ENSIAS

BOUKHEROUAA Taha Groupe 2

OUDAOUD El Mehdi Groupe 5

Encadré par : Mr. Hassounny

REMERCIEMENTS

2019-2020

On tient à remercier toutes les porsonnes qui ont contribué au succés de ce projet et qui nous ont aidé lors de la réalisation de ce projet.

On voudrait dans un premier temps remercier Mr.ELHASSOUNY qui a eu le soin d'examiner et encadrer ce travail et ainsi nous offrir une véritable opportunité d'apprentissage.

Nous aimerions aussi gratifier les efforts de nos professeurs des modules de « Structures de données » : Monsieur Abdellatif EL FAKER , et de « Techniques de programmation » : Monsieur Hatim GUERMAH pour leur disponibilité, leur savoir-faire et les méthodes de programmation et raisonnement qu'ils nous ont transmis.

ABSTRACT

2019-2020

The DNA sequence contains necessary informations of living beings to survive and to breed. Establishing this sequence is also usefull for researchs on how organisms live. In medecine, it can be used to identify, diagnose and potentially find treatments to genetic diseases.

In this subject we look to assemble a DNA from an FQ file created using global sequencage method (or Shotgun), also to look if it contains a bacterial strain (obtained from a database).

PLAN

Chapitre 1 : CADRE GENERAL ET CONCEPTION DU PROJET :

I. Introduction

II. Cahier de charge

III. Analyse et conception

Chapitre 2 : MISE EN ŒUVRE ET REALISATION :

I. Outils utilisés

II. Exécution

Chapitre 3 : Bibliographie et conclusion :

CHAPITRE 1 : CADRE GENERAL ET CONCEPTION DU PROJET

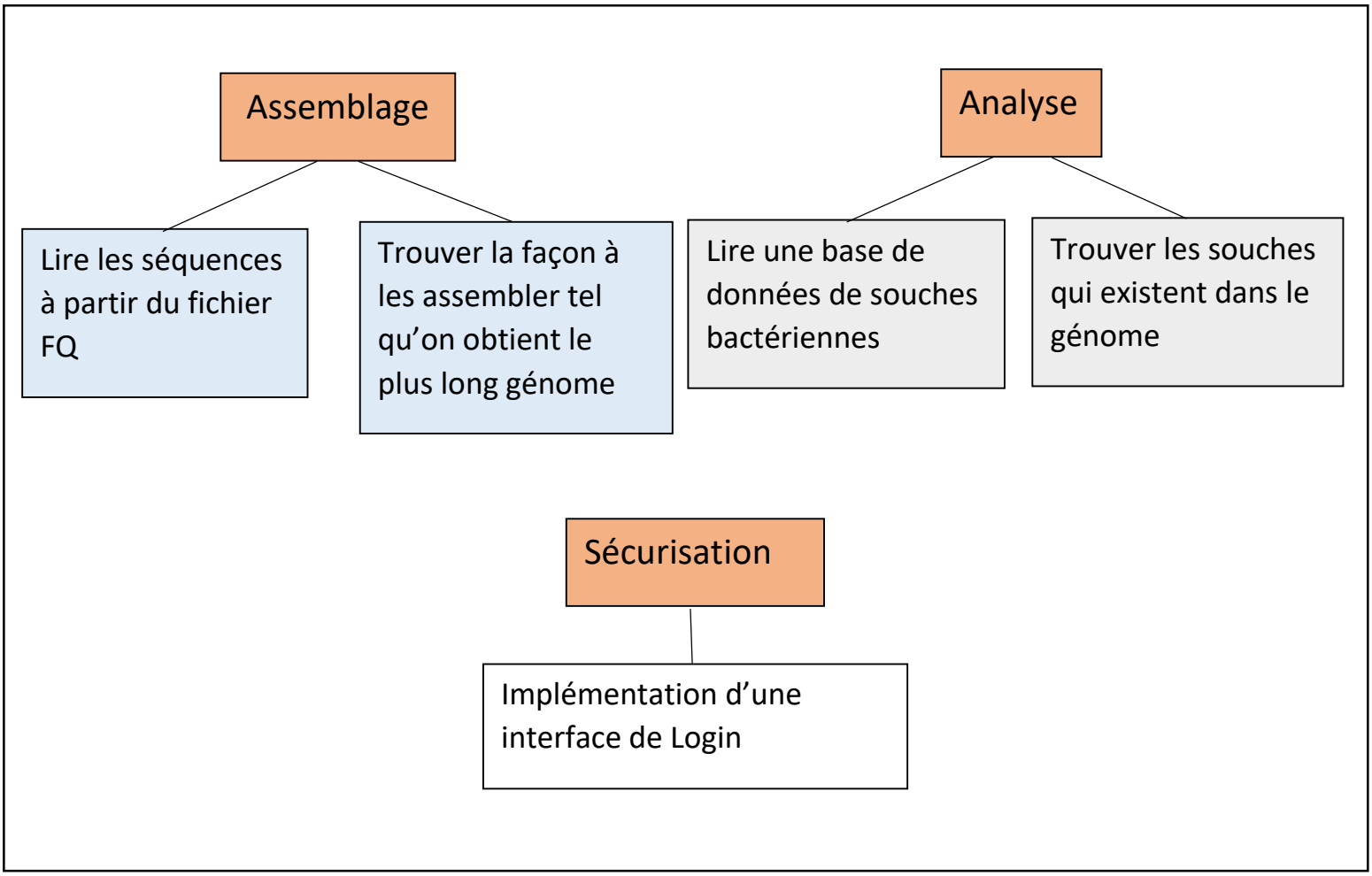
Dans ce chapitre on parlera des méthodes utilisées pour assembler l'ADN notamment la méthode « shortest superstring using greedy setcover », et La méthode de « De BRUIJN » avec une implémentation de correcteur d'erreur de lecture. Ensuite analyser

