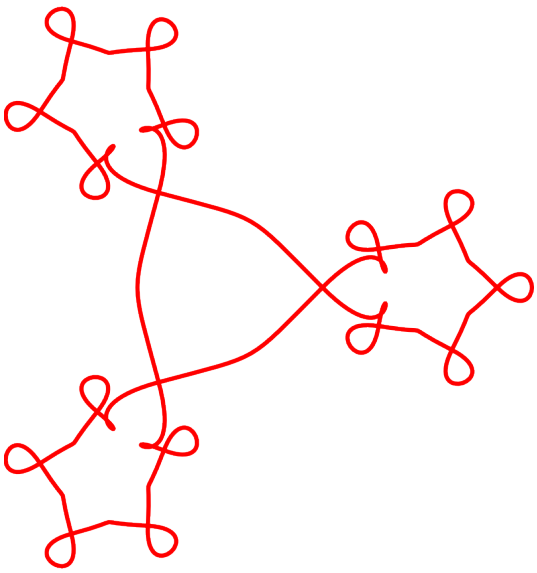
Sac de nœuds

Algo1

10 décembre 2014



1 Introduction

Nous cherchons pour ce sujet à implémenter un algorithme de tracé de nœuds celtiques.

L'algorithme est expliqué sur le site *images des mathématiques* du CNRS à l'adresse `http://images.math.cnrs.fr/De-beaux-entrelacs.html` (nous nous intéressons ici uniquement à la version la plus simple).

Nous vous fournissons un ensemble de fichiers d'entrée, contenant des graphes au format *kn* (maison) ainsi qu'un générateur automatique de graphes *bananes*[1]. Votre objectif est de transformer automatiquement un fichier *kn* en dessin de nœud au format *svg*.

L'accent sur ce projet est porté sur la conception de structures de données. Votre programme va ici manipuler plusieurs sortes de données. À vous de choisir les structures permettant une écriture d'algorithme le plus simple

1. `http://graphstream-project.org/doc/Generators/Overview-of-generators_1.0/`

possible. Pour information, la correction utilise 6 différents types de structures.

2 Format de fichiers d'entrée

Un fichier *kn* est un fichier ASCII respectant la spécification suivante :

- la première ligne contient un entier :le nombre s de sommets.
- suivent s groupes de 3 lignes contenant chacune un sommet défini par :
  - la position du sommet définie par 2 flottants (séparés par des espaces) ;
  - un nombre entier δ_i indiquant combien d'arètes sont adjacentes au sommet i ;
  - δ_i entiers indiquants les indices des sommets extrémités des arètes.

Par exemple, le graphe de la figure 1 sera encodé par :

```
4
0 0
3
2 3 4
-0.5 -1
2
1 3
0.5 -1
2
1 2
1 1
1
1
```

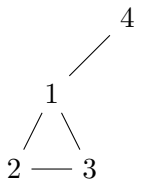


FIGURE 1 – Graphe exemple

3 Objectif intermédiaire

Avant de tracer le nœud en lui même, on se propose de tracer les dessins intermédiaires correspondant aux arètes ainsi qu'aux points d'intersection.

Il s'agit ici de dessiner dans un fichier svg :

— toutes les arètes,

— une croix par arète, formée de deux segments de taille identique, tournés à 45 degrés par rapport à l'arète.

On obtient ainsi le dessin de la figure 2 sur notre exemple précédent. Plusieurs autres exemples sont disponibles parmi les fichiers fournis. Le tracé intermédiaire y est dessiné en noir.

Les 4 points formant les extrémités de la croix sont importants et serviront pour le tracé du nœud.

Le rendu intermédiaire permet donc de s'assurer que le graphe est chargé correctement et que les points sont calculés correctement. Notons qu'il est possible d'obtenir des dessins plus jolis en prenant un diamètre de croix variable (égal à la moitié de la longueur de l'arète par exemple).

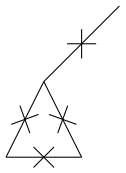


FIGURE 2 – Tracé intermédiaire

4 Objectif final

On s'intéresse maintenant au tracé final du nœud. Comme expliqué sur le site web du CNRS, nous progressons de milieu d'arète en milieu d'arète, dessinant à chaque fois un morceau du nœud.

Il nous faut donc répondre aux deux questions suivantes :

— comment trouver le milieu suivant ?

— comment tracer le morceau du nœud en svg ?

En fait ces deux questions sont liées aux quatre points des croix dessinées lors du rendu intermédiaire. Commençons par le tracé en lui même. Pour tracer une courbe en *svg*, le mieux est d'utiliser une courbe de Bézier. Notre courbe sera ici constituée de 4 points. Aux extrémités, le point de départ, au milieu du segment de départ, ainsi que le point d'arrivée au milieu du segment d'arrivée. Au milieu, deux points de contrôle, qui permettent de "courber" ce qui ne serait autrement qu'un segment. Le premier point fait partie de la croix de départ, et le second de la croix d'arrivée.

Pour mieux comprendre les choses, nous proposons de nommer plus précisemment ces points de contrôle. Étant donné un segment (a, b) on dispose de 4 points de contrôles. Deux d'entre eux sont proches de a et les deux autres de b. Regardons maintenant les 2 points proches de a. Si l'on fait tourner

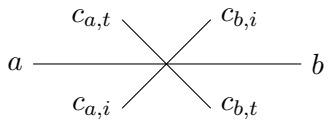


FIGURE 3 – Points de contrôle

(a, b) autour de a dans le sens trigonométrique on recontrera d'abord un point, puis l'autre. On peut ainsi nommer les 4 points de contrôle comme sur la figure 3 ('t' dénote le sens trigonométrique et 'i' le sens inverse).

