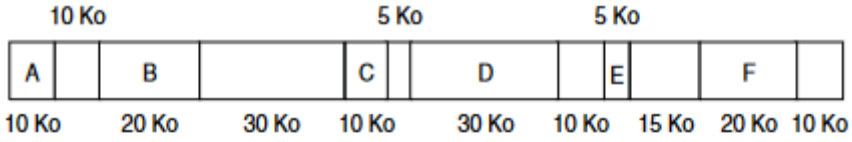
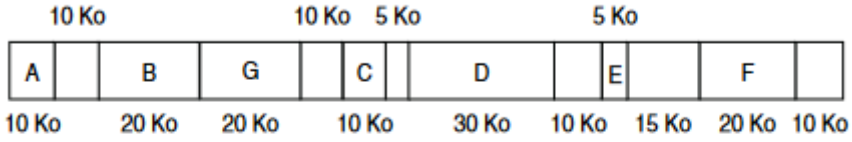
Exercice 1

1. Stratégie First Fit

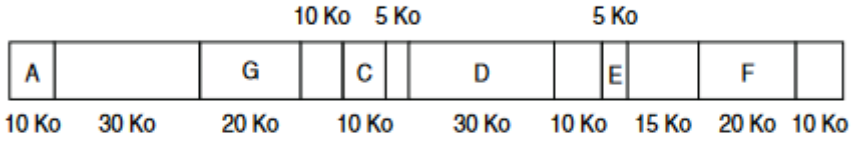
État Initial



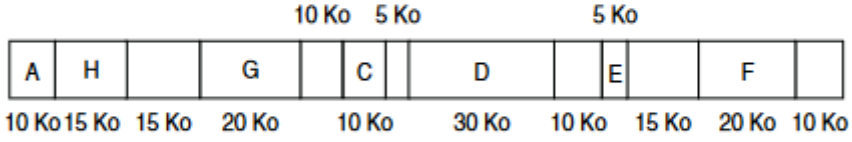
Arrivée du programme G, taille 20 Ko



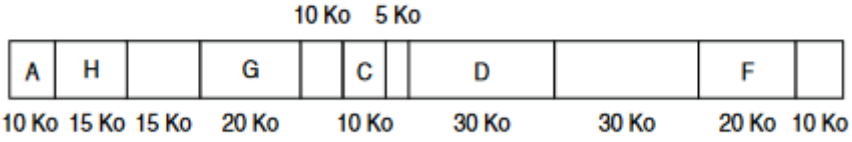
Départ du programme B



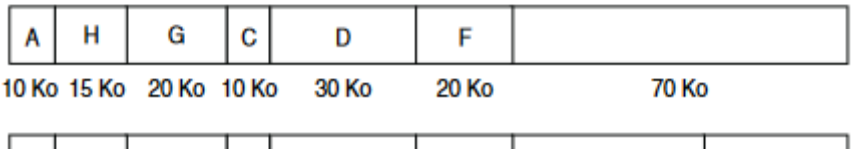
Arrivée du programme H, taille 15 Ko

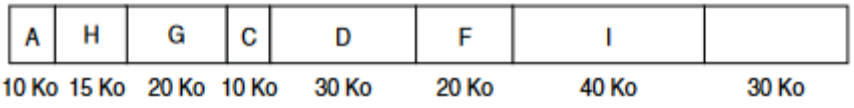


Départ du programme E



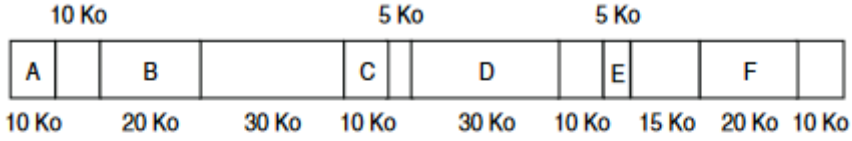
Arrivée du programme I, taille 40 Ko. L'état de la mémoire est fragmentéee. Il faut compacter :



