

IKHLEF Rafik
LI213_groupe 1
n° 2303887

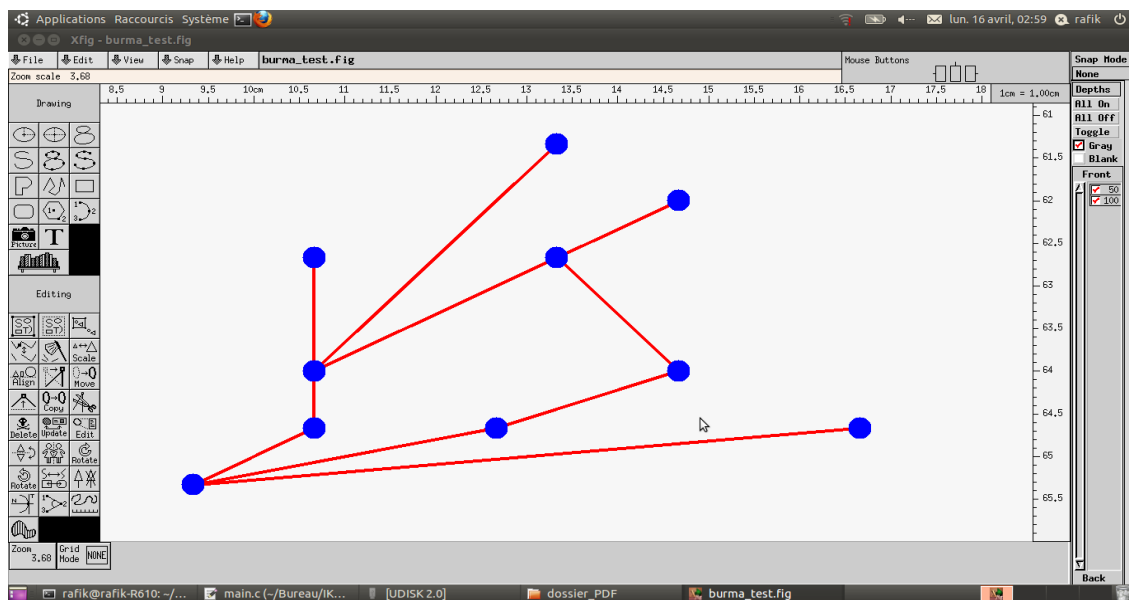
EXERCICE 1 :

les fonctions sont dans le fichier fonction.c , fonction.h et structure1.h

création des instances a partir des listes des chaines. Et affichage des fichiers XFIG :

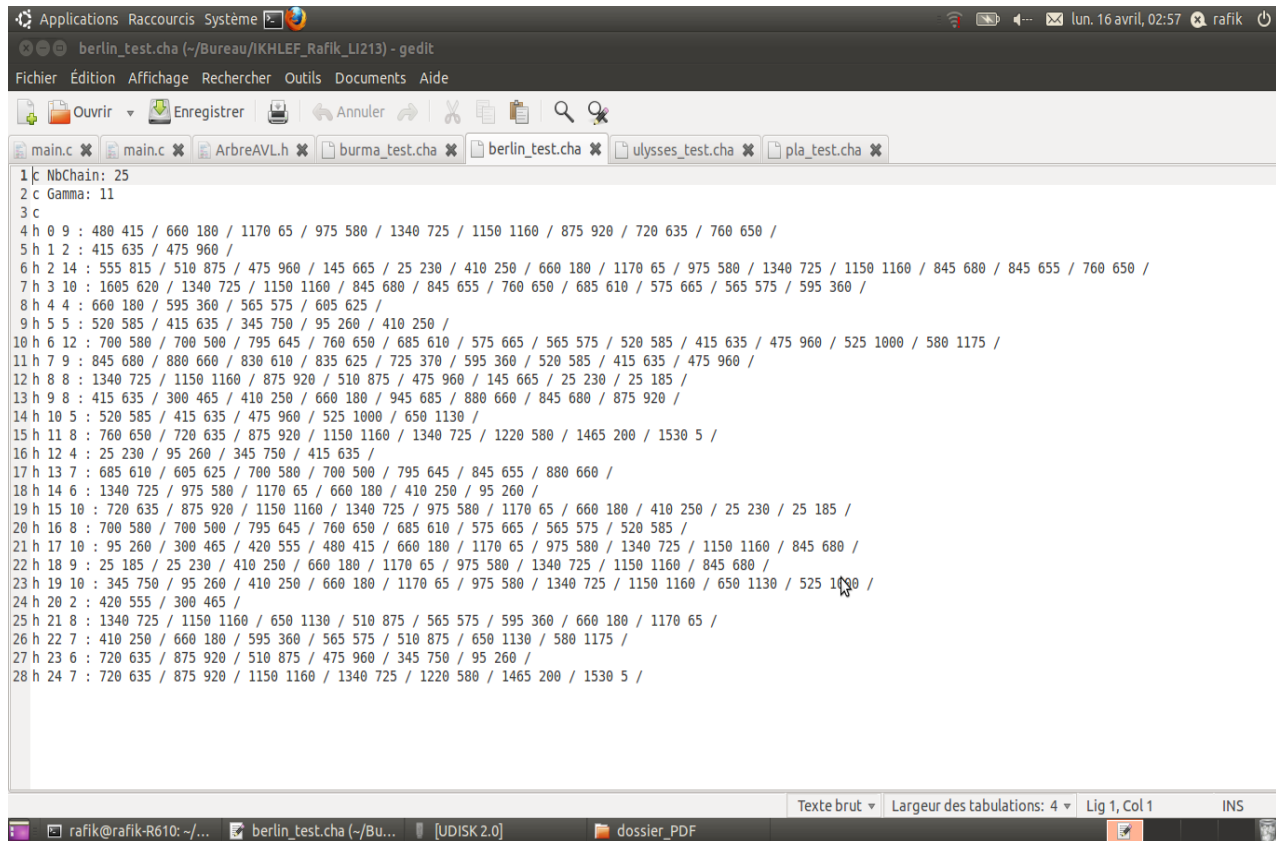
instance n°1 : « 00014 burma.cha » :

```
1 k NbChain: 8
2 c Gamma: 4
3 c
4 h 0 6 : 25.23 97.24 / 14.05 98.12 / 16.3 97.38 / 16.47 94.44 / 16.47 96.1 / 20.09 92.54 /
5 h 1 5 : 14.05 98.12 / 16.3 97.38 / 16.47 94.44 / 16.47 96.1 / 22.39 93.37 /
6 h 2 5 : 16.47 96.1 / 16.47 94.44 / 16.3 97.38 / 14.05 98.12 / 25.23 97.24 /
7 h 3 4 : 14.05 98.12 / 19.41 97.13 / 22 96.05 / 20.09 94.55 /
8 h 4 2 : 22.39 93.37 / 16.47 96.1 /
9 h 5 3 : 14.05 98.12 / 19.41 97.13 / 22 96.05 /
10 h 6 3 : 16.3 97.38 / 14.05 98.12 / 25.23 97.24 /
11 h 7 4 : 22.39 93.37 / 16.47 96.1 / 16.47 94.44 / 16.3 97.38 /
```