Avec ces notations l'algorithme s'exprime de manière plus simple. Supposons que l'on se trouve sur une arète $e = (v_1, v_2)$, en partance vers le point de contrôle $c_{v_1,t}$. On sait que l'on se dirige vers v_1. Pour trouver l'arète suivante, il nous faut donc tourner autour de v_1. Dans quelle direction ? Encore une fois, le point de contrôle choisi nous indique la réponse : il faut ici tourner dans le sens trigonométrique. On calcule donc l'angle que fait chaque arète incidente à v_1 avec l'arète courante puis on prend celui d'angle minimal (une variante plus "nouée" serait de prendre l'angle maximal ; vous pouvez prendre ici le min ou max, au choix). Soit $e_2 = (v_1, v_3)$ l'arète ainsi obtenue. Vers quel point de contrôle sur cette arète nous dirigeons nous ? En fait nous sommes toujours proches de v_1, en arrivant par contre de l'autre côté. Nous sommes donc sur le point $c_{v_1,i}$ de e_2, en direction du milieu de e_2. On continuera ensuite à l'itération suivante en traversant le milieu de e_2 pour repartir en direction de $c_{v_3,i}$.

La figure 4 illustre ce processus sur l'exemple donné.

5 Consignes

5.1 Code

Il est une fois encore important de garder un code le plus clair possible et la clarté de ce que vous écrivez sera le critère le plus important pour la notation. N'hésitez donc pas à structurer votre projet correctement : pas de procédures trop longues ; des commentaires sur les parties complexes ; des noms de variables faisant sens ; un code indenté (attention : soit tabulations soit espaces) ; des paquetages permettant de bien organiser le projet. Attention donc à bien réfléchir aux meilleurs types de données.

Vous rendrez une archive contenant le code source et le rapport via *TEIDE*. Attention à ne pas inclure les fichiers issus de la compilation dans l'archive, les correcteurs recompileront votre code source pour obtenir l'exécutable. Attention aussi aux fichiers cachés (ceux dont le nom commence par un '.')

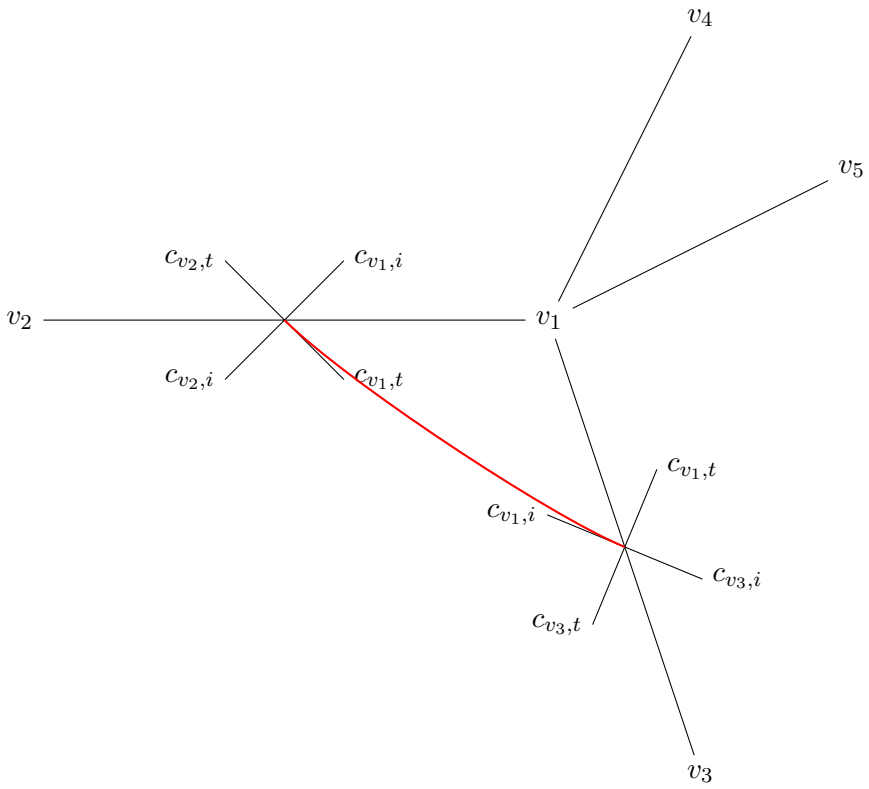


FIGURE 4 – Avancement

5.2 Rapport

Nous vous demandons 2 pages (max) de rapport au format *pdf* que vous incluerez dans l'archive que vous rendrez électroniquement sur *TEIDE*.

Le rapport a pour objectif de permettre aux correcteurs une plongée plus facile dans votre code.

5.3 Rendu et évaluation

Le projet est à rendre au plus tard le vendredi 23 janvier 2015 à 23h59. Tout retard est sujet à une pénalité.

L'accent du projet est mis sur la clarté. Il est possible de réaliser plus de fonctionnalités que prévues pour le projet (plus de points de contrôle, meilleur rendu svg ...). Néanmoins ces fonctionnalités ne seront pas un critère déterminant de votre note. C'est par contre l'occasion de vous faire plaisir ou de profiter du retour des correcteurs sur votre programme.