Bracquier Benjamin Solier Lilian Cateau Pierrick

TD-TP1 Méthodes Numériques :

1- Préparation du TP:

- **/proc/cpuinfo:**
  model name : AMD Ryzen 5 4600H with Radeon Graphics
  cache size : 512 KB
  cpu cores : 6
  apicid : 12
- **lstopo:**
  L3 : 4096Kb
  L2 : 512Kb
  L1: 32Kb

- **./test_poly p1 p2:**
  addition 2040 cycles
  p1+p2 4 operations 0.005098 GFLOP/s
  addition 21360 cycles
  p4+p5 1025 operations 0.124766 GFLOP/s

- **Performance Matrice:**
  Python3 : 34,3 min
  Java11 : 49,8s
  C:
  -o : 3,4 min
  -O3: 1,1 min
  ikj -O3: 2,8s
  ikj -O3 parallel : 2,7s

2ème ordinateur:

- **/proc/cpuinfo:**
  model name : Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz
  cache size : 8192 KB
  cpu cores : 6
  apicid : 0
- **lstopo:**
  L3 : 8192Kb
  L2 : 256Kb
  L1: 32Kb

- **./test_poly p1 p2:**
  addition 3506 cycles
  p1+p2 4 operations 0.002966 GFLOP/s
  addition 27665 cycles
  p4+p5 1025 operations 0.096331 GFLOP/s

- **Performance Matrice:**
  Python3 :30,416 m
  Java11 :27,369 s
  C:26.285501 s

3ème ordinateur:

- **/proc/cpuinfo:**
  model name : Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz
  cache size : 6144 KB
  cpu cores : 4
  apicid : 7

- **lstopo:**
  L3 : 6144Kb
  L2 : 512Kb
  L1: 32Kb

- **./test_poly p1 p2:**
  addition 458 cycles
  p1+p2 4 operations 0.022707 GFLOP/s
  addition 3854 cycles
  p4+p5 1025 operations 0.691489 GFLOP/s

- **Performance Matrice:**
  Python3 : 32mn15
  Java11 : 1mn04
  C: 26.928867s

2 Polynômes pleins

Nous avons choisi une implémentation par tableau, afin de stocker les coefficients des polynômes, et une structure contenant un tableau de float pour les coefficients et un entier pour le degré du polynôme.
Nous allons détailler notre code pour quelques fonctions clés :
1)creer_polynome :
Cette fonction doit retourner un pointeur sur un polynôme et doit allouer un tableau pour stocker les coefficients du polynôme.

Pour créer ce polynôme, nous allouons de la place en mémoire, celle de la structure et celle du tableau de coefficients.Cette allocation est importante car elle nous permet ensuite de gérer la libération de cette mémoire plus facilement, à l'aide de la fonction free() et elle permet aussi d'initialiser le tableau en fonction du degré.

2)init_polynome:

L'objectif de cette fonction est d'affecter la valeur x à tous les coefficients du polynômes ce qu'on a fait avec une boucle for .

3)detruire_polynome:

Pour détruire le polynôme nous libérons la place que nous avons alloué en mémoire lorsque nous avons créé le polynôme.

4)egalite_polynome:

Nous devons indiquer si 2 polynômes sont égaux avec la condition que leurs coefficients sont égaux .Ainsi nous regardons d'abord si les degrés sont égaux et sinon les polynômes ne sont pas égaux et après cela on regarde pour tous les coefficients des 2 polynômes s'ils sont égaux.

5)addition_polynome:

Dans cette fonction nous souhaitons additionner deux polynômes et retourner le polynôme total .

Pour faire cela nous allons créer un nouveau polynôme avec la fonction créer_polynome qui sera le polynôme résultat à retourner.Ensuite nous additionnons les coefficients des polynômes entre eux en fonction de leur degré associé et si un polynôme à un degré supérieur à l'autre,nous prenons à partir que l'autre polynômes n'ait plus de coefficient,le coefficient du polynôme au degré supérieur.

6)multiplication_polynome_scalaire:

Pour cette fonction,nous calculons la somme de deux polynômes et nous retournons un polynôme contenant cette somme.

Nous créons tout d'abord un nouveau polynôme que nous actualisons puis nous utilisons une boucle for afin de multiplier chaque coefficient du polynôme par le scalaire passé en paramètre.

7)eval_polynome:

Le but de cette fonction est de calculer la valeur du polynôme avec la valeur x fournie en paramètre.

Pour faire cela,nous faisons une boucle qui multipliera pour chaque coefficient du polynôme,le coefficient du polynôme avec la valeur x à la puissance du degré correspondant .

8)multiplication_polynome:

Cette fonction calcule un nouveau polynôme qui est la multiplication des deux polynômes p1 et p2.Pour cela,nous créons un polynôme ayant pour degré l'addition des degrés des deux polynômes à multiplier.Nous initialisons chaque coefficient du

polynôme à 0 au début afin de pouvoir additionner les termes (coefficients) qui ont le même degré.
Puis nous utilisons deux boucles for imbriquées afin d'imiter la distribution lors de la multiplication entre deux polynômes.
9)puissance_polynome:
L'objectif de cette fonction est de calculer un nouveau polynôme qui est le polynôme p à la puissance n.
Ainsi nous multiplions le polynômes p-1 fois par lui-même.
10)composition_polynome:
Cette fonction a pour objectif de calculer un nouveau polynôme qui est la composition de p par q($p \circ q$).
Tout d'abord,nous créons un polynôme avec un degré qui égale la multiplication des 2 polynômes ,nous avons initialisé chaque coefficient du polynôme à 0 au début afin de pouvoir additionner les termes (coefficients) qui ont le même degré puis nous réalisons une boucle for allant jusqu'au degré de p.La composition est l'addition répétée du polynôme initialisé avec la multiplication de par le coefficient de p.

3 Polynomes creux

11)La structure pour le polynôme creux est :

```
typedef struct polyf_creux_t {
  int degre ;
  float coeff;
  p_polyf_creux_t *next;
} polyf_creux_t, *p_polyf_creux_t;
```

12)Nous avons donc refait quelques fonctions pour les polynômes creux et je vais en présenter quelques une:
creer_polynome_creux:
Pour créer un monôme, nous allouons la place de la structure en mémoire et aussi pour le monôme.Nous mettons ensuite le pointeur sur le monôme suivant à next et nous retournons le monôme.
detruire_polynome_creux:
Afin de détruire un polynôme creux, nous libérons la place allouée en mémoire lorsque nous avons créé les monômes.De plus nous utilisons une boucle qui permet d'appeler la fonction sur le monôme suivant à chaque fois afin de parcourir tout le polynôme.
egalite_polynome_creux:
Pour voir si les 2 polynômes creux sont égaux,on vérifie tout d'abord si ils ne sont pas nuls,puis si les monômes ont les mêmes degrés et enfin si un monômes d'un des 2 polynômes est plus grand que le monôme de l'autre.