Questionnaire TP AOD 2023-2024 à compléter et rendre sur teide

Binôme (NOM_1 $Prénom_1$ – NOM_2 $Prénom_2$) : .....

1 Préambule 1 point

. Le programme récursif avec mémoisation fourni alloue une mémoire de taille $N.M$. Il génère une erreur d'exécution sur le test 5 (c-dessous) . Pourquoi ?
Réponse: A COMPLETER

```
distanceEdition-recmemo    GCA_024498555.1_ASM2449855v1_genomic.fna 77328790 20236404   \
                           GCF_000001735.4_TAIR10.1_genomic.fna 30808129 19944517
```

Important. Dans toute la suite, on demande des programmes qui allouent un espace mémoire $O(N + M)$.

2 Programme itératif en espace mémoire $O(N + M)$ (5 points)

Expliquer très brièvement (2 à 5 lignes max) le principe de votre code, la mémoire utilisée, le sens de parcours des tableaux.

Analyse du coût théorique de ce programme en fonction de N et M en notation $\Theta(...)$

1. place mémoire allouée (ne pas compter les 2 séquences X et Y en mémoire via `mmap`) :
2. travail (nombre d'opérations) :
3. nombre de défauts de cache obligatoires (sur modèle CO, y compris sur X et Y):
4. nombre de défauts de cache si $Z \ll \min(N, M)$:

3 Programme cache aware (3 points)

Expliquer très brièvement (2 à 5 lignes max) le principe de votre code, la mémoire utilisée, le sens de parcours des tableaux.

Analyse du coût théorique de ce programme en fonction de N et M en notation $\Theta(...)$)

1. place mémoire (ne pas compter les 2 séquences initiales X et Y en mémoire via `mmap`) :
2. travail (nombre d'opérations) :
3. nombre de défauts de cache obligatoires (sur modèle CO, y compris sur X et Y):
4. nombre de défauts de cache si $Z \ll \min(N, M)$:

4 Programme cache oblivious (3 points)

Expliquer très brièvement (2 à 5 lignes max) le principe de votre code, la mémoire utilisée, le sens de parcours des tableaux.

Analyse du coût théorique de ce programme en fonction de N et M en notation $\Theta(...)$)

1. place mémoire (ne pas compter les 2 séquences initiales X et Y en mémoire via `mmap`) :
2. travail (nombre d'opérations) :
3. nombre de défauts de cache obligatoires (sur modèle CO, y compris sur X et Y):
4. nombre de défauts de cache si $Z \ll \min(N, M)$:

5 Réglage du seuil d'arrêt récursif du programme cache oblivious (1 point)

Comment faites-vous sur une machine donnée pour choisir ce seuil d'arrêt? Quelle valeur avez vous choisi pour les PC de l'Ensimag? (2 à 3 lignes)

6 Expérimentation (7 points)

Description de la machine d'expérimentation:
Processeur: A PRECISER – Mémoire: A PRECISER – Système: A PRECISER

6.1 (3 points) Avec `valgrind --tool =cachegrind --D1=4096,4,64`

```
distanceEdition ba52_recent_omicron.fasta 153 N wuhan_hu_1.fasta 116 M
```

en prenant pour N et M les valeurs dans le tableau ci-dessous.

Les paramètres du cache LL de second niveau est : ... *mettre ici les paramètres: soit ceux indiqués ligne 3 du fichier cachegrind.out.(pid) généré par valgrind: soit ceux par défaut, soit ceux que vous avez spécifiés à la main* [1] *pour LL.*

Le tableau ci-dessous est un exemple, complété avec vos résultats et ensuite analysé.

		récursif mémo			itératif			cache aware			cache oblivious		
N	M	#Irefs	#Drefs	#D1miss	#Irefs	#Drefs	#D1miss	#Irefs	#Drefs	#D1miss	#Irefs	#Drefs	#D1miss
1000	1000												
1000	1000												
2000	1000												
4000	1000												
2000	2000												
4000	4000												
6000	6000												
8000	8000												

Important: analyse expérimentale: ces mesures expérimentales sont elles en accord avec les coûts analysés théroiquement (justifier) ? Quel algorithme se comporte le mieux avec valgrind et les paramètres proposés, pourquoi ?

6.2 (3 points) Sans valgrind, par exécution de la commande :

```
distanceEdition  GCA_024498555.1_ASM2449855v1_genomic.fna  77328790  M
                 GCF_000001735.4_TAIR10.1_genomic.fna      30808129  N
```

On mesure le temps écoulé, le temps CPU et l'énergie consommée avec : *[présicer ici comment vous avez fait la mesure:* `time` *ou* `/usr/bin/time` *ou* `getimeofday` *ou* `getrusage` *ou...*

L'énergie consommée sur le processeur peut être estimée en regardant le compteur RAPL d'énergie (en microJoule) pour chaque core avant et après l'exécution et en faisant la différence. Le compteur du core K est dans le fichier `/sys/class/powercap/intel-rapl/intel-rapl:K/energy_uj` .

Par exemple, pour le cœur 0: `/sys/class/powercap/intel-rapl/intel-rapl:0/energy_uj`

Nota bene: pour avoir un résultat fiable/reproductible (si variailité), il est préférable de faire chaque mesure 5 fois et de reporter l'intervalle de confiance [min, moyenne, max].

		itératif			*cache aware*			*cache oblivious*		
N	*M*	*temps cpu*	*temps écoulé*	*energie*	*temps cpu*	*temps écoulé*	*energie*	*temps cpu*	*temps écoulé*	*energie*
10000	*10000*									
20000	*20000*									
30000	*30000*									
40000	*40000*									

Important: analyse expérimentale: *ces mesures expérimentales sont elles en accord avec les coûts analysés théroiquement (justifier) ? Quel algorithme se comporte le mieux avec valgrind et les paramètres proposés, pourquoi ?*

6.3 (1 point) Extrapolation: estimation de la durée et de l'énergie pour la commande :

```
distanceEdition  GCA_024498555.1_ASM2449855v1_genomic.fna  77328790  20236404
                 GCF_000001735.4_TAIR10.1_genomic.fna      30808129  19944517
```

A partir des résultats précédents, le programme préciser itératif/cache aware/ cache oblivious *est le plus performant pour la commande ci dessus (test 5); les ressources pour l'exécution seraient environ:* (préciser la méthode de calcul utilisée)

- *Temps cpu (en s) : ...*
- *Energie (en kWh) : ... .*

Question subsidiaire: comment feriez-vous pour avoir un programme s'exécutant en moins de 1 minute ? donner le principe en moins d'une ligne, même 1 mot précis suffit!

[1] par exemple: `valgrind --tool=cachegrind --D1=4096,4,64 --LL=65536,16,256 ...` mais ce n'est pas demandé car cela allonge le temps de simulation.