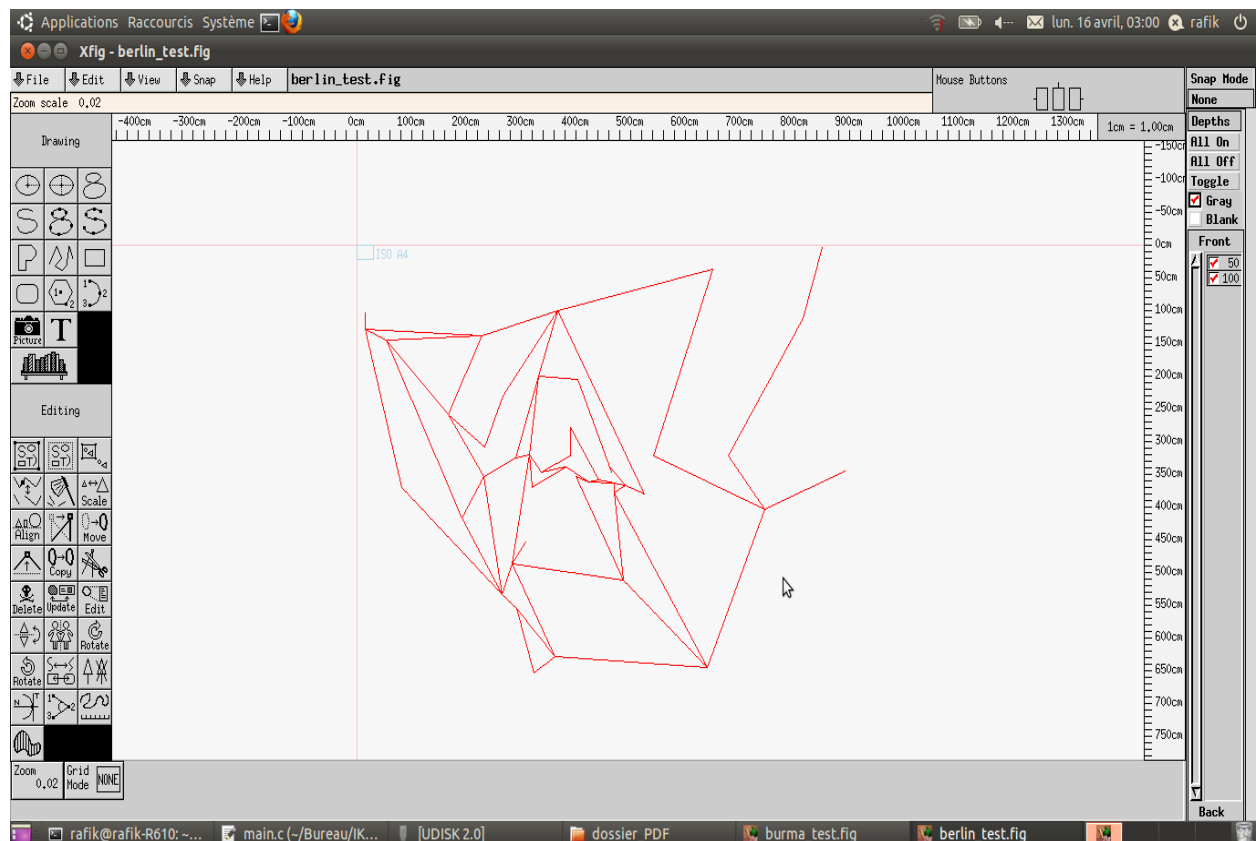


longueur totale est : 105,11
nombre de point est : 32

instance n°2 : « 00052 berlin.cha » : longueur totale est : 45552.52 . le nombre de points est : 188

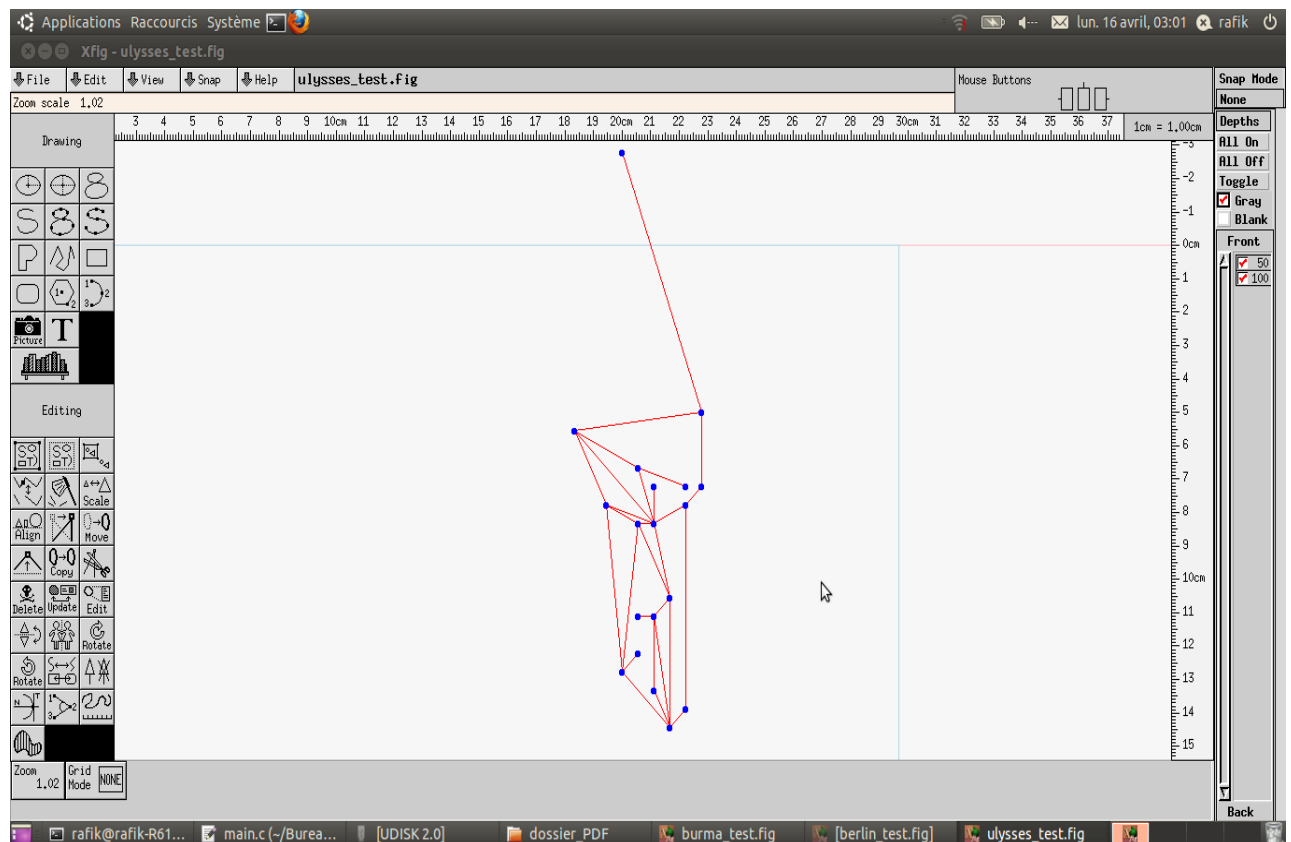


```
1 c NbChain: 25
2 c Gamma: 11
3 c
4 h 0 9 : 480 415 / 660 180 / 1170 65 / 975 580 / 1340 725 / 1150 1160 / 875 920 / 720 635 / 760 650 /
5 h 1 2 : 415 635 / 475 960 /
6 h 2 14 : 555 815 / 510 875 / 475 960 / 145 665 / 25 230 / 410 250 / 660 180 / 1170 65 / 975 580 / 1340 725 / 1150 1160 / 845 680 / 845 655 / 760 650 /
7 h 3 10 : 1605 620 / 1340 725 / 1150 1160 / 845 680 / 845 655 / 760 650 / 685 610 / 575 665 / 565 575 / 595 360 /
8 h 4 4 : 660 180 / 595 360 / 565 575 / 685 625 /
9 h 5 5 : 520 585 / 415 635 / 345 750 / 95 260 / 410 250 /
10 h 6 12 : 780 580 / 780 580 / 795 645 / 760 650 / 685 610 / 575 665 / 565 575 / 520 585 / 415 635 / 475 960 / 525 1000 / 580 1175 /
11 h 7 9 : 845 680 / 880 660 / 830 610 / 835 625 / 725 370 / 595 360 / 520 585 / 415 635 / 475 960 /
12 h 8 8 : 1340 725 / 1150 1160 / 875 920 / 510 875 / 475 960 / 145 665 / 25 230 / 25 185 /
13 h 9 8 : 415 635 / 300 465 / 410 250 / 660 180 / 945 685 / 880 660 / 845 680 / 875 920 /
14 h 10 5 : 520 585 / 415 635 / 475 960 / 525 1000 / 650 1130 /
15 h 11 8 : 760 650 / 720 635 / 875 920 / 1150 1160 / 1340 725 / 1220 580 / 1465 200 / 1530 5 /
16 h 12 4 : 25 230 / 95 260 / 345 750 / 415 635 /
17 h 13 7 : 685 610 / 685 625 / 700 580 / 700 580 / 795 645 / 845 655 / 880 660 /
18 h 14 6 : 1340 725 / 975 580 / 1170 65 / 660 180 / 410 250 / 95 260 /
19 h 15 10 : 720 635 / 875 920 / 1150 1160 / 1340 725 / 975 580 / 1170 65 / 660 180 / 410 250 / 25 230 / 25 185 /
20 h 16 8 : 780 580 / 780 580 / 795 645 / 760 650 / 685 610 / 575 665 / 565 575 / 520 585 /
21 h 17 10 : 95 260 / 300 465 / 420 555 / 480 415 / 660 180 / 1170 65 / 975 580 / 1340 725 / 1150 1160 / 845 680 /
22 h 18 9 : 25 185 / 25 230 / 410 250 / 660 180 / 1170 65 / 975 580 / 1340 725 / 1150 1160 / 845 680 /
23 h 19 10 : 345 750 / 95 260 / 410 250 / 660 180 / 1170 65 / 975 580 / 1340 725 / 1150 1160 / 650 1130 / 525 1130 /
24 h 20 2 : 420 555 / 300 465 /
25 h 21 8 : 1340 725 / 1150 1160 / 650 1130 / 510 875 / 565 575 / 595 360 / 660 180 / 1170 65 /
26 h 22 7 : 410 250 / 660 180 / 595 360 / 565 575 / 510 875 / 650 1130 / 580 1175 /
27 h 23 6 : 720 635 / 875 920 / 510 875 / 475 960 / 345 750 / 95 260 /
28 h 24 7 : 720 635 / 875 920 / 1150 1160 / 1340 725 / 1220 580 / 1465 200 / 1530 5 /
```