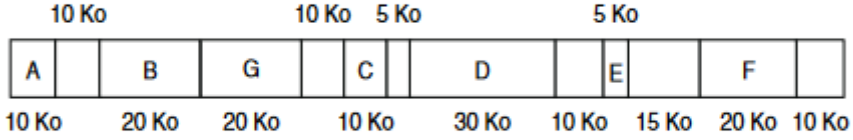


2- Stratégie Best Fit

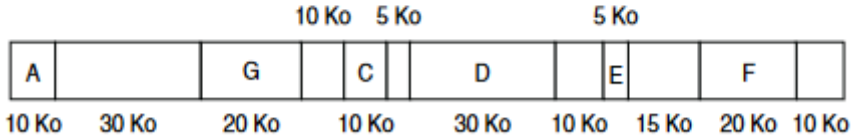
État Initial



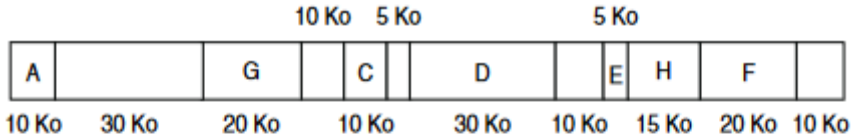
Arrivée du programme G, taille 20 Ko



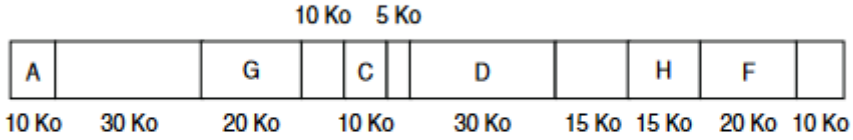
Départ du programme B



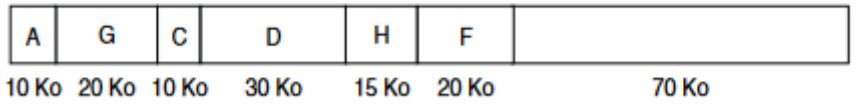
Arrivée du programme H, taille 15 Ko



Départ du programme E

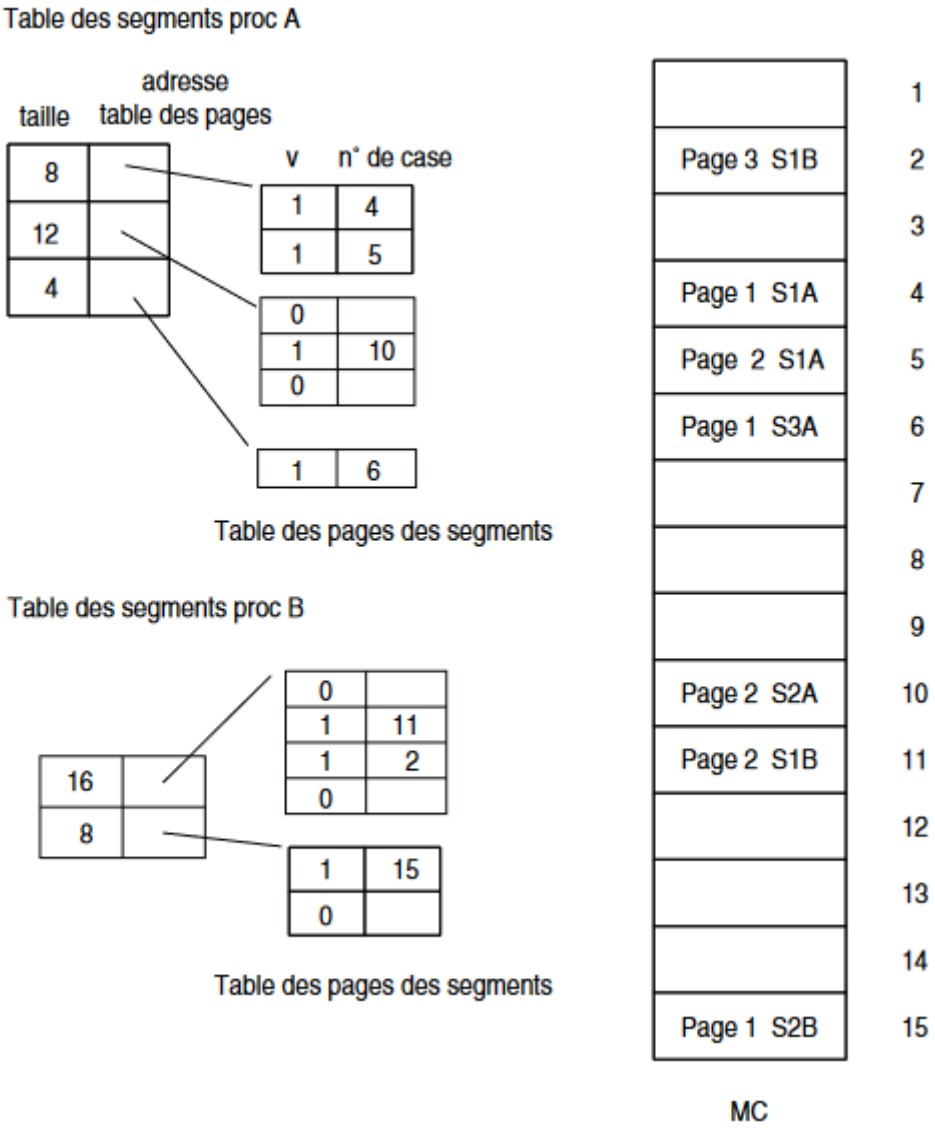


Arrivée du programme I, taille 40 Ko. L'état de la mémoire est fragmentéee. Il faut compacter :



Exercice 2

1- Tables des pages et tables des segments pour les processus A et B.



2- . adresse logique <S1A, page 1, 12>. = adresse réelle <12 Ko, 12> = 12 Ko + 12 = 12 300.

3- . adresse logique <S2B, page 2, 10>. La page 2 du segment 2 du processus B n'est pas en mémoire centrale. Il se produit un défaut de page. Nous pouvons supposer que la page manquante est chargée dans la première case libre, c'est-à-dire la case 1. Il s'ensuit alors que l'adresse physique correspondante est 10.