I. INTRODUCTION :

La séquence d'ADN contient l'information nécessaire aux êtres vivants pour survivre et se reproduire. Déterminer cette séquence est donc utile aussi bien pour les recherches visant à savoir comment vivent les organismes que pour des sujets appliqués. En médecine, elle peut être utilisée pour identifier, diagnostiquer et potentiellement trouver des traitements à des maladies génétiques.

Dans ce sujet on cherche à faire l'assemblage de l'ADN à partir d'un fichier FQ crée à l'aide de la méthode de séquençage globale (ou Shotgun), ainsi que chercher s'il contient des souches bactériennes (à partir d'une base de données).

II. CAHIER DE CHARGE



III. ANALYSE ET CONCEPTION :

La conceptualisation du projet n'a pas était facile vu que les methodes d'assemblage sont très nombreuse cepandant nous avons decider de nous focaliser sur : une approche naive avec la méthode « Greedy Set Cover » qui cherche à trouver la meilleur concatènation possible entre tout element éxistant des le fichier des séquences (.fq) et refaire cette procédure jusqu'à avoir créer l'ADN (avec une marge d'erreur).

Figure : Fichier contenant l'ensemble de lectures obtenusà partir d'un séquenceur

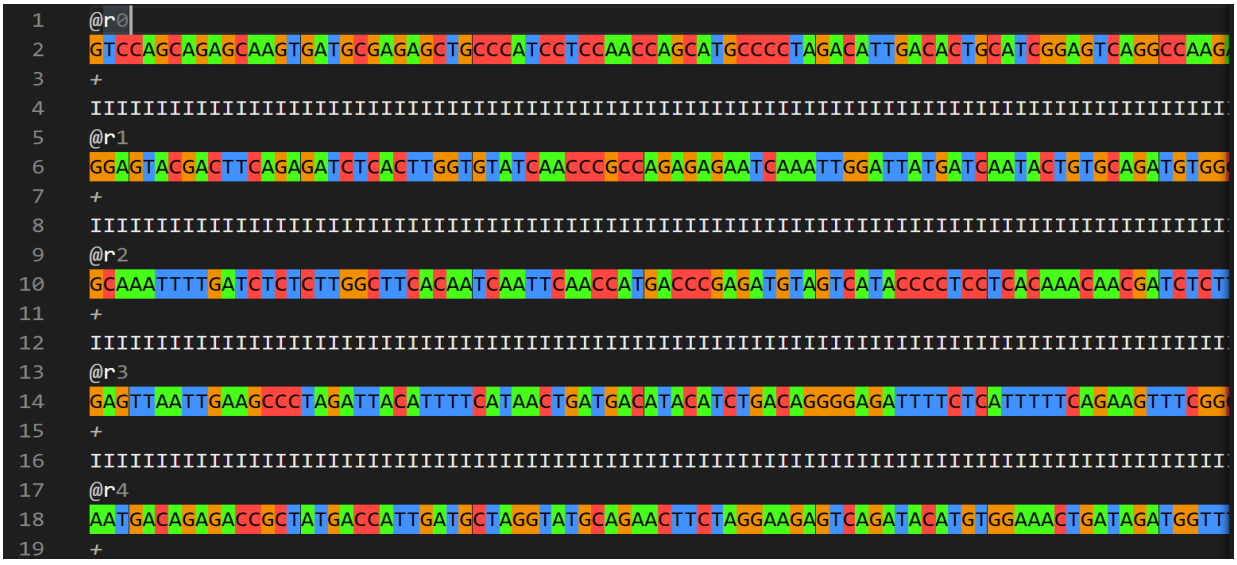


Figure : Base de données des pseudo-souches bactériennes

Cepandant cette méthode s'avère assez complexe en terme de temps (NP-hard ou complexité exponentiel) ,Ainsi on a penser a implementer une methode plus avancé (utilisé dans les assembleurs modernes) . C'est la méthode des graph de Debruijn.

