Réalisation d'un programme pour apprendre les algorithmes de tri

PROJET DE DEVELOPPEMENT LOGICIEL

MEHDAOUI Abderrahim, ZEISEL Helmut Michael
CENTRALESUPELEC

MEHDAOUI Abderrahim
ZEISEL Helmut Michael

Table des matières

Introduction

Le but de ce projet est de développer un programme permettant à l'utilisateur d'apprendre le fonctionnement d'un algorithme de tri. Il doit également être en mesure de trier un liste d'entiers générée aléatoirement et permettre à l'utilisateur de rentrer sa propre liste afin que cette dernière soit triée.

Un algorithme de tri permet d'organiser une collection d'objets selon un critère donné. Ils sont utilisés dans de très nombreuses situations et particulièrement utiles à de nombreux algorithmes plus complexes dont les algorithmes de recherches comme l'algorithme de recherche par dichotomie. Ils peuvent également servir à mieux organiser les données pour faciliter leur utilisation futurs.

La collection à trier est souvent donnée sous forme de tableau/liste, afin de permettre un accès simplifier aux différents éléments de la collection.

Bon nombre d'algorithmes de tri procèdent par comparaisons successives, et peuvent donc être définis indépendamment de l'ensemble auquel appartiennent les éléments et de la relation d'ordre associée. Un même algorithme peut par exemple être utilisé pour trier des réels et des chaînes de caractères.

Les algorithmes de tri sont souvent étudiés dans les cours d'informatique pour introduire des notions comme la complexité.

Dans ce projet nous somme penchés sur quelques algorithmes connus dont le « tri rapide, » le « tri insertion » et enfin le « tri … » En accord avec les contraintes du sujet les données en entrés seront exclusivement des entiers naturels.

En ce qui concerne le visuel général de l'application, nous souhaitions afficher un premier menu qui permet à l'utilisateur de choisir quel algorithme il souhaite étudier. Une fois l'algorithme sélectionné une nouvelle fenêtre affiche une description de l'algorithme, son code en pseudocode ainsi qu'une animation permettant de comprendre le fonctionnement de l'algorithme via un exemple. L'utilisateur peut choisir de lancer l'animation avec une liste aléatoire ou charger sa propre liste à partir d'un fichier Excel. L'utilisateur peut également comparer deux algorithmes de son choix et les lancer en parallèles pour une liste donnée.

Ce projet s'est articulé autour de deux axes majeurs. Tout d'abord il a fallu développer un premièr modèle pour un algorithme qui soit facilement adaptable pour d'autres. Cette étape consiste surtout à réfléchir sur la présentation générale dudit algorithme mais surtout sur une animation. Enfin le second axe consiste simplement à rajouter d'autres algorithmes en utilisant le modèle trouvé précédemment.

Nous détaillerons donc dans un premier temps notre recherche d'informations ainsi que les outils sélectionnés, puis la structure générale du modèle choisi ainsi que les difficultés rencontrées durant le projet. Nous dresserons enfin un bilan du projet.

Recherche d'informations et sélection des outils disponibles

Cette étape du projet consiste à recueillir des informations sur les algorithmes de tri, les langages pouvant être utilisés ainsi que leurs environnements de développements. Cela nous permet d'avoir un premier aperçu de ces derniers et sélectionner les outils qui seront utiliser durant le développement.

Cette étape s'est déroulée sans encombre, la majorité des informations à connaitre sur les algorithmes a été recueillie via des recherches sur internet afin de retrouver les acquis d'il y a quelques années.

Il nous a semblé utile pour notre apprentissage d'utiliser le langage python, chaque membre du binôme l'ayant utilisés durant les différents travaux de laboratoire au cours de son cursus respectif. Ce projet était l'occasion de faire de la programmation objets en python pour l'un des membres du binôme. Nous avons par ailleurs choisi d'utiliser l'environnement de développement PyCharm pour réaliser ce projet, il semble complet, accessible et reconnu dans le monde académique.

En ce qui concerne les bibliothèques utilisées nous sommes restés dans ce qu'il y a de connu comme Tkinter (ou tkk pour avoir un rendu graphique plus moderne) et d'autres bibliothèques comme numpy, random, time ou threading. Pour traiter les données rentrées par l'utilisateur nous avons choisi d'utiliser la bibliothèque Pandas qui permet de traiter de larges volumes de données, elle semble donc légèrement trop sophistiquée pour le projet mais les fonctions disponibles nous permettent de produire un programme répondant aux normes du sujet.

Structure du code

Nous avons choisi de structurer notre code en utilisant l'architecture modèle-vue-contrôleur.

Le modèle rassemble les classes regroupant les algorithmes (quicksort, insertion sort et bubble) sort La vue regroupe les classes « Display