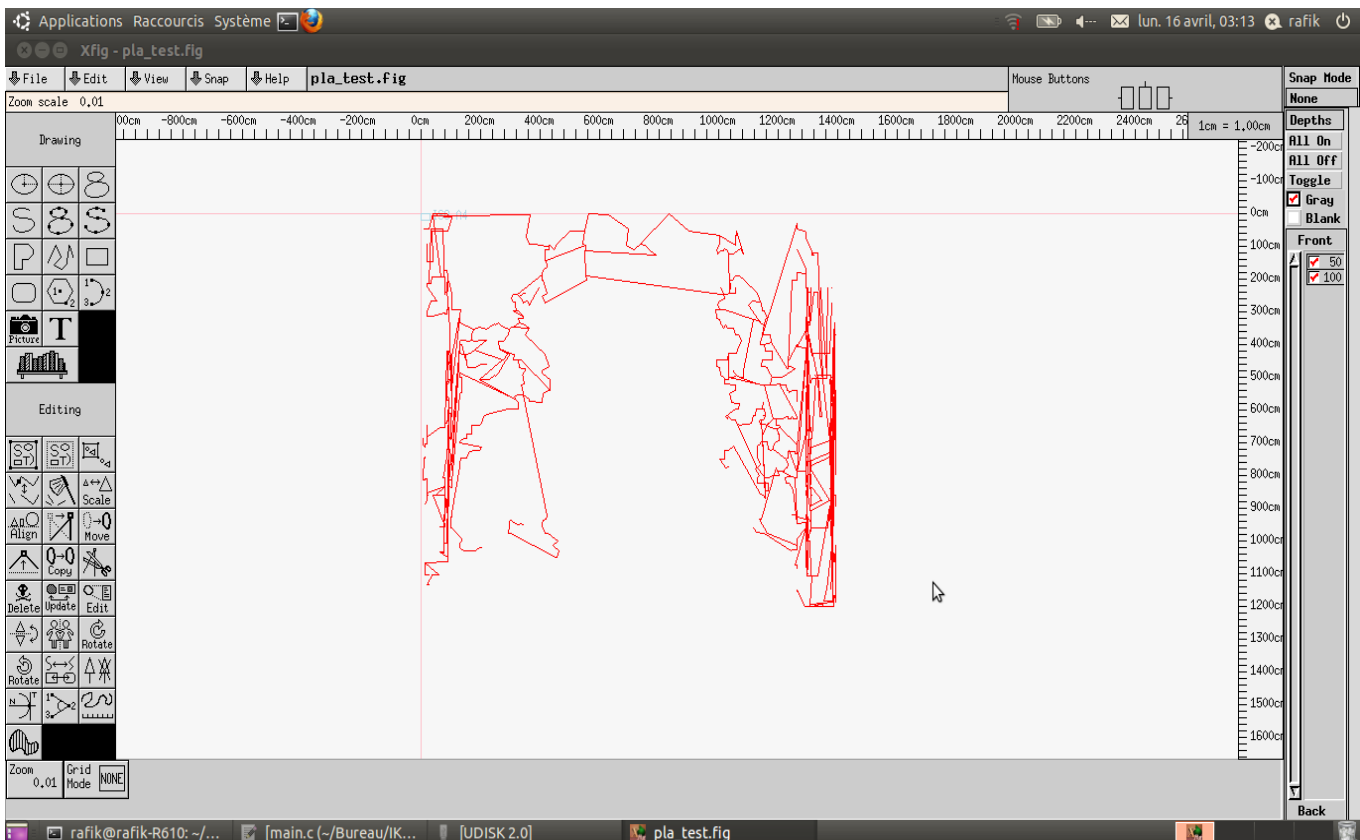
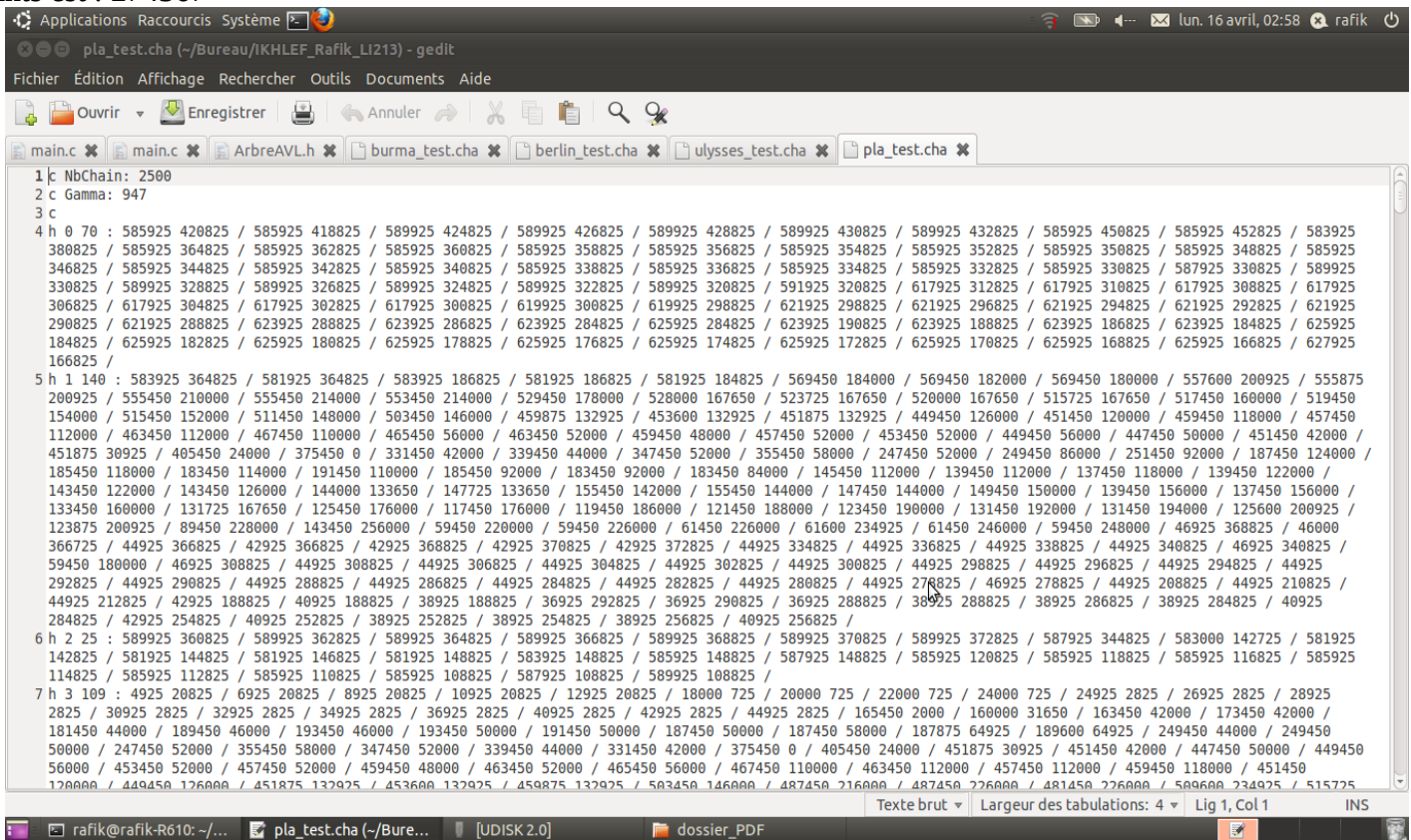