Et puis après avoir assembler le genome on va chercher l'analyser détéctant ainsi les souches bactèriennes qui y réside.

Pour accomplir cette tache on a penser à créer une base de données de souches bactériennes (virus pour simplifier).

Analyse fonctionnelle : Greedy set cover

STRUCTURE	UTILITE
Element	C'est un nœud d'une liste chainée composée d'un ensemble de chaines de caractère (représente l'ensemble des substrings).
Liste	C'est un pointeur sur la tete de la liste chainée d' « Element ».
Tuple	Représente un nœud d'une combinaison entre de chaine de caractère différentes (sera utile pour trouver le superstring le plus long)
Dictionnaire	C'est un pointeur sur la tete de la liste chainée de « Tuples »

L’implementation de cette methode est simple :

Trouver le plus grand « overlap » (GTA et TAC on pour overlap 2) entre deux lecture du fichier.fq et les concaténer en respectant le principe (CAAT+ATGG = CAATGG) et sauvegarder le résultat enfin.

Fonction	Utilité
initialisation_liste	Initialise une liste d’ « élément » et la renvoie
initialisation_dico	Initialise une liste de « Tuple » et la renvoie
insertion_liste	Insère (brutement) un « element » dans la liste à partir d’une chaine de caractère.
insert_dico	Insère (sans répition*) un « Tuple » dans le dictionnaire à partir de (Gauche,Droit) .
Recherche_liste	Cherche si une chaine éxiste dans la table d’éléments.
lire_sequence	Lit le fichier de format spéciale (.fq)
chargement_input	Convertit les lectures dans .fq en des éléments dans une liste.
Find_overlap_start	Trouve l’element à parti de quelle on a overlap sinon retourne -1
subString	Retourne une sous_chaine qui commence en début et fini en fin.
overlap	Calcule l’overlap entre deux chaine.
permutations	Retourne un Dictionnaire contenant toute combinaison possible de séquence
Overlap_maximale	Chercher dans un Dictionnaire la plus grande valleur d’overlap entre gauche et droit
Greedy_setcover	L’aplication de l’algorithme
Sauvegarde_superstring	Sauvegarde le superstring dans un fichier

MALHEUREUSEMENT CETTE METHODE EST TRES INEFFICACE (ON NE PEUT PAS VRAIMENT ASSEMBLER LE GENOME EN ENTIER).

Analyse fonctionnelle : De BRUIJN Graphs

Implemontons une structure de graphe :

STRUCTURE	UTILITE
DBN_noeud	Représente un nœud du graphe de De Bruijn
arc_list	Représente une liste des arcs du graphe de De Bruijn
Nœud_list	Représente une liste des nœuds du graphe de De Bruijn
DBN_GRAPH	Représente le graphe de De Bruijn

Cette méthode passe par 3 phases clefs :

1- Création du graphs à partir k-1 mers de chaque read (voir documents) .
2- Correction d'erreurs de lecture commise pas le séquenceur ainsi que la correction de la topologie du graph(assurer l'unicité du chemin eulerien).
3- Parcours du graph à l'aide de l'algorithme de parcours eulerien.

Fonction	Utilité
create_noeud	Créer un nœud
init_DBN_graph	cette fonction recherche un str dans le graph et retourne le sommet contenant ce str
add_to_universe	Ajoute un sommets à l'ensemble des sommets
read_seq	Lire le .fq
append_arc	Ajoute un arc entre deux nœud

DBN_graph_builder	Créer le DBN graph
calculer_moyenne_occurences	Calcule le nombre d'occurrence moyen des sommets distinger les sommets erroné (moins frequent).
is_brother	Affirme si deux chaine ne diffère que pas un seul nucléotide (taux d'erreur suposée 1% au max)
arc_sortant_list_union	Fait l'union de deux liste d'arc_sortant
arc_entrant_union	Fait l'union de deux liste d'arc_entrant
replace_noeud	Remplace un nœud moins fréquent par son frère plus frèquent
corriger_erreur_type_read	Implementation de la correction d'erreur (remplacement de neoud)
read_chemin_eulerien	La recherche du chemin eulerien
chemin_eulerien_makeNread	Le parcours du chemin eulerien et sauvegarde du superstring