4- 4 098 pour le processus A : adresse virtuelle <S1, page 2, déplacement 2> = adresse réelle <case 5, déplacement 2> = octet 16 386

12 292 pour le processus A : adresse virtuelle <S2, page 2, déplacement 4> = adresse réelle <case 10, déplacement 4> = octet 331 780.

8 212 pour le processus B : adresse virtuelle <S1, page 3, déplacement 20> = adresse réelle <case 2, déplacement 20> = octet 4 116

Exercice 3

- le segment : S1
- le numéro de page dans le segment : 3
- le déplacement dans la page20
- le numéro de case 0
- le déplacement dans la case 20
- l'adresse physique (en décimal et en binaire) 4

Exercice 4

1. Optimal :

Page appelée	0-L	1-E	2-L	3-L	4-E	1-E	2-L	4-L	0-E	1-L
Cadre 1	0	0	0	3	4	4	4	4	0	0
Cadre 2		1	1	1	1	1	1	1	1	1
Cadre 3			2	2	2	2	2	2	2	2
Défauts de page	x	x	x	x	x	x			x	

6/10

2. LRU

Page appelée	0-L	1-E	2-L	3-L	4-E	1-E	2-L	4-L	0-E	1-L
Cadre 1	0	0	0	3	3	3	2	2	2	1
Cadre 2		1	1	1	4	4	4	4	4	4
Cadre 3			2	2	2	1	1	1	0	0
Défauts de page	x	x	x	x	x	x	x		x	x