instance n°3 : « 00022 ulysses.cha » : la longueur totale est : 336 . le nombre de point est : 90

```
Applications Raccourcis Système
ulysses_test.cha (~Bureau/IKHLEF_Rafik_LI213) - gedit
Fichier Édition Affichage Rechercher Outils Documents Aide
Ouvrir Enregistrer Annuler
main.c main.c ArbreAVL.h burma_test.cha berlin_test.cha ulysses_test.cha pla_test.cha
1 k NbChain: 20
2 c Gamma: 6
3 c
4 h 0 5 : 36.08 -5.21 / 41.23 9.1 / 33.48 10.54 / 35.49 14.32 / 36.09 23 /
5 h 1 6 : 40.44 13.57 / 37.56 12.19 / 33.48 10.54 / 35.49 14.32 / 37.51 15.17 / 36.26 23.12 /
6 h 2 3 : 37.52 20.44 / 38.24 20.42 / 38.09 24.36 /
7 h 3 3 : 37.57 22.56 / 36.26 23.12 / 37.51 15.17 /
8 h 4 4 : 41.23 9.1 / 33.48 10.54 / 35.49 14.32 / 36.09 23 /
9 h 5 4 : 38.09 24.36 / 39.57 26.15 / 40.56 25.32 / 40.37 14.23 /
10 h 6 4 : 39.36 19.56 / 38.15 15.35 / 35.49 14.32 / 36.09 23 /
11 h 7 5 : 37.57 22.56 / 36.26 23.12 / 39.57 26.15 / 40.56 25.32 / 40.37 14.23 /
12 h 8 2 : 36.08 -5.21 / 41.23 9.1 /
13 h 9 3 : 41.23 9.1 / 41.17 13.05 / 40.37 14.23 /
14 h 10 6 : 37.56 12.19 / 38.47 15.13 / 40.37 14.23 / 40.56 25.32 / 39.57 26.15 / 39.36 19.56 /
15 h 11 4 : 33.48 10.54 / 38.15 15.35 / 39.36 19.56 / 38.24 20.42 /
16 h 12 4 : 35.49 14.32 / 36.09 23 / 36.26 23.12 / 37.57 22.56 /
17 h 13 6 : 38.42 13.11 / 38.47 15.13 / 38.15 15.35 / 37.51 15.17 / 36.26 23.12 / 39.57 26.15 /
18 h 14 7 : 36.08 -5.21 / 41.23 9.1 / 33.48 10.54 / 35.49 14.32 / 37.51 15.17 / 39.36 19.56 / 38.24 20.42 /
19 h 15 6 : 37.52 20.44 / 38.24 20.42 / 39.57 26.15 / 40.56 25.32 / 40.37 14.23 / 38.47 15.13 /
20 h 16 3 : 36.26 23.12 / 39.57 26.15 / 40.56 25.32 /
21 h 17 7 : 40.56 25.32 / 40.37 14.23 / 38.47 15.13 / 33.48 10.54 / 35.49 14.32 / 37.51 15.17 / 39.36 19.56 /
22 h 18 2 : 33.48 10.54 / 35.49 14.32 /
23 h 19 6 : 37.56 12.19 / 38.47 15.13 / 38.15 15.35 / 39.36 19.56 / 38.24 20.42 / 38.09 24.36 /
Texte brut Largeur des tabulations: 4 Lig 1, Col 1 INS
```



instance n°4: « 07397 pla.cha plus grande instance » : longueur totale est : 3309558964.59 . le nombre de points est : 274367

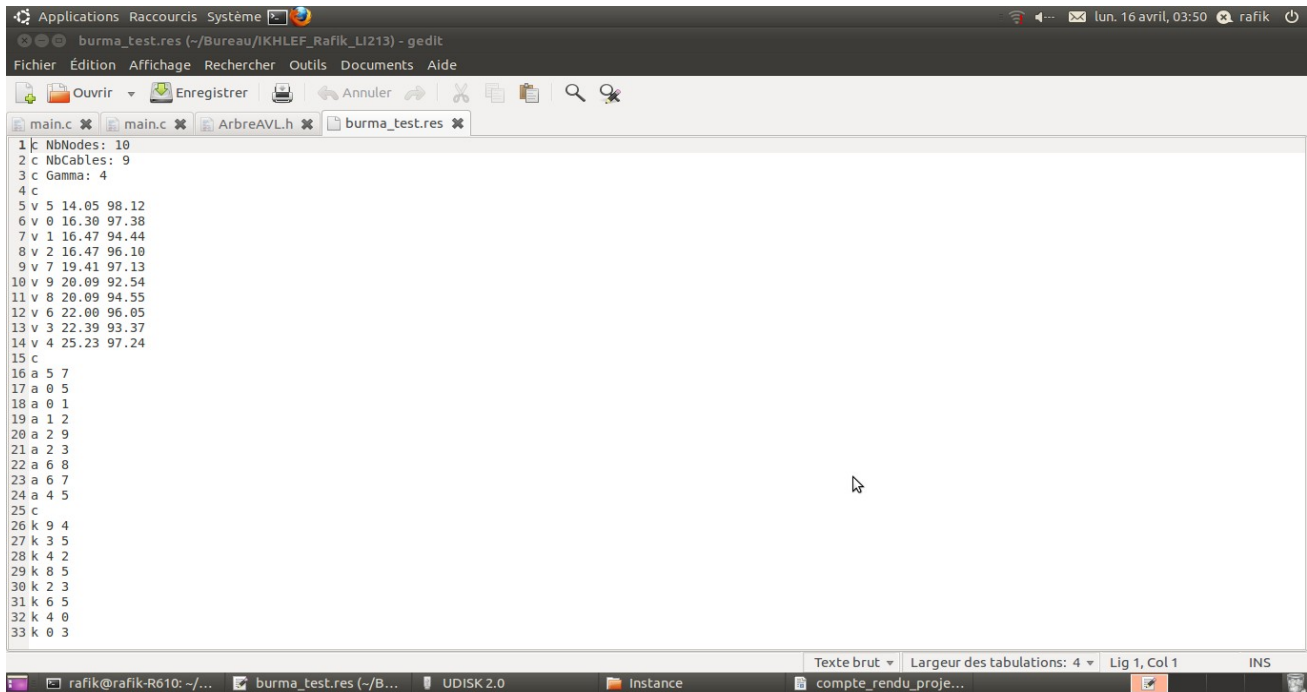


complexité pire cas est de $O(n^2)$ dans le cas où il y a N chaînes et dans chaque chaîne on trouve N points.

