more to execute process 80

→ $V_{min} = 50$
(80 − 80 = 0 available resources)

6. alloc. 80 → P_3 $\xrightarrow{P_3}$ 90 resources
   (90 − 85 = 5 available resources)
7. alloc. 85 → P_5 $\xrightarrow{P_5}$ 90 + 5 = 95 resources
   (95 − 95 = 0 available resources)
8. alloc. 95 → P_4 $\xrightarrow{P_4}$ 95 resources ⇒ system is in safe state so $V_{min} = 50$

a2)

Process	Has	Need	Max	
P_1	25	10	35	✓
P_2	5	55	60	✓
P_3	10	80	90	✓
P_4	5	90	95	✓
P_5	5	85	90	✓
P_6	10	35	45	✓
P_7	15	15	30	✓
P_8	15	15	30	✓

Since all the processes executed and there are no deadlocks, then $V_{min} = 10$

→ $V_{min} = 10$

1. alloc. 10 → P_1 ⇒ 35 available resources
   (35 − 35 = 0 remaining resources)
2. alloc. 35 → P_6 ⇒ 45 available resources
   (45 − 15 = 30 remaining resources)
3. alloc. 15 → P_7 ⇒ 30 + 30 = 60 avail. r.
   (60 − 15 = 45 remaining resources)
4. alloc. 15 → P_8 ⇒ 30 + 45 = 75 avail. r.
   (75 − 55 = 20 remaining resources)
5. alloc. 55 → P_2 ⇒ 60 + 20 = 80 avail. r.
   (80 − 80 = 0 remaining resources)
6. alloc. 80 → P_3 ⇒ 90 avail res.
   (90 − 90 = 0 remaining resources)
7. alloc. 90 → P_4 ⇒ 95 avail. res.
   (95 − 85 = 10 remaining resources)
8. alloc. 85 → P_5 ⇒ 90 + 10 = 100 a.r.

b1)

Process	Has	Need	Max
P_1	15	20	35
P_2	5	55	60
P_3	10	80	90
P_4	0	95	95
P_5	5	85	90
P_6	0	45	45
P_7	5	25	30
P_8	5	25	30

$V_{min} = 50$

In the end at point a1 we had 91 available resources.
To trigger a deadlock we can set the request for P_4 to 96 and the resources won't be sufficient to solve.

b2)

Process	Has	Need	Max	
P_1	15	20	35	✓
P_2	5	55	60	✓
P_3	10	80	90	✓
P_4	0	95	95	✓
P_5	5	85	90	✓
P_6	0	45	45	✓
→ P_7	5	35	40	✓
→ P_8	5	35	40	✓

There are no deadlocks when we allocate an extra 10 resources to P_8 as well to P_7.

→ $V_{min} = 50$
(50 - 20 = 30 remaining resources)
1. alloc. 20 → P_1 $\xRightarrow{P_1}$ 35 + 30 = 65 a.r.
(65 - 35 = 30 remaining resources)
2. alloc. 35 → P_7 $\xRightarrow{P_7}$ 40 + 30 = 70 a.r.
(70 - 35 = 35 remaining resources)
3. alloc. 35 → P_8 $\xRightarrow{P_8}$ 40 + 35 = 75 a.r.
(75 - 55 = 20 remaining resources)
4. alloc. 55 → P_2 $\xRightarrow{P_2}$ 60 + 20 = 80 a.r.
(80 - 80 = 0 remaining resources)
5. alloc. 80 → P_3 $\xRightarrow{P_3}$ 90 a.r.
(90 - 85 = 5 remaining resources)
6. alloc. 85 → P_5 $\xRightarrow{P_5}$ 90 + 5 = 95 a.r.
(95 - 95 = 0 remaining resources)
7. alloc. 95 → P_4 $\xRightarrow{P_4}$ 95 a.r
(95 - 45 = 50 remaining resources)
8. alloc. 45 → P_6 $\xRightarrow{P_6}$ 50 + 45 = 95 a.r.