Analyse fonctionnelle : Recherche de souche malade

Il suffit de chercher si une souche est une sous-chaine du genome assemblé

Fonction	Utilité
recherche_souche_super	Cherche si une souche est une sous-chaine du genome.
read_adn	Lit le fichier assemblé (ADN)
recherche	Fait le parcours toute les souches dans la base de données virus et affirme s'ils éxistent où non

Analyse fonctionnelle : Securité

Pas une grande sécurité car le login et le mot de passe sont « hard codé » mais le procèdure d'authentification marche très bien

Login : oudaoud

Mot de passe : taha123

CHAPITRE 2 : MISE EN ŒUVRE ET REALISATION

Dans ce chapitre on parlera des outils informatiques et logiciels utilisées pour accomplir ce projet ainsi qu'une simulation de l'execution du programme

I. OUTILS UTILISES

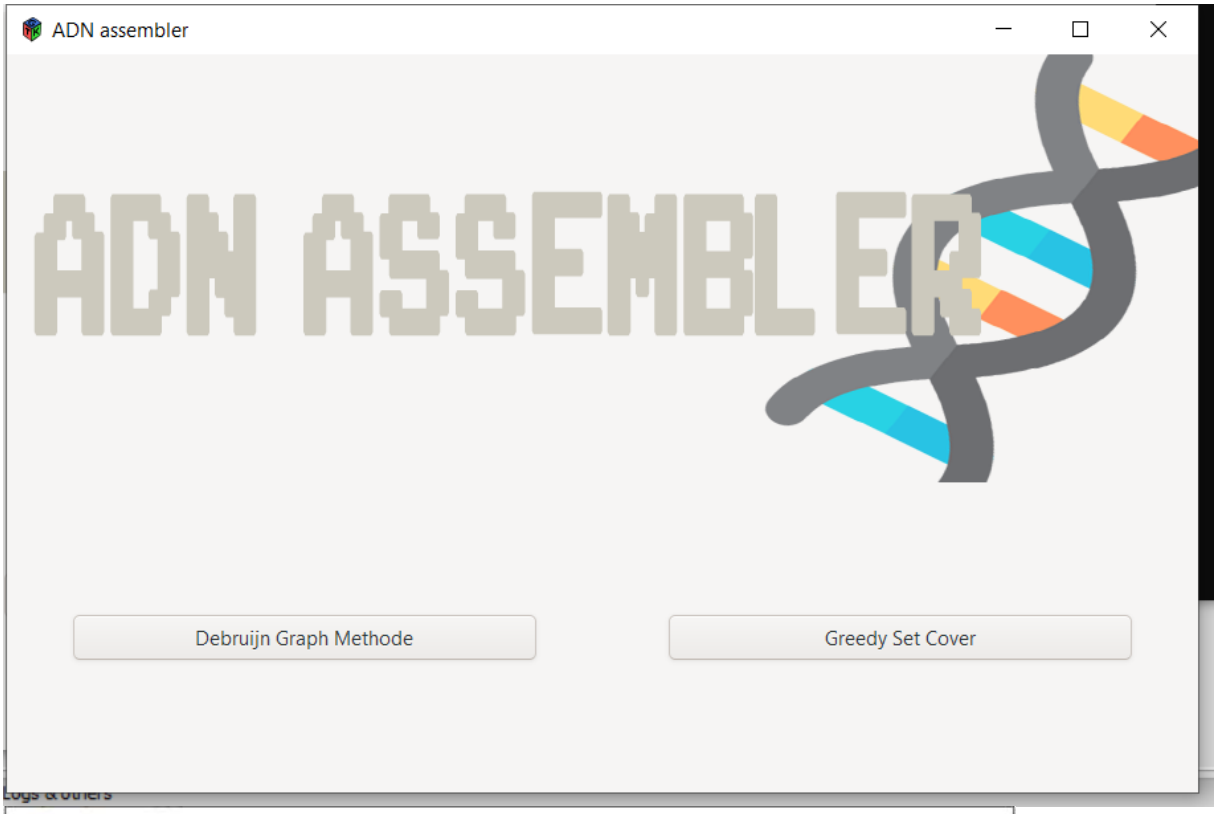
Afin de réaliser ce projet on a compté sur plusieurs outils informatiques autre que le language C dont :