Exercices 2 :

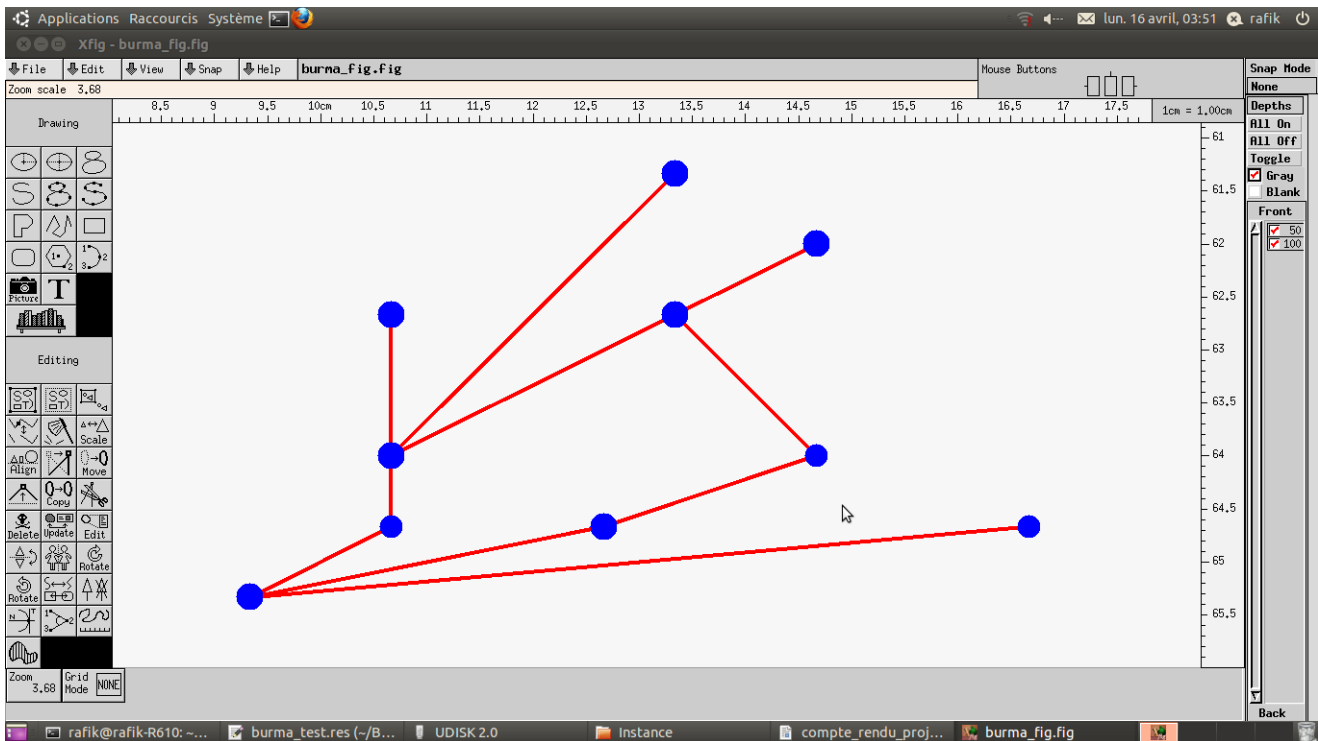
les fonctions sont dans les fichiers : fonction_reseau_exo2.c . fonction_reseau_exo2.h. Et structure_reseau.h
le but est de reproduire l'instance reseau.

instance n°1 : « 00014_burma.res » ==> burma_test.res:



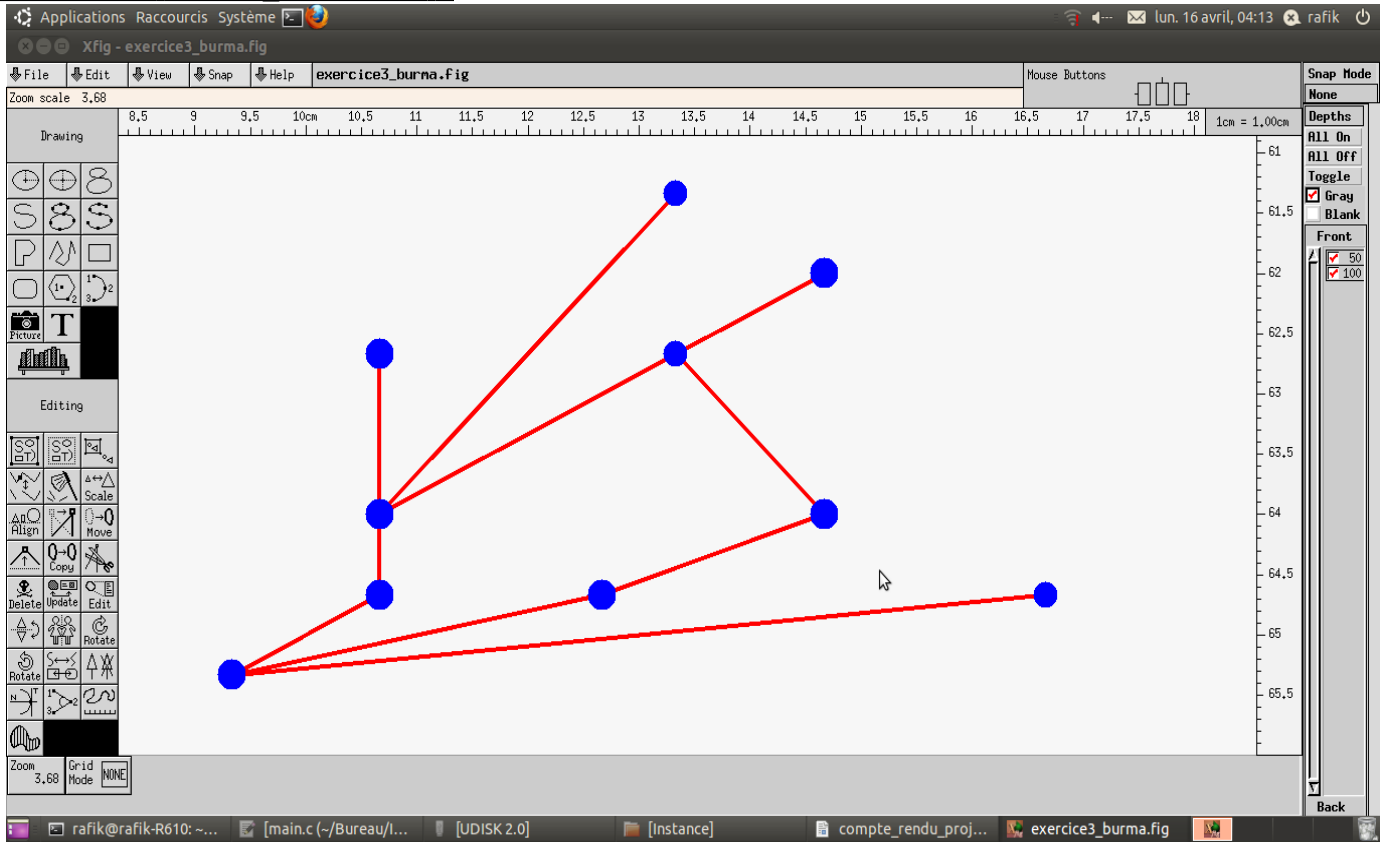
```
1 c NbNodes: 10
2 c NbCables: 9
3 c Gamma: 4
4 c
5 v 5 14.05 98.12
6 v 0 16.30 97.38
7 v 1 16.47 94.44
8 v 2 16.47 96.10
9 v 7 19.41 97.13
10 v 9 20.09 92.54
11 v 8 20.09 94.55
12 v 6 22.00 96.05
13 v 3 22.39 93.37
14 v 4 25.23 97.24
15 c
16 a 5 7
17 a 0 5
18 a 0 1
19 a 1 2
20 a 2 9
21 a 2 3
22 a 6 8
23 a 6 7
24 a 4 5
25 c
26 k 9 4
27 k 3 5
28 k 4 2
29 k 8 5
30 k 2 3
31 k 6 5
32 k 4 0
33 k 0 3
```

instance n°2 : « 00014_burma.res » ==> burma_test.fig:

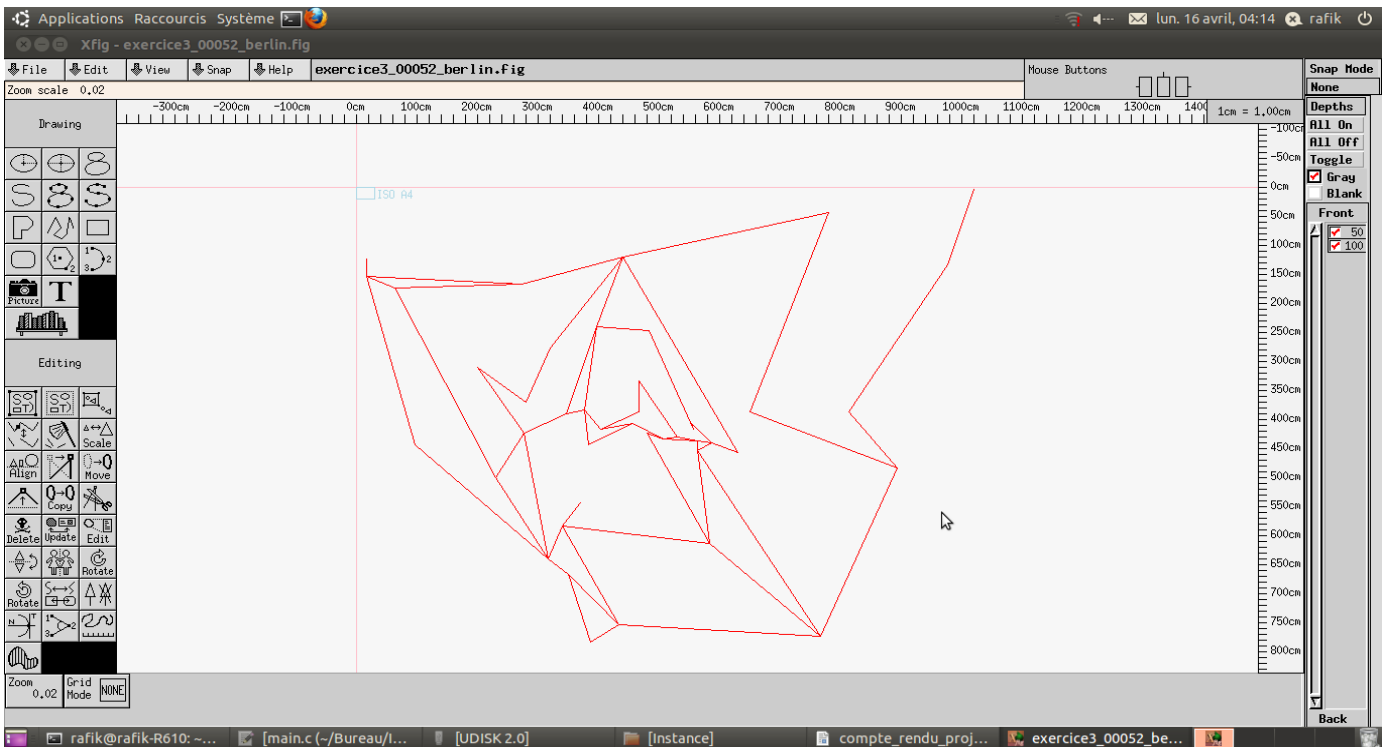


Exercice 3 : les fonctions sont dans les fichiers : exo3.c et exo3.h
construire le reseau a partir de la liste des chaines.
Dans cette partie la lecture et l'écriture d'une chaine est toujours au debut.

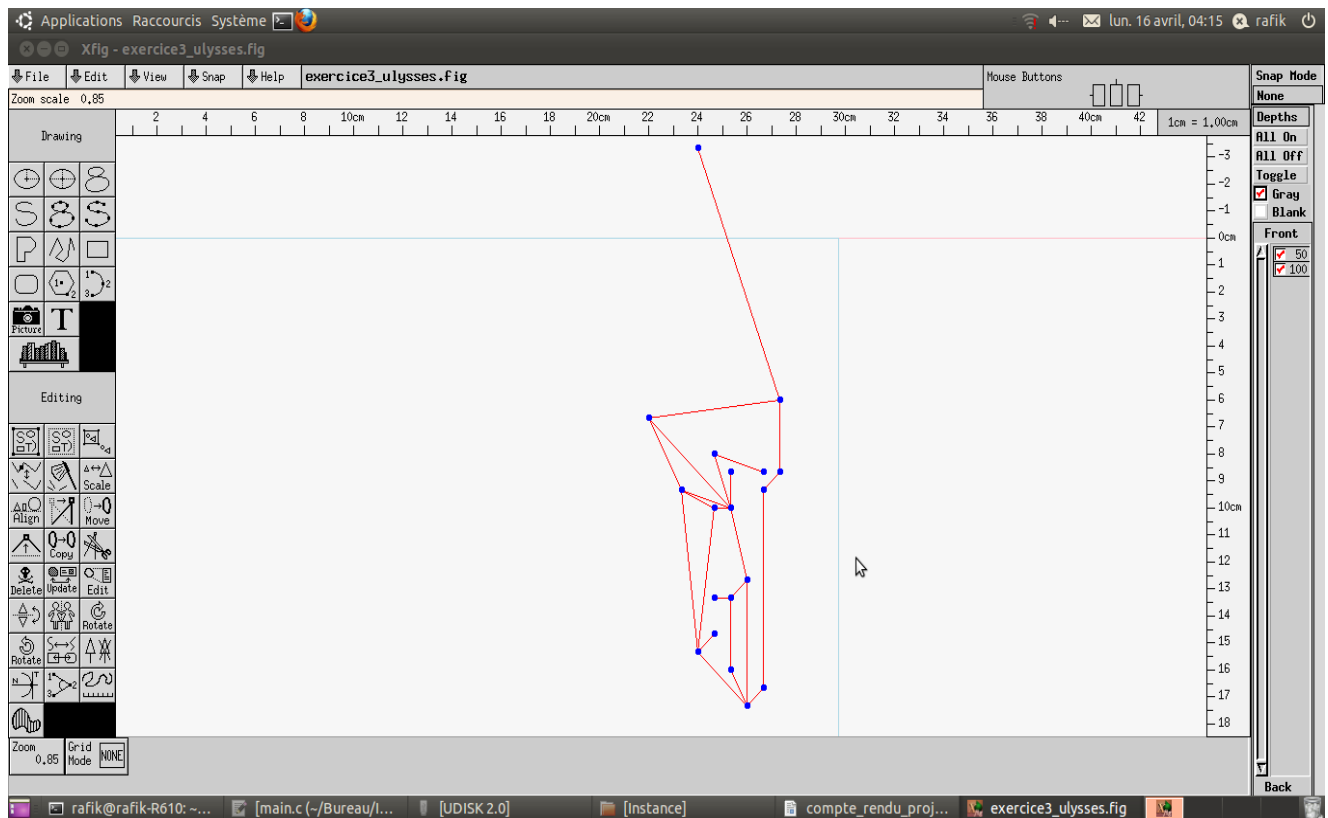
Teste numero 1 : 00014 burma.cha :