9/10

3. Seconde chance

Page appelée	0-L	1-E	2-L	3-L	4-E	1-E	2-L	4-L	0-E	1-L
Cadre 1	0	0	0	3	3	3	2	2	2	2
Cadre 2		1	1	1	4	4	4	4	4	1
Cadre 3			2	2	2	1	1	1	0	0
Défauts de page	x	x	x	x	x	x	x		x	x
R de page 0	1	1	1	-	-	-	-	-	1	1
R de page 1		1	1	0	-	1	0	0	-	1
R de page 2			1	0	0	-	1	1	1	0
R de page 3				1	1	1	-	-	-	-
R de page 4					1	1	0	1	0	-

9/10

4. NRU : Nous supposons que toutes les 4 instructions, les bits sont remis à zéro.

Page appelée	0-L		1-E		2-L		3-L		4-E		1-E		2-L		4-L		0-E		1-L	
Cadre 1	0		0		0		3		4		4		4		4		4		0	
Cadre 2			1		1		1		1		1		1		1		1		1	
Cadre 3					2		2		2		2		2		2		0		4	
Défauts de page	x		x		x		x		x								x			
M et R de cadre 1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0
M et R de cadre 2			1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1
M et R de cadre 3					0	1	0	1	0	0	0	0	0	1	0	1	1	1	1	1

6/10

1ers cas seulement R est mis à 0 après 4 instructions

Exercice 7

Question 1 :

Taille d'une page = 200 octets. Taille d'une instruction = 4 octets

Pour déterminer le nombre de pages nécessaires au programme, il faut compter le nombre d'instructions du programme.

Le corps du programme comporte les instructions suivantes :

Programme initial	Programme équivalent	N°Instruction

Main( )
{
int A[20][20];
int i,j;
int B[20][20];
for (i=0; i<20; i+5)
for (j=0; j<20; j+19)
B[i][j] =A[i][j];
}
	Début
	i=0;
	e1 : si i ≤ 19
	alors j=0;
	e2 : si j ≤ 19
	alors B[i, j]=A[i,j];
	j=j+19;
	goto e2;
	fsi;
	i=i+5;
	goto e1;
	Fsi;
	Fin
		
		Inst 1
		Inst 2
		Inst 3
		Inst 4
		Inst 5
		Inst 6
		Inst 7
		Inst 8
		Inst 9
		Inst10
		Inst11

Nombre de pages nécessaires

Le programme est constitué du code et des données :

1. Déterminer la taille de la partie du code :

Taille code = Nombre d'instructions * Taille d'une instruction
= 11* 4 = 44 octets ⇒ une page suffit pour le code.

⇨ Comme une page est suffisante pour le code, il sera stocké dans la page P0.

2. Déterminer la taille de la partie données

Remarque : Dans cet exercice, nous supposons que les variables i et j sont stockées avec les élements des deux matrices A et B.

Selon la partie déclaration du programme, nous avons la déclaration de la matrice A, ensuite i et j et en dernier la matrice B :

- **Matrice A**: A=20*20 entiers= 400 entiers ➔ 400*2 = 800 octets ➔ **A occupe une taille de 4 pages.**
- **variable i** : i = 1 entier = 1mot ➔ 2 octets.
- **variable j** : j = 1 entier = 1mot ➔ 2 octets.
- **Matrice B** : B = 20x20 entiers = 400 entiers = 400x2 octets = 800 octets ➔ **B occupe 4 pages**

⇒ Total données : 800+4+800 =1604 octets.

Taille d'une Page = 200 octets ⇒ Nombre de pages nécessaires **aux données** = 1604/200= **9 pages**, numérotées de de $\mathbf{P_1}$ à $\mathbf{P_9}$.

Le programme sera donc sur 10 pages (P0 à P9)

Organisation des pages du programme

Comme l'ordre des déclarations est A, i, j, puis B, le stockage des données va suivre cet ordre. D'où, la représentation des pages est comme suit :

$\mathbf{P_0}$: Code (programme)

$\mathbf{P_1}$: A [0, 0], ..., A[0, 19]
A [1, 0], …, A[1, 19]
A [2, 0]
A [3, 0]
A [4, 0], ….., A[4, 19]

$\mathbf{P_2}$: A [5, 0], ….., A[5, 19]
…..