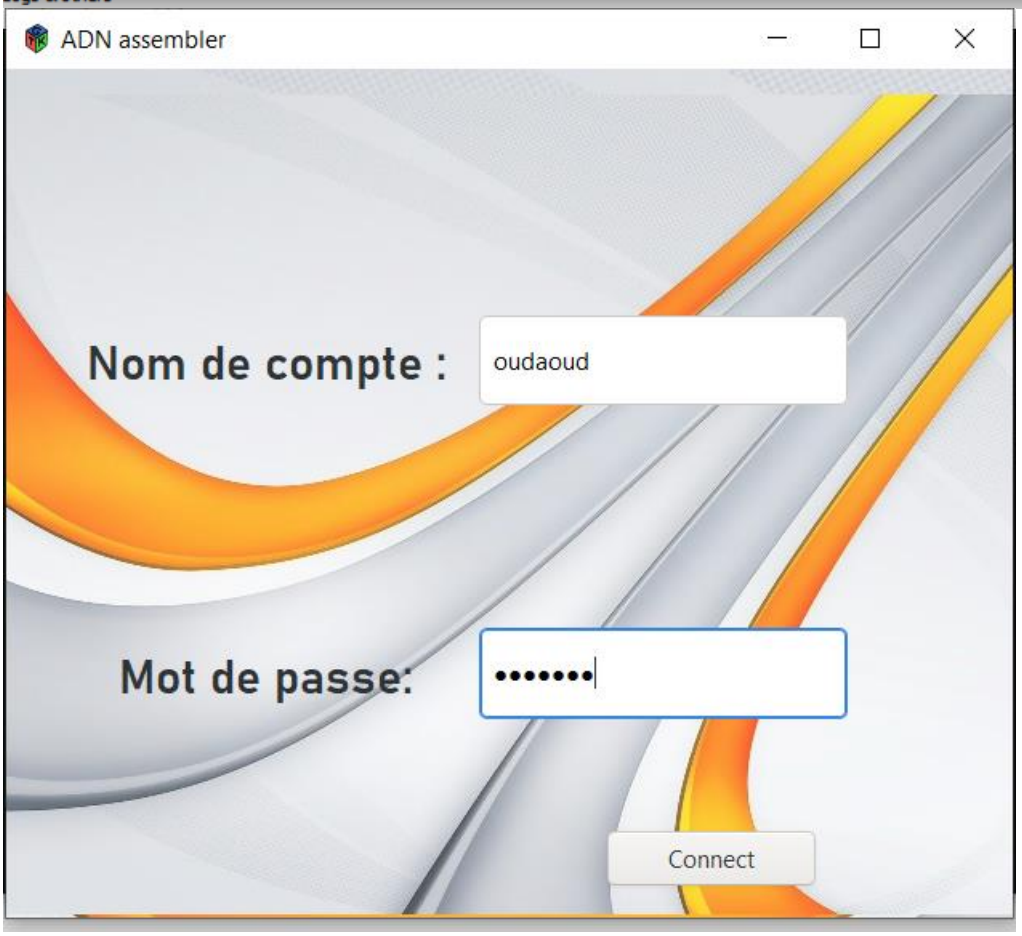
Le langage python pour génerer les souches bactériennes

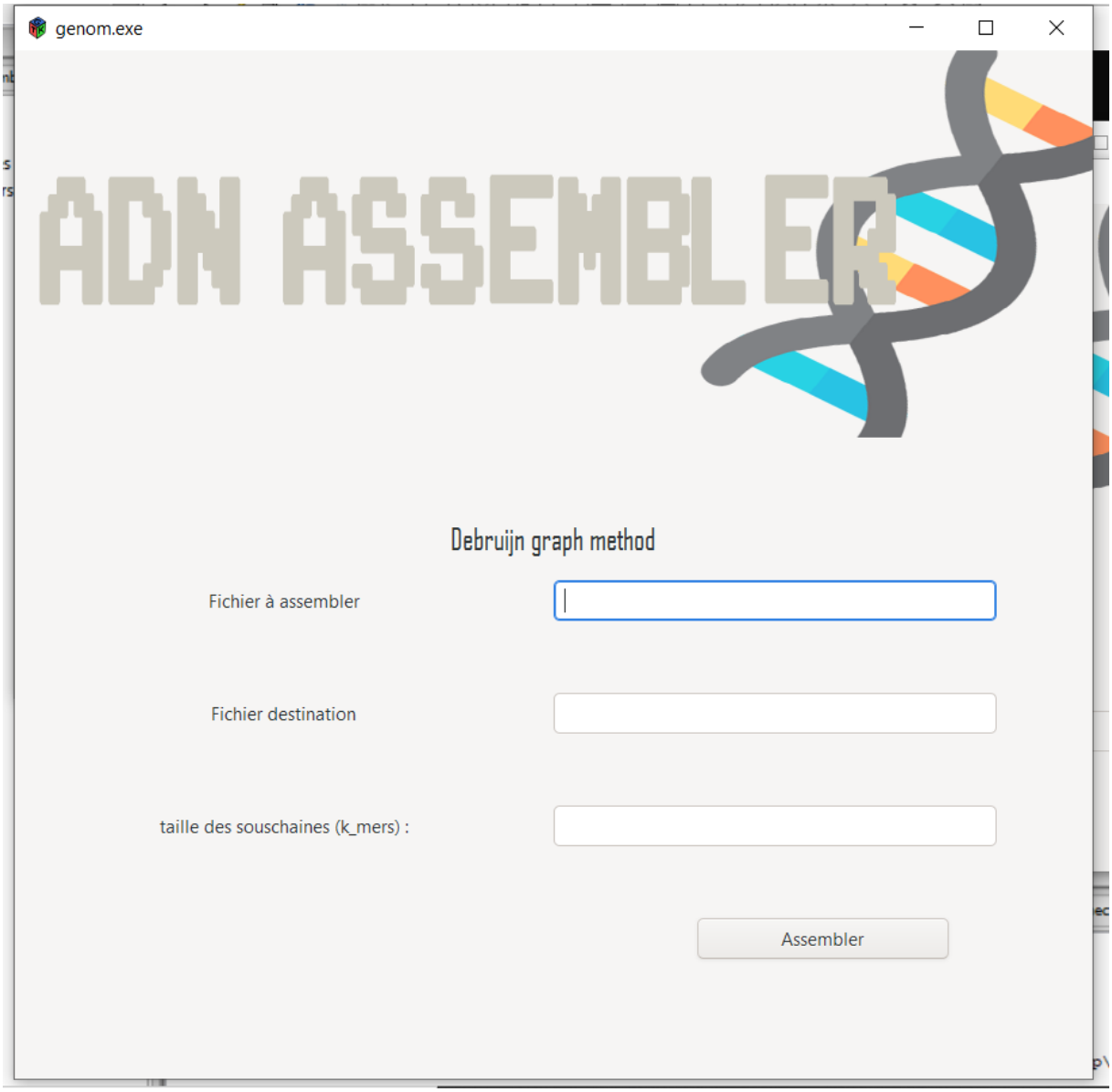
La bibliothèque GTK pour l'interface graphique.

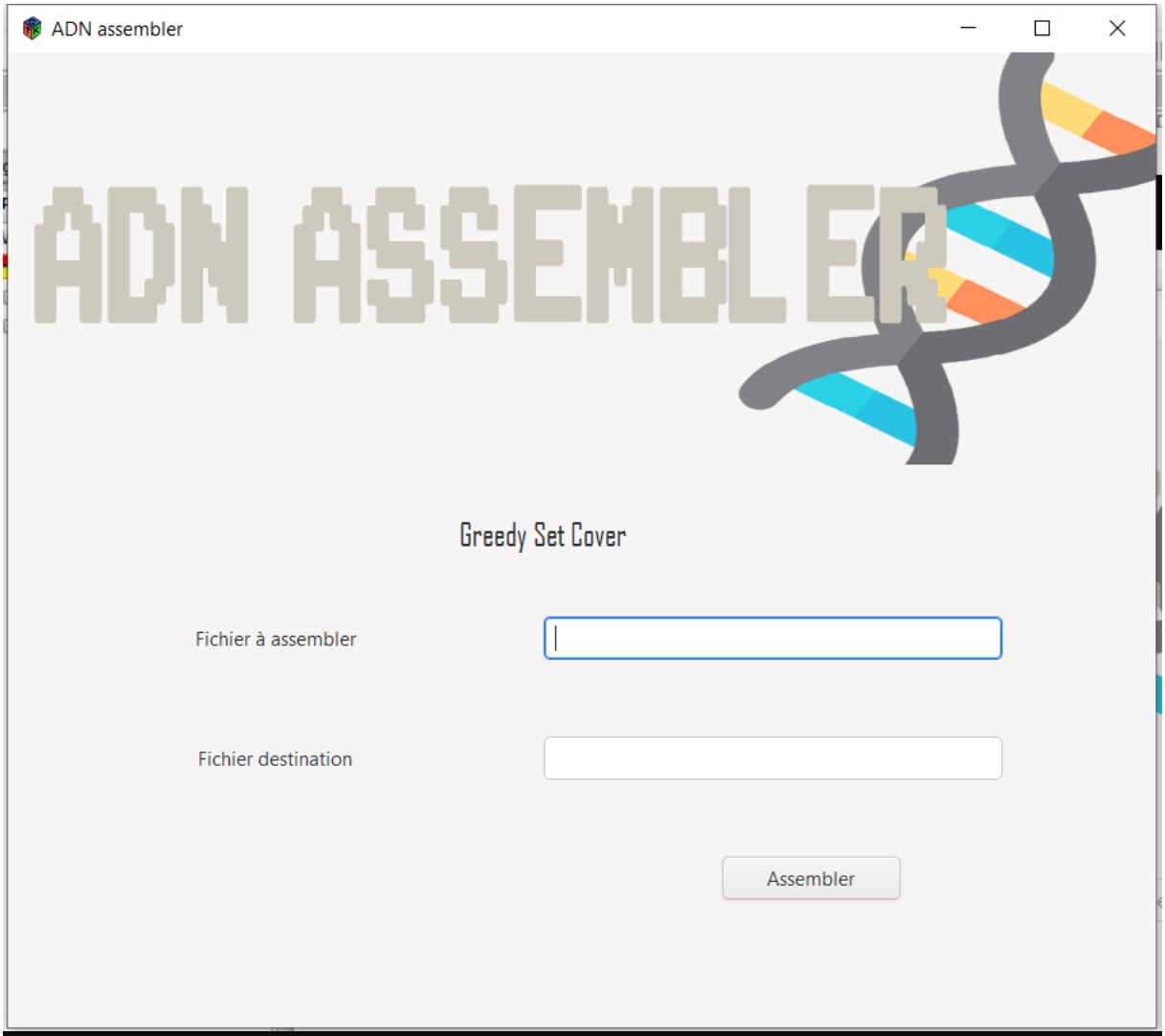
Code Blocks pour la compilation et linkage du projet.

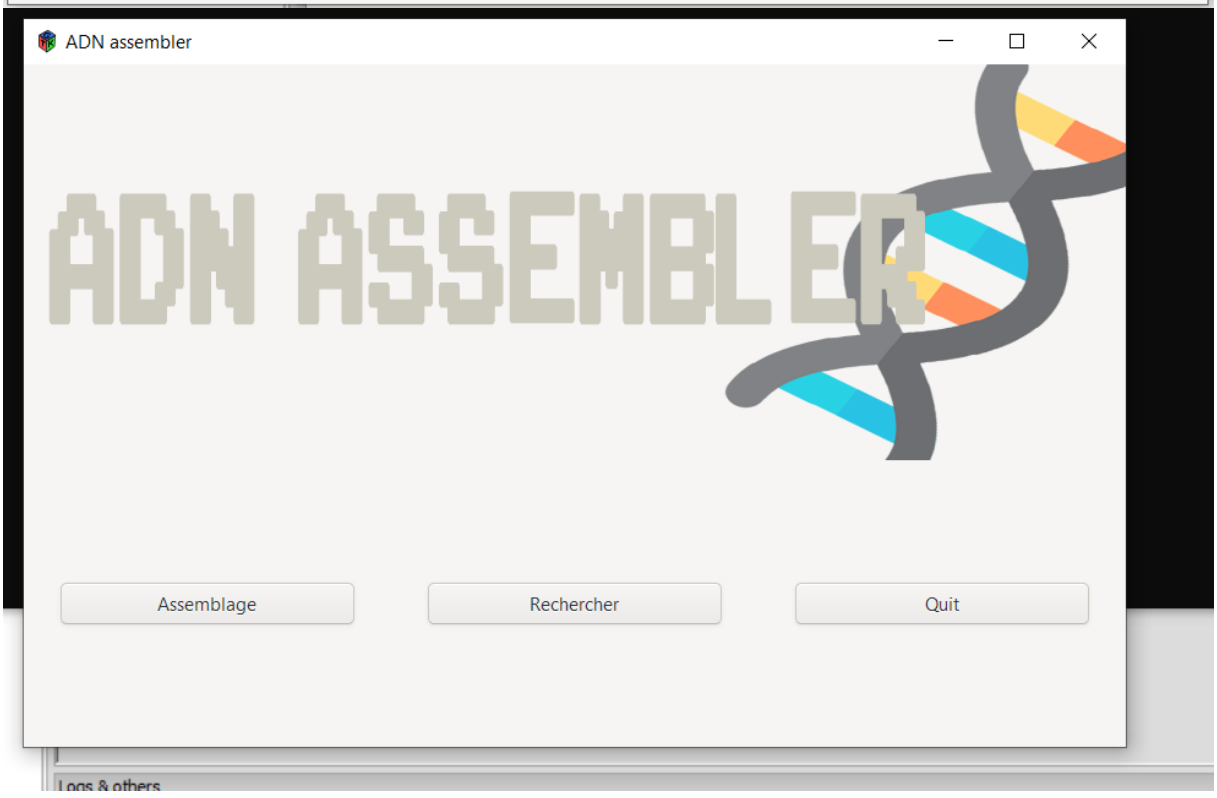
II. EXECUTION



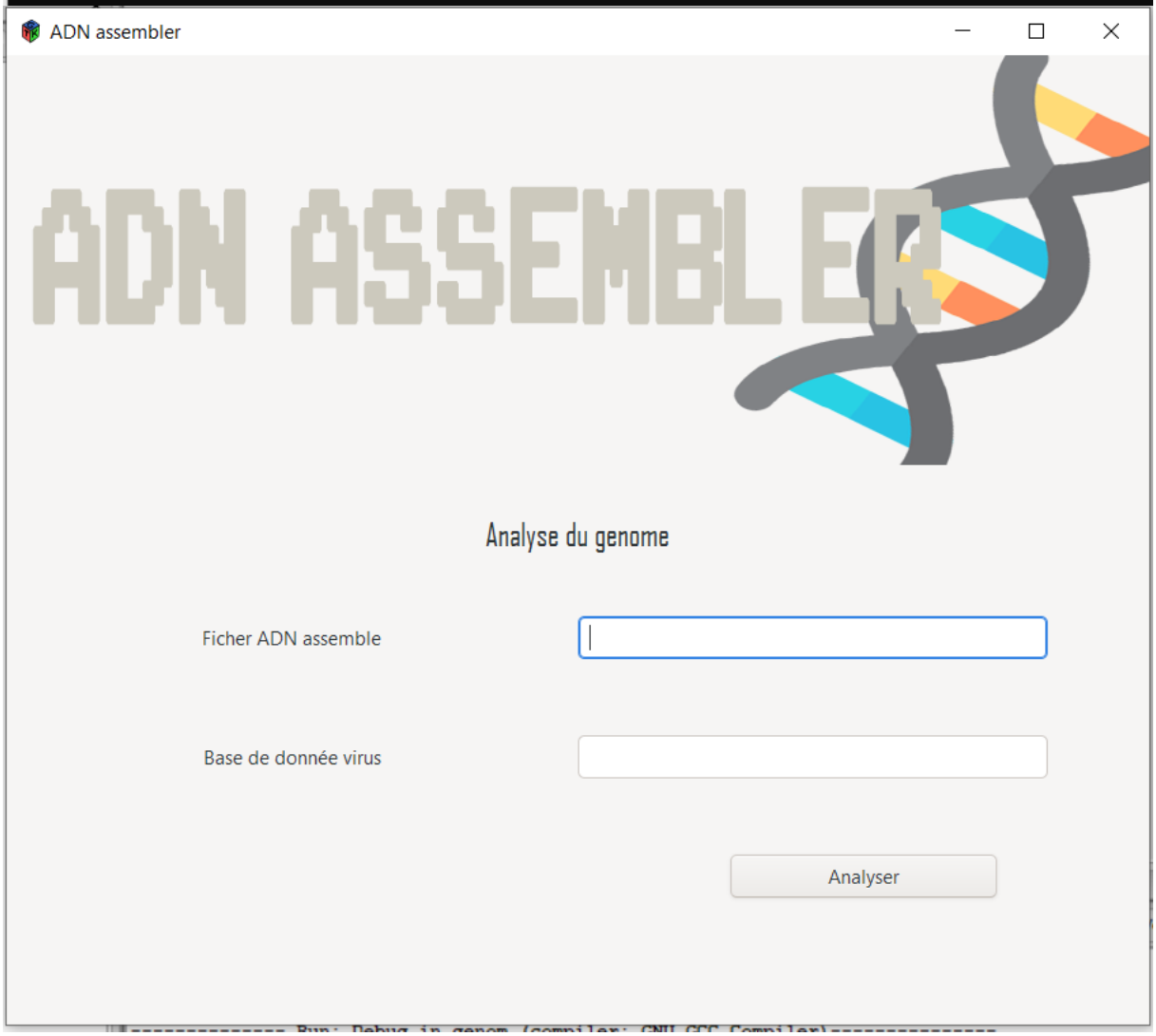



genom.exe
ADN ASSEMBLER
Debruijn graph method
Fichier à assembler
Fichier destination
taille des souschaines (k_mers) :
Assembler


ADN assembler
ADN ASSEMBLER
Greedy Set Cover
Fichier à assembler
Fichier destination
Assembler


ADN assembler
ADN ASSEMBLER
Assemblage
Rechercher
Quit

```
Le motif: GGAGATCCACGAGCTCCTGAAACTCCAATCCA
Le motif: TCCACGAGCTCCTGAAACTCCAATCCAGAG
Appuyez sur une touche pour continuer...
```



Fichier Edition Format Affichage Aide

GGAGATCCACGAGCTCCTGAAACTCCAATCCAGAGGCAACAACTTTCCGAAGCTTGG
GCAATTGTGGGAGGCGCAGTTAGTAGAGGTGATATCAACCCTATTCTGAAAAAACTTA
TCATACCCCTCCTCACAAACAACGATCTCTTAATAAGGATGGCACTGTTGCCAGGAG
ACTGATAGATGGTTTCTTCCCTGCACTCGAGTTAATTGAAGCCCTAGATTACATTTT