teste numero 2 : 00052 berlin.cha



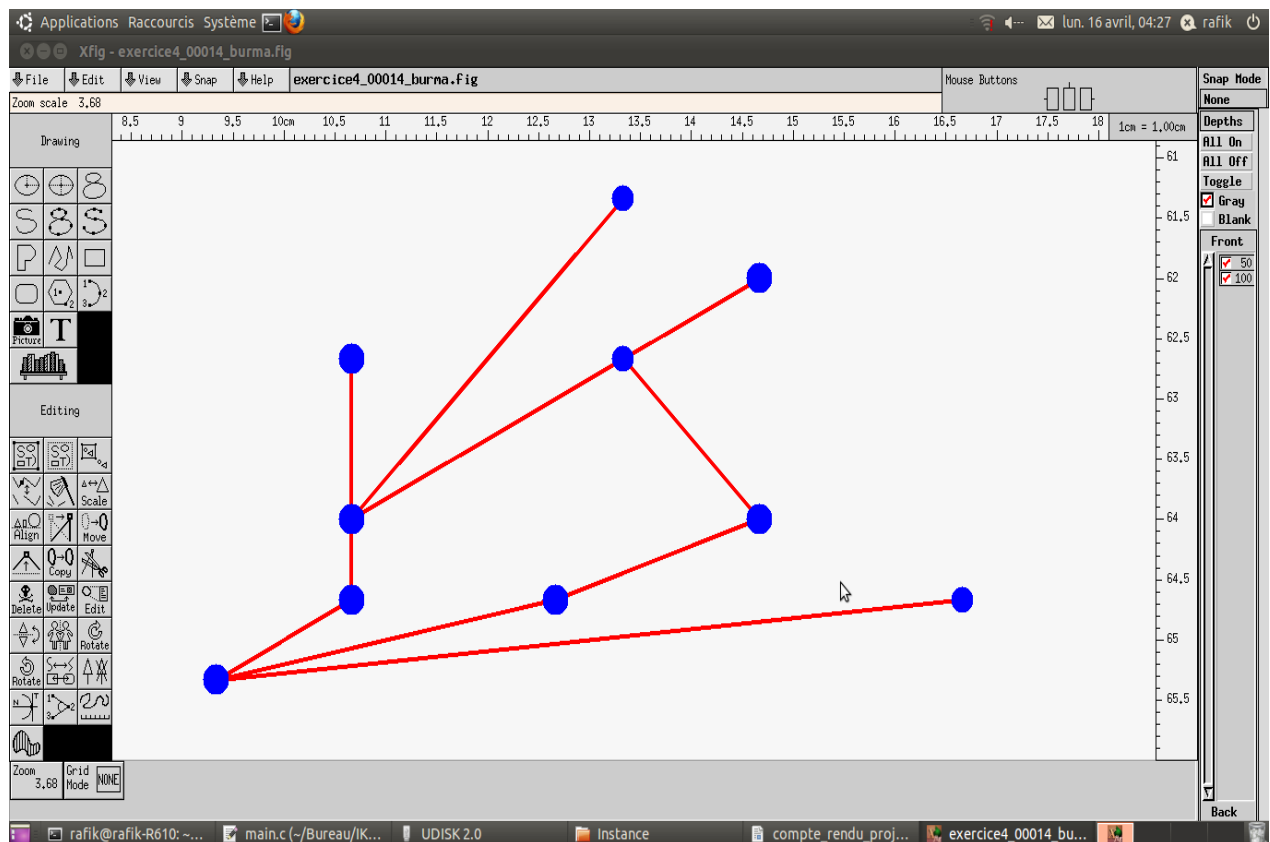
teste numero 3 : 00022_ulysses.cha



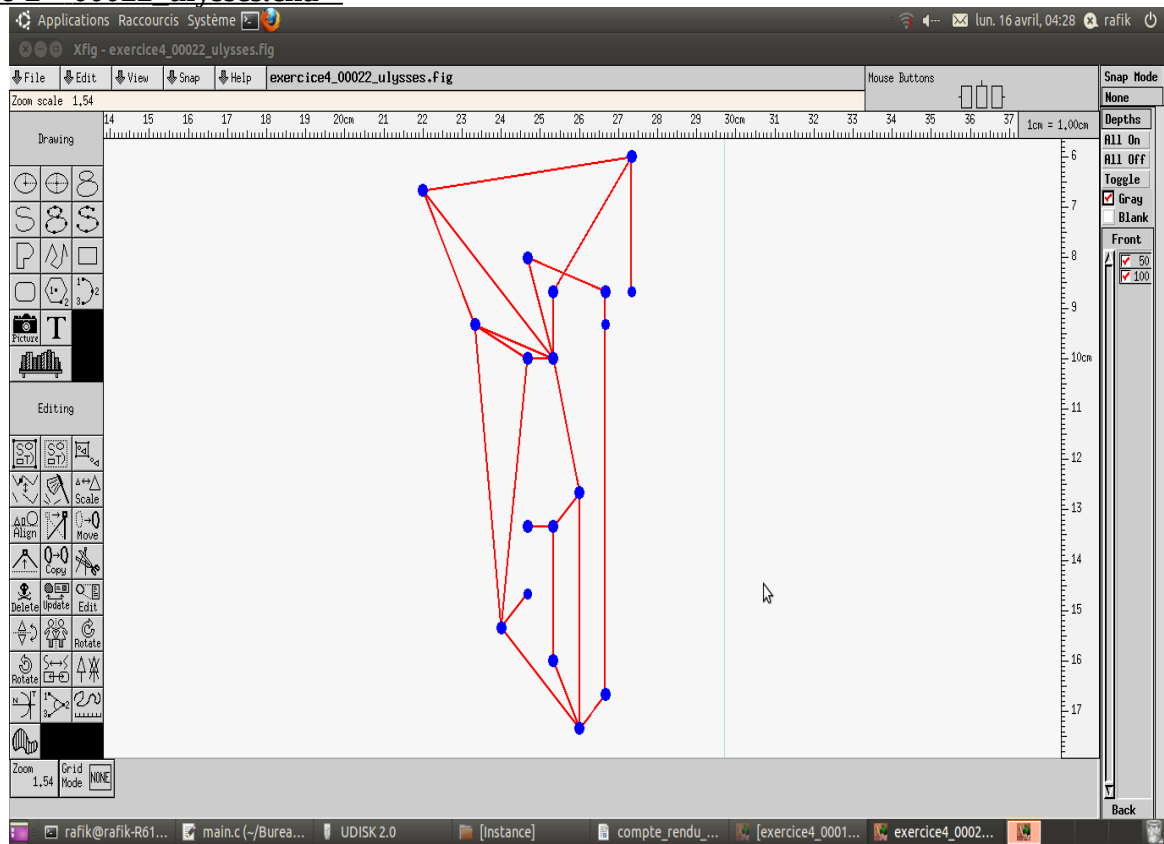
Exercice 4 : exo4.c exo4.h

cette fois ci on utilise une table de hachage pour definir le reseau.

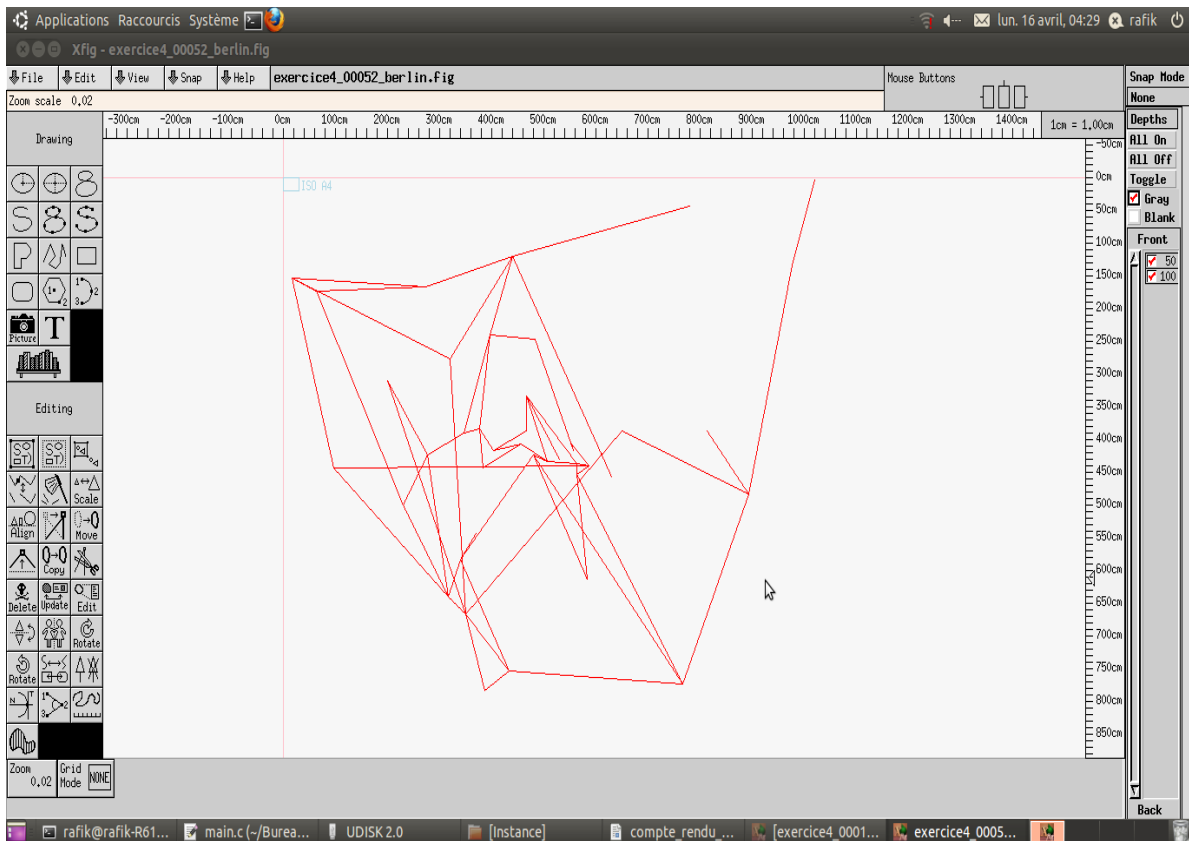
Teste numero 1 = 00014_burma.cha :



teste numero 2 « 00022_ulysses.cha »