A [9, 0], …..., A [9, 19]

$\mathbf{P_5}$: i, j,
B[0, 0] …..., B [0, 19]
B[1, 0],….., B [1, 19]

B[4, 0],….., B [4, 17]

$\mathbf{P_6}$: B[4, 18],….., B [4, 19]
B[5, 0] ,. …., B[5, 19]
…….
B[9, 0], …..., B[9, 17]

P_3 : A [10, 0],…., A[10, 19]
…….
A [14, 0], …, A[14, 19]

P_4 : A [15, 0], …., A[15, 19]
……..
A [19, 0], …., A[19, 19]

P_7 : B[9, 18],….., B [9, 19]
B[10, 0],….., B [10, 19]
……
B[14, 0],….., B [14, 17]

P_8 : B[14, 18],….., B [14, 19]
B[15, 0], …..., B [15, 19]
……
B[19, 0],…... , B[19, 17]

P_9 : B[19, 18] , B [19, 19]

Question 2 :

Exécution détaillée du programme (références mémoires générées)

i=0	P0P5	//P0 contient l'instruction et P5 contient la variable i Cond(i)
	P0P5	//P0 contient l'instruction (i<=19) et P5 contient la variable i
j=0	P0P5	//P0 contient l'instruction et P5 contient la variable j
Cond(j)	P0P5	//P0 contient l'instruction (j<=19) et P5 contient la variable j
B[0,0] = A[0,0]	P0P1P5	//P0 contient l'instruction (affectation), P1 contient l'élément A[0,0] et P5 contient les variables i et j et l'élément B[0,0]
j= j+19	P0P5	
cond(j)	P0P5	
B[0,19] = A[0,19]	P0P1P5	
j= j+19	P0P5	
Cond(j)	P0P5	
i= i+4 (i=4)	P0P5	
Cond(i)	P0P5	
j=0	P0P5	
Cond(j)	P0P5	
B[4,0] = A[4,0]	P0P1P5	
j = j+19	P0P5	
cond(j)	P0P5	
B[4,19] = A[4,19]	P0P1P6	
j = j+19	P0P5	
Cond(j)	P0P5	
i= i+4 (i=8)	P0P5	
Cond(i)	P0P5	
j=0	P0P5	
Cond(j)	P0P5	
B[8,0] = A[8,0]	P0P2P6	
j = j+19	P0P5	
Cond(j)	P0P5	
B[8,19] = A[8,19]	P0P2P6	
j = j+19	P0P5	
Cond(j)	P0P5	
i= i+4 (i=12)	P0P5	
Cond(i)	P0P5	
j=0	P0P5	
Cond(j)	P0P5	
B[12,0] = A[12,0]	P0P3P7	
j = j+19	P0P5	

Matrice A
A[0,0]…A[4,19] page P1
A[5,0]…A[9,19] page P2
A[10,0]…A[14,19] page P3
A[15,0]…A[19,19] page P4

Matrice B
B[0,0]…B[4,17] page P5
B[4,18]…B[9,17] page P6
B[9,18]…B[14,17] page P7
B[14,18]…B[19,17] page P8
B[19,18], B[19,19] page P9

Cond(j)	P0P5
B[12,19] = A[12,19]	P0P3P7
j = j+19	P0P5
Cond(j)	P0P5
i= i+4 (i=16)	P0P5
Cond(i)	P0P5
j=0	P0P5
Cond(j)	P0P5
B[16,0] = A[16,0]	P0P4P8
J= j+19	P0P5
Cond(j)	P0P5
B[16,19] = A[16,19]	P0P4P8
J= j+19	P0P5
Cond(j)	P0P5
i= i+4 (i=20)	P0P5
Cond(i)	P0P5

(i>19) donc "Arrêt du programme"

Question 3 :

La chaine de référence

P0P5 P0P5 P0P5 P0P5 P0P1P5 P0P5P0P5 P0P1P5 P0P5 P0P5 P0P5 P0P5 P0P5 P0P5 P0P1P5P0P5 P0P5 P0P1P6 P0P5 P0P5 P0P5 P0P5 P0P5 P0P5 P0P2P6 P0P5 P0P5 P0P2P6 P0P5P0P5 P0P5 P0P5 P0P5 P0P5 P0P3P7 P0P5 P0P5 P0P3P7 P0P5 P0P5 P0P5 P0P5 P0P5 P0P5 P0P4P8 P0P5 P0P5 P0P4P8 P0P5 P0P5 P0P5 P0P5.

