AOD - Distance de Fréchet

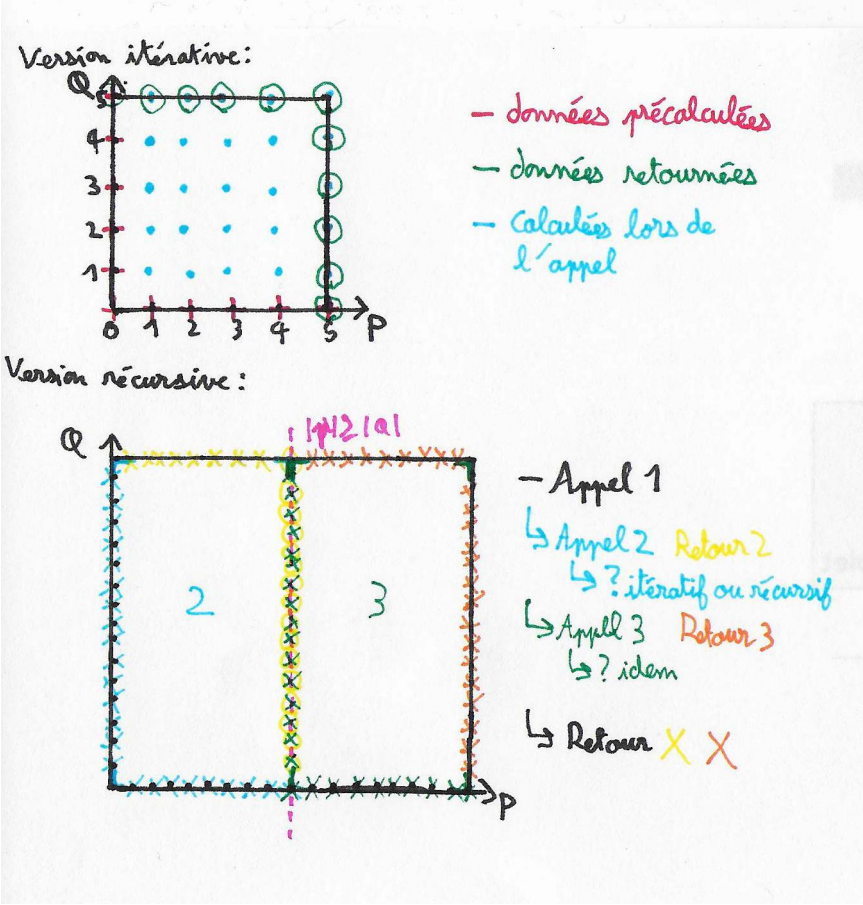
Gourgoulhon Maxime et Bouvier Pierre

Question 5 - Programmation dynamique

Voir frechet.c

Question 6 - Analyse Théorique

L'algorithme utilisé est récursif et itératif : il est est récursif jusqu'à un certain seuil, puis itératif. Il s'agit d'un algorithme de type "cache-oblivious". Voir le schéma :



Au niveau de l'implémentation, les chemins sont représentés par des listes chaînées. L'implémentation a l'avantage d'être économe en mémoire.

Version itérative

La version itérative reçoit les deux chemins du fichier d'entrée, avec les deux points départ et arrivé, ainsi que les tableau précalculés et un pointeur vers le résultat (n.b. : voir le code source pour la documentation précise des fonctions).

La fonction itérative a un coût asymptotique en nombre d'opération (qui se trouve dans la boucle principale : 12,67 instructions en moyenne) de : $(Max_p - Min_p) * (Max_q - Min_q) = n^2$

Le coût asymptotique en espace mémoire lors de l'exécution de la fonction itérative est : $((sizeof(l\_sols*) + sizeof(l\_sols)) * (Max_p - Min_p) * (Max_q - Min_q)$.

À la fin de l'exécution, les tableaux servant à écrire le résultat sont déjà alloués, le coût mémoire est :

- dans le meilleur des cas : $sizeof(l\_sols) * (Max_p - Min_p + Max_q - Min_q)$
- dans le pire cas : $sizeof(l\_sols) * (Max_p - Min_p) * (Max_q - Min_q)$

La différence se situe dans la longueur des chemins retournés, générés lors de l'exécution (nous utilisons des listes chainées, avec fusion des chemins antérieurs lorsque cela est possible)

On suppose le cache suffisemment grand pour pouvoir contenir plusieurs lignes du tableau de calcul.

Le nombre de défauts de cache dans la boucle principale est $(1 + \frac{6}{L})$. Au total (n^2) défauts de cache.

VERSION RÉCURSIVE

Pour la version récusive (hors appel récursif) :
La consomation en mémoire asymptotique : $2 * sizeof(l\_sols*) * (Max_p - Min_p + Max_q - Min_q)$
Le coût en nombre d'opérations :
- meilleur des cas : $\Theta(Max_p - Min_p + Max_q - Min_q)$
- pire des cas (cela dépend de la profondeur des élagage à faire lors de la libération des tableaux de résultats intermédiaires) : $\Theta((Max_p - Min_p) * (Max_q - Min_q))$
Le nombre de défauts de cache : $\Theta((Max_p - Min_p) + (Max_q - Min_q))$

QUESTION 7 - ANALYSE PRATIQUE

Sur un ordinateur avec Ubuntu 16.04, 8Go de RAM et un processeur i7, on observe les performances suivantes :

benchmark	temps moyen d'éxécution (en s)	mémoire max utilisée (en Mo)	défauts de cache (Valgrind %)
benchmark1	0,01	2,988	1,6
benchmark2	0,024	3,684	1,7
benchmark3	0,064	6,780	1,8
benchmark4	0,204	9,164	1,8

On a testé différents sueils pour trouver une une bonne valeur sur la machine testée (ici 1000).

	10	100	500	1000	10000
défauts de cache (Valgrind, données %)	0,4	1	1,5	1,9	2,3
temps d'exécution (Réel, en s)	0,905	0,380	0,316	0,309	0,310