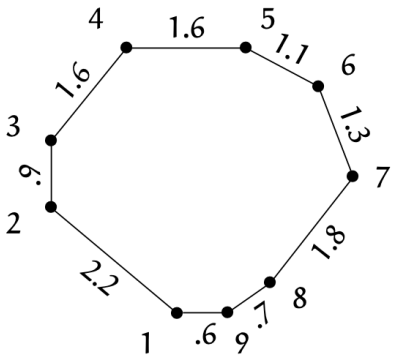
Programmation parallèle

Meilleure division du périmètre d'un polygone

Exercice 1.

On considère un polygone quelconque d'ordre N fini (N > 2) dont les sommets sont étiquetés de 1 à N dans le sens des aiguilles d'une montre. On recherche l'une quelconque des cordes qui partage le périmètre p du polygone « le plus exactement possible », c'est-à-dire telle que la valeur absolue de la différence entre la somme des longueurs des côtés délimitées par la corde soit minimale. La complexité est évaluée en nombre de sommets visités.

Définitions – Notations – Représentation
Dans la suite, |a| dénote la valeur absolue de a. Le côté qui a comme origine le sommet i est noté (i). La corde qui joint les sommets i et j est notée (i, j). L'arc qui va du sommet i au sommet j est noté y (i, j). La longueur d'un arc a est notée ||a||. Un polygone d'ordre N est représenté par un tableau d (d $\in$ 1 .. N $\rightarrow$ R+) tel que d[i] est la longueur du côté (i). La figure 3.1, illustre le cas d'un polygone d'ordre 9.
Fig. 1 – Exemple d'un polygone d'ordre 9 et de sa représentation par le tableau d



d[1]	=	2.2
d[2]	=	0.9
d[3]	=	1.6
d[4]	=	1.6
d[5]	=	1.1
d[6]	=	1.3
d[7]	=	1.8
d[8]	=	0.7
d[9]	=	0.6

Plus formellement, si l'opérateur · dénote la concaténation d'arcs adjacents, un arc (non vide) se définit de façon inductive de la manière suivante comme une succession de côtés.

Définition 1 (Arc) : Cas de base :

$$\begin{cases} (i, \overset{\frown}{i+1}) & = (i) \\ (\overset{\frown}{N,1}) & = (N). \end{cases} \qquad i \in 1 \,..\, N-1$$

Cas inductif :

$$\begin{cases} (\overset{\frown}{i,j}) & = (i, \overset{\frown}{i+1}) \cdot (\overset{\frown}{i+1}, j) \\ (\overset{\frown}{N,j}) & = (\overset{\frown}{N,1}) \cdot (\overset{\frown}{1,j})) \end{cases} \qquad \begin{matrix} j \neq i+1 \text{ et } i \neq N \\ j \neq 1. \end{matrix}$$

De même, la longueur d'un arc se définit comme suit.

Définition 2 (Longueur d'un arc) :

Cas de base :

$$\begin{cases} \|(i, \overset{\frown}{i+1})\| & = d[i] \\ \|(\overset{\frown}{N, 1})\| & = d[N]. \end{cases} \qquad i \in 1\,..\,N-1$$

Cas inductif :

$$\begin{cases} \|(\overset{\frown}{i, j})\| & = d[i] + \|(\overset{\frown}{i+1, j})\| \\ \|(\overset{\frown}{N, j})\| & = d[N] + \|(\overset{\frown}{1, j})\| \end{cases} \qquad \begin{matrix} j \neq i+1 \text{ et } i \neq N \\ j \neq 1. \end{matrix}$$

Définition 3 (Corde localement optimale) :

La corde (j, k) est localement optimale par rapport à son origine j si :

$$\left| \|(\overset{\frown}{j, k})\| - \|(\overset{\frown}{k, j})\| \right| = \min_{i \in 1..N-\{j\}} \left(\left| \|(\overset{\frown}{j, i})\| - \|(\overset{\frown}{i, j})\| \right| \right).$$

Définition 4 (Corde globalement optimale) :

La corde (j, k) est globalement optimale si :

$$\left| \|(\overset{\frown}{j, k})\| - \|(\overset{\frown}{k, j})\| \right| = \min_{\substack{u \in 1\,..\,N \text{ et} \\ v \in 1\,..\,N \text{ et} \\ u \neq v}} \left(\left| \|(\overset{\frown}{u, v})\| - \|(\overset{\frown}{v, u})\| \right| \right). \qquad \textbf{Equation 3.1.}$$