La chaine réduite est :

P0P5 P0P1P5 P0P5 P0P1P5 P0P5 P0P1P5 P0P5 P0P1P6P0P5 P0P2P6P0P5 P0P2P6 P0P5 P0P3P7 P0P5P0P3P7 P0P5 P0P4P8 P0P5 P0P4P8 P0P5.

On peut encore réduire la chaine en éliminant les séquences répétitives **(sachant qu'on a 3 frames libres au minimum)**

La chaine de référence compactée devient :

P0P5 P0P1P5 P0P5 P0P1P6 P0P5 P0P2P6 P0P5 P0P3P7 P0P5 P0P4P8 P0P5

d. L'algorithme de remplacement LRU avec 3 blocs

	P0	P5	P0	P1	P5	P0	P5	P0	P1	P6	P0	P5	P0	P2	P6	P0	P5	P0	P3	P7
F1	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0
F2		P5	P5	P5	P5	P5	P5	P5	P5	P6	P6	P6	P6	P2	P2	P2	P5	P5	P5	P7
F3				P1	P1	P1	P1	P1	P1	P1	P1	P5	P5	P5	P6	P6	P6	P6	P3	P3
Dfpage	D	D		D						D		D		D	D		D		D	D

	P0	P5	P0	P4	P8	P0	P5
F1	P0	P0	P0	P0	P0	P0	P0
F2	P7	P7	P7	P4	P4	P4	P5
F3	P3	P5	P5	P5	P8	P8	P8
Df_page		D		D	D		D

Taux de Défaut de pages = 14/27

Question 4 :

Autre algorithme (LRU avec bit de choix)

Nouvelle stratégie : consiste à combiner LRU avec un **bit de choix** associé à chaque page. Ce bit est initialement à 0 pour toutes les pages. Dès qu'une page est sélectionnée comme page victime, son bit de choix augmente de 1. L'algorithme choisit alors comme page victime la page la moins récemment utilisée ayant la plus faible valeur du bit de choix.

L'algorithme de remplacement ***LRU avec bit* de choix (**MC sur 3 blocs)

	P0	P5	P0	P1	P5	P0	P5	P0	P1	P6	P0	P5	P0	P2	P6	P0	P5	P0	P3	P7
F1	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P6	P6	P6	P6	P3	P7
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
F2		P5	P5	P5	P5	P5	P5	P5	P5	P6	P6	P6	P6	P2	P2	P0	P0	P0	P0	P0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
F3				P1	P1	P1	P1	P1	P1	P1	P1	P5	P5	P5	P5	P5	P5	P5	P5	P5
				0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Dfpage	D	D		D						D		D		D	D	D			D	D

	P0	P5	P0	P4	P8	P0	P5
F1	P7	P7	P7	P4	P8	P8	P8
	0	0	0	0	0	0	0
F2	P0	P0	P0	P0	P0	P0	P0
	1	1	1	1	1	1	1
F3	P5	P5	P5	P5	P5	P5	P5
	1	1	1	1	1	1	1
Df_page				D	D		

Taux de Défaut de pages = 12/27

Nous constatons qu'il y a une amélioration par rapport à l'algorithme précédent car une page qui est déjà victime rentre de nouveau en mémoire avec une valeur de bit *incrémentée de 1* (évitera de la faire sortir une autre fois aussitôt). Donc, il s'agit d'augmenter les chances des pages (victimes) les plus fréquemment référencées pour rester le plus possible en mémoire centrale.