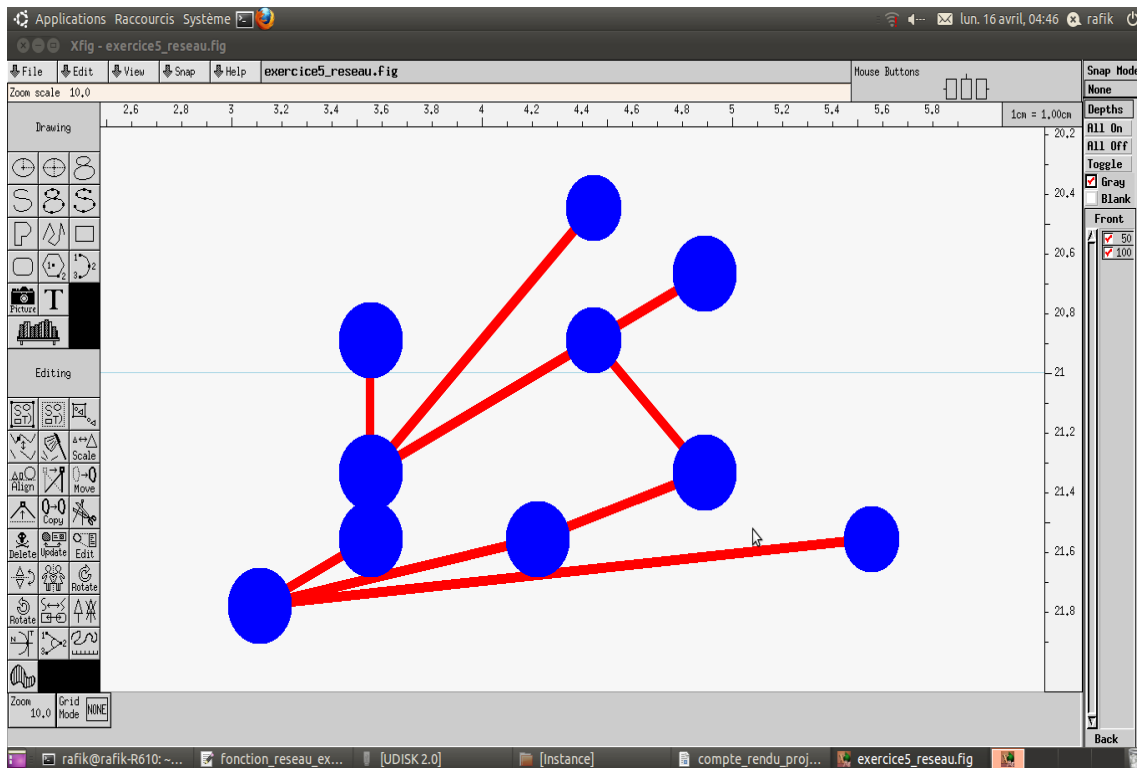


teste numero 3 :00052_berlin.cha



exercice 5 les fonctions et la structure de l'avl sont dans le fichier ArbreAVL.h
on construit le reseau a partir de la liste chaine, on utilisant un AVL

Teste numero 1 = 00014 burma.cha :



exercice 6

1 – structure d'un element du tas :

```
typedef struct element{
    int numero;
    double clef;
}Element;
```

-structure d'un tas :

```
typedef struct tas{
    Element *tab[n+1];           // structure du tas , tableau d'elements de taille taille
    int taille;
}Tas;
```

quelques exemples :

taille du tas : 5

l'element du tas:(4, 2.90)

l'element du tas:(1, 5.30)

l'element du tas:(2, 23.67)

l'element du tas:(3, 13.87)

l'element du tas:(0, 10.20)

le min du tas :(4, 2.90)

l'element du tas recherché est :

l'element du tas:(3, 13.87)

exercice 7 et algo de dijkstra

– l'algorithme de Dijkstra sert à résoudre le problème du plus court chemin. Il permet de déterminer le plus court chemin pour se rendre d'une ville à une autre, il s'applique à un graphe connexe (les sommets sont accessibles les uns des autres) dont le poids (distance, temps, valeur...etc) lié aux arêtes (arcs) est positif ou nul.

- le principe :

à partir des données initiales on construit un sous graphe dont lequel sont classés les différents sommets par ordre croissant de leurs distances minimales (la somme des poids des arêtes empruntées) au sommet de départ.

Étape 1 : mettre de côté le sommet de départ, repérer la distance du sommet de départ aux autres sommets du graphe.

Étape 2 : mettre de côté le sommet qui a la plus courte distance au sommet.

Pour tous les autres sommets, on compare la distance trouvée précédemment à celle que l'on obtiendra via le sommet que l'on vient de mettre de côté, on conserve la plus petite valeur, on continue jusqu'à ce qu'il n'y ait plus de sommet à visiter et jusqu'à sélection du sommet d'arrivée.

l'implémentation : - liste d'adjacence vu en cours, on stocke le graphe sous forme de listes d'adjacence et en utilisant un tas pour réaliser la fonction trouver minimum.

– Le graphe possède m arcs et n sommets, les comparaisons des longueurs est constante, de plus le tas est de complexité binomiale, alors la complexité de l'algorithme de Dijkstra est : $O((m+n) \cdot \log(n))$.

graphe : (0, 0)

La liste des sommets

sommet : (5, 14.05, 98.12)

sommet : (0, 16.30, 97.38)

sommet : (1, 16.47, 94.44)

sommet : (2, 16.47, 96.10)

sommet : (7, 19.41, 97.13)

sommet : (9, 20.09, 92.54)

sommet : (8, 20.09, 94.55)

sommet : (6, 22.00, 96.05)

sommet : (3, 22.39, 93.37)

sommet : (4, 25.23, 97.24)

La liste des arêtes

Arête : (0, 5, 0)

Arête : (1, 0, 1)

Arête : (2, 1, 2)

Arête : (3, 2, 3)

Arête : (4, 7, 5)

Arête : (5, 9, 2)

Arête : (6, 8, 6)

Arête : (7, 6, 7)

Arête : (8, 4, 5)

le chemin:

sommet : (4, 25.23, 97.24)

sommet : (5, 14.05, 98.12)

sommet : (0, 16.30, 97.38)

sommet : (1, 16.47, 94.44)

sommet : (2, 16.47, 96.10)

sommet : (9, 20.09, 92.54)

le chemin:

sommet : (5, 14.05, 98.12)

sommet : (0, 16.30, 97.38)

sommet : (1, 16.47, 94.44)

sommet : (2, 16.47, 96.10)

sommet : (3, 22.39, 93.37)
le chemin:
sommet : (2, 16.47, 96.10)
sommet : (1, 16.47, 94.44)
sommet : (0, 16.30, 97.38)
sommet : (5, 14.05, 98.12)
sommet : (4, 25.23, 97.24)
le chemin:
sommet : (5, 14.05, 98.12)
sommet : (7, 19.41, 97.13)
sommet : (6, 22.00, 96.05)
sommet : (8, 20.09, 94.55)
le chemin:
sommet : (3, 22.39, 93.37)
sommet : (2, 16.47, 96.10)
le chemin:
sommet : (5, 14.05, 98.12)
sommet : (7, 19.41, 97.13)
sommet : (6, 22.00, 96.05)
le chemin:
sommet : (0, 16.30, 97.38)
sommet : (5, 14.05, 98.12)
sommet : (4, 25.23, 97.24)
le chemin:
sommet : (3, 22.39, 93.37)
sommet : (2, 16.47, 96.10)
sommet : (1, 16.47, 94.44)
sommet : (0, 16.30, 97.38)
////////////////////////*****fin*****////////////////////////////////

- Compilation et Execution : voir le Makefile
- 1 - il suffit de se positionner dans le repertoire courant
 - 2- make
 - 2- ./main
 - 3- make clean : effacer les *.o et le main
 - 4- make PHONY : effacer les *.res , *.cha et *.fig