Exercice 3.

$A(A_1, A_2, A_3)$	$B(B_1, B_2)$
200 tuples / page	40 tuples / page
Instance 2000 tuples = 10 pages	Instance 120 400 tuples = 3010 pages
600 avec $A_1 = 4$ 3 pages	80000 avec $B_1 = 17$ 2000 pages
600 avec $A_1 = 8$ 3 pages	40 400 avec $B_1 = 10$ 1010 pages
800 avec $A_1 = 12$ 4 pages	

$A \bowtie_{A_1 = B_1} B$

3.1) ~~Avec A externe~~ Simple NL join, coût :

$C = M + M * pR * N$

- M : nb pages rel externe
- pR : nb tuples par page rel externe
- N : nb pages rel interne

Avec A externe :

$C = 10 + 10 * 200 * 3010 = 6020010$

Avec B externe :

$C = 3010 + 3010 * 40 * 10 = 1207010$

Meilleur choix : B externe

(En général, on choisit plutôt la rel qui a le moins de tuples / page)

3.2) 6 buffers, hash join avec "A "externe".

1. Partitionnement : 5 buffers disponibles.

$h(x) = x \bmod 4 \Rightarrow$ 4 valeurs possibles $\Rightarrow$ 4 partitions ($4 \leq 5$)

$h(x) = x \bmod 3 \Rightarrow$ 3 valeurs possibles $\Rightarrow$ 3 partitions ($3 \leq 5$)

Les deux fonctions sont donc compatibles avec le nb. de buffers disponibles. Le coût du partitionnement est indépendant de la fonction de hash utilisée :

$C = 2 * (M + N) = 2 * (10 + 3010) = 6040$

- M : nb pages A
- N : nb pages B

On peut même omettre le calcul de ce coût car il ne différencie pas les deux fonctions.

2. "Recroisement"

a. Avec $h(x) = x \bmod 4$

Relation A

Taille partition "0" = 10 pages ($h(x) = 0$)
car tous les tuples de A "tombent dans la partition "0" :
$h(4) = 0$
$h(8) = 0$
$h(12) = 0$

Taille partition "1" = 0 pages
Taille partition "2" = 0 pages
Taille partition "3" = 0 pages

Relation B

Taille partition "0" = 0 pages

Taille partition "1" = 2000 pages ($B_1 = 17$)
Taille partition "2" = 1010 pages ($B_1 = 10$)
Taille partition "3" = 0 pages

Coût ("0" vs "0") = 0 ou 10 (les deux réponses sont acceptées)
Un algorithme de jointure bien implémenté "voit" en effet que la partition "0" de B est de taille 0, et ne procède même pas au "recroisement" de ces deux partitions.
Toutefois, on peut supposer que l'algorithme va "charger" la partition "0" de A, avant de "se rendre compte" qu'il faut que la partition correspondante de B est de taille 0. Dans ce deuxième cas, on charge les 10 pages de la partition "0" de A, donc le coût devient 10.
Le recroisement des autres partitions donne un coût 0.

b. Avec $h(x) = x \bmod 3$

Relation A

Taille partition "0" = 4 pages ($A_1 = 12$)
Taille partition "1" = 3 pages ($A_2 = 4$)
Taille partition "2" = 3 pages ($A_2 = 8$)

Relation B

Taille partition "0" = 0 pages
Taille partition "1" = 1010 pages ($B_1 = 10$)
Taille partition "2" = 2000 pages ($B_1 = 17$)

Pour calculer le coût, nous pouvons faire le "raccourci" suivant :

Comme on a 6 buffers, donc 4 buffers disponibles pour le chargement des partitions "externes" lors du recroisement", toutes les partitions externes peuvent être chargées "en une seule fois", donc le coût total est

$C = M + N = 10 + 3010 = \underline{3020}$

Notez que ce calcul revient à supposer un algo de join qui ne regarde pas si la 2ème partition est de taille 0 avant de charger la partition externe ! Si on fait en plus la supposition que l'algorithme va éviter tout chargement si la partition "interne" est de taille 0, alors on a :

Coût ("0" vs "0") = 0 (autrement, c'était 4)
Coût ("1" vs "1") = 3 + 1010 = 1013
Coût ("2" vs "2") = 3 + 2000 = 2003

Donc le coût total $C = 2003 + 1013 = \underline{3016}$

Résumé :

Coût partitionnement = 6040 pour les deux fonctions
Coût recroisement x mod 4 = 0 ou 10
Coût recroisement x mod 3 = 3016 ou 3020

À préférer donc : $h(x) = x \bmod 4$