Le problème

L'objectif de l'exercice est de trouver une corde globalement optimale (le problème possède au moins une solution). Une méthode consiste à évaluer la formule 3.1 et à retenir l'un des couples de sommets qui la satisfait. L'algorithme correspondant est en $\Theta(N2)$. Il est également possible d'exploiter l'identité suivante afin d'éviter des calculs inutiles (p est le périmètre du polygone) :

$$\left| \|(\overset{\frown}{i, j})\| - \|(\overset{\frown}{j, i})\| \right| = 2 \cdot \left| \|(\overset{\frown}{i, j})\| - \frac{p}{2} \right|. \qquad \textbf{Équation 3.2.}$$

Les couples de sommets (j, k) qui satisfont la formule (3.1) sont également ceux qui vérifient :

$$\frac{\left| \|(\overset{\frown}{j, k})\| - \|(\overset{\frown}{k, j})\| \right|}{2} = \min_{\substack{u \in 1\,..\,N \text{ et} \\ v \in 1\,..\,N \text{ et} \\ u \neq v}} \left| \|(\overset{\frown}{u, v})\| - \frac{p}{2} \right|.$$

Mais, puisque |a − b| = |b − a|, pour visiter tous les arcs, la variable v n'a pas besoin de parcourir la totalité de l'ensemble 1 .. N−{u}. Il suffit qu'elle prenne ses valeurs dans l'intervalle u + 1 .. N ∪ {1}, d'où :

$$\frac{\left|\|(\overset{\frown}{j,k})\| - \|(\overset{\frown}{k,j})\|\right|}{2} = \min_{\substack{u \in 1\,..\,N \text{ et} \\ v \in u+1\,..\,N \cup \{1\}}} \left|\|(\overset{\frown}{u,v})\| - \frac{p}{2}\right|.$$

Si (j, k) est une corde localement optimale issue de j, la valeur $\left(\left|\|(\overset{\frown}{j,k})\| - p/2\right|\right)$ est notée δj . Les longueurs des arcs intervenant dans le calcul peuvent être rassemblées dans la matrice triangulaire suivante (appelée triangle des longueurs dans la suite) :

$$\begin{array}{ccccccccc}
\|(\overset{\frown}{1,2})\| & \|(\overset{\frown}{1,3})\| & & \cdots\cdots\cdots & & & \|(\overset{\frown}{1,1})\| \\
 & \|(\overset{\frown}{2,3})\| & & \cdots\cdots\cdots & & & \|(\overset{\frown}{2,1})\| \\
 & & & & \vdots & & \\
 & & \|(i,\overset{\frown}{i+1})\| & & \cdots\cdots & & \|(\overset{\frown}{i,1})\| \\
 & & & & \ddots & & \vdots \\
 & & & & & \ddots & \vdots \\
 & & & & & & \vdots \\
 & & & & & & \|(\overset{\frown}{N,1})\|
\end{array}$$

On notera que le coin supérieur droit du triangle donne la valeur du périmètre p = k y (1, 1)k du polygone. Dans le cas du polygone de la figure 3.1 page 37, p = 11.8, et ce triangle s'instancie de la manière suivante :

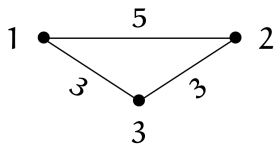
	2	3	4	5	6	7	8	9	1
1	2.2	3.1	4.7	6.3	7.4	8.7	10.5	11.2	11.8
2		0.9	2.5	4.1	5.2	6.5	8.3	9.0	9.6
3			1.6	3.2	4.3	5.6	7.4	8.1	8.7
4				1.6	2.7	4.0	5.8	6.5	7.1
5					1.1	2.4	4.2	4.9	5.5
6						1.3	3.1	3.8	4.4
7							1.8	2.5	3.1
8								0.7	1.3
9									0.6

Cependant, ce type de triangle contient (N(N + 1))/2 éléments et les évaluer tous conduit à nouveau à une solution en Θ(N2). Ce résultat peut être amélioré sur la base des quatre observations suivantes :

Pour un sommet j donné, il existe soit une, soit deux cordes optimales locales.
Si la corde (j, k) est la première 1 corde optimale locale issue de j, alors aucune des cordes (j + 1, j + 2), (j + 1, j + 3),. . . ,(j + 1, k − 1) n'est globalement aussi bonne que (j, k).
Soit (j, k) la première corde optimale locale issue de j. Si elle existe, la corde (j + 1, k) peut être meilleure que (j, k).
Lorsque, pour le sommet j, on a découvert une corde localement optimale (j, k), il est inutile d'évaluer l'optimalité de (j, k + 1),. . . , (j, N + 1) (principe classique de « l'arrêt au plus tôt »). Il convient de noter que, pour ce qui concerne l'observation 3, d'autres cordes issues du sommet j + 1 peuvent surclasser la corde (j + 1, k).

