

**RAPPORT STRUCTURES DE DONNÉES**

Dylan EDWARD PANKIRASA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jérémy BRAKHA

2021-2022

# Phases du projet

Pour une raison de temps, nous n’avons effectué que les trois premières implémentations et la lecture du fichier. Nous en présenterons les résultats.

La réalisation du programme nous servant à connaître les temps de calcul va suivre 3 phases distinctes :

* Phase de conception
  + Prérequis : analyse du sujet et du contexte, connaissances des besoins
  + Contenu : définition des fonctionnalités, analyse technique du projet, le choix de l’architecture et des outils techniques, définition des données du programme
* Phase de développement
  + Prérequis : définition et acceptation des spécifications fonctionnelles du programme
  + Contenu : réalisation des spécifications fonctionnelle, création des différents packages, classes et des différents éléments vus au cours de la phase de conception
* Phase de test
  + Prérequis : définition des tests à réaliser et des résultats attendus
  + Contenu : déploiement des méthodes nous permettant de vérifier nos résultats

# Architecture

Pour résoudre notre problème, nous avons créé plusieurs packages :

* Ensemble : ce package contient les différents ensembles correspondants aux différentes implémentations des structures de données. Par exemple, pour la première implémentation, un ensemble contient un élément correspondant à une liste chainée simple et un représentant.
* Liste : contient les classes servant à implémenter les ensembles. Dans le cas de la première implémentation, nous avons une classe Nœud représentant contenant une valeur et un pointeur vers un nœud suivant. Nous nous servons de cette classe pour implémenter une seconde classe pour ListeChaineeS qui correspond à un élément d’un ensemble vu précédemment.
* Models : regroupe nos classes Arete et Graphe. Une arête possède deux sommets et un poids. Un graphe contient une liste de sommets et une liste d’arêtes.
* Unionfile : classe regroupant les différéntes méthodes nous permettant d’unir les les ensembles.
* Utils : le fichier .txt d’un graphe et la classe nous permettant de lire ce fichier de résoudre le notre problème.

# Les temps de calcul

**Pour un graphe peu dense de taille 10 :**

Une image contenant texte

Description générée automatiquement

Figure : première implémentation

Une image contenant texte

Description générée automatiquement

Figure : seconde implémentation

**Pour un graphe peu dense de taille 100 :**

Une image contenant texte

Description générée automatiquement

Figure : première implémentation

Une image contenant texte

Description générée automatiquement

Figure : seconde implémentation

**Pour un graphe peu dense de taille 1000 :**

Une image contenant texte

Description générée automatiquement

Figure : première implémentation

**Pour un graphe très dense de taille 10 :**

**Une image contenant texte

Description générée automatiquement**

Une image contenant texte

Description générée automatiquement

**Pour un graphe très dense de taille 100 :**

Une image contenant texte

Description générée automatiquement

Figure : première implémentation

Une image contenant texte

Description générée automatiquement

Figure : seconde implémentation

Pour des raisons pratiques liées aux performances de nos nos ordinateurs respectifs, nous ne présenterons pas les résultats pour un graphe très dense de taille 1000. Nous avions cependant testé avec succès une première fois l’exécution du programme et constater qu’il nous fallait environ 25 minutes. Nous n’avions pas pris de capture d’écran.

Conclusion des temps de calculs :

A travers ces résultats, nous pouvons constater que les différences de temps de calcul ne s’améliorent qu’à partir de la troisième implémentation. Les résultats des deux premières sont très proches, les temps de la seconde dépassant même la première régulièrement.