Questions :

1. Dans quel cas existe-t-il deux cordes optimales locales issues d'un même sommet j ? Proposer un exemple.
2. Démontrer la proposition relative à la seconde observation ci-dessus.
3. Démontrer la proposition relative à la troisième observation ci-dessus.
4. Dans l'exemple de la figure 3.1, quels sont les arcs dont la longueur intervient dans le calcul ? Quelle corde globalement optimale est-elle découverte ?
5. Considérons l'exemple suivant :



    a. Fournir, sous la forme d'un triangle des longueurs, les arcs dont la longueur intervient dans le calcul. Que constate-t-on ?
6. On décide de construire une solution algorithmique sur la base d'une seule boucle.
    a. Proposer un invariant pour cette boucle.
7. Proposer une stratégie de division de problèmes qui permet de résoudre efficacement la version parallèle de cet algorithme. En déduire l'algorithme correspondant.
8. Quelles sont les contraintes auxquelles vous êtes confrontées: types de division, communication overhead, équilibrage de charges, ou autre ?
9. Proposer sur MPI en C un exemple d'implémentation de cet algorithme parallèle et en déduire le gain de temps en appliquant la méthode d'Amdahl.
10. Serait-il possible d'adapter la charge du problème avec les ressources, comme indiqué dans le modèle de Gustafson ? Appuyer votre réponse par des commentaires tirés du cours.

Circuits et chemins eulériens – tracés d'un seul trait

Exercice 2.

Cet exercice s'intéresse aux parcours eulériens dans un graphe orienté connexe. Partant de l'algorithme pour la recherche d'un circuit eulérien dans un graphe orienté obtenu dans la troisième question, on demande de le transformer afin de rechercher un chemin eulérien dans un graphe non orienté. Une application aux tracés d'un seul trait (c'est-à-dire aux tracés pour lesquels on ne lève pas la plume et on ne repasse pas sur un trait) est étudiée.

On considère un graphe orienté G = (N, V) dans lequel N est l'ensemble des sommets et V l'ensemble des arcs. Posons n = card(V). On étudie tout d'abord le problème des circuits eulériens, puis celui des chemins eulériens.

Questions :

Pour chacun des graphes ci-dessous, fournir, s'il en existe, l'un des circuits eulériens.
Dans le cadre de la recherche de tous les circuits eulériens pour le graphe orienté G, on choisit une structure d'énumération X contenant des sommets. Compléter la définition de X considérée comme une fonction, en déduire le type de patron qui doit s'appliquer.
Instancier la procédure générique ToutesSolutions(i) (voir figure ??, page ??), afin d'obtenir un algorithme à essais successifs permettant d'afficher tous les circuits eulériens d'un graphe orienté connexe.
Le problème du tracé sans lever la plume se pose en des termes différents de la recherche d'un circuit dans un graphe orienté. En effet, un graphe non orienté connexe est fourni et il s'agit de découvrir un chemin eulérien (c'est-à-dire une succession de sommets qui passe une et une seule fois par chacune des arêtes – soit dans un sens, soit dans un autre – sans nécessairement revenir au sommet de départ.

En revanche, ce chemin peut franchir un nœud autant de fois qu'on l'estime nécessaire).

La partie (a) du schéma ci-dessous représente le dessin qu'il faut réaliser sans lever la plume et sans passer plusieurs fois sur le même trait (les cercles placés aux sommets ne font pas partie du dessin).
La partie (b) montre une solution possible. Il s'agit d'un chemin eulérien débutant au sommet h ; ce chemin constitue un graphe orienté qui se superpose au graphe initial. Chaque arc est accompagné du numéro d'ordre du parcours. Expliquer les modifications qu'il faut apporter à l'algorithme de la troisième question pour résoudre cette variante du problème initial.

5. Discuter des possibilités de parallélisation de cet algorithme.

6. Quel est le meilleur choix en termes de structures de données ? Linéaire, arborescente, etc
7. Présenter un algorithme parallèle avec répartition de tâches sur plusieurs processeurs pour un graphe de de dimension N assez grand.
8. Estimer les temps de calcul et de performance en se référent aux deux modèles vus en cours: la loi d'Amdahl et la loi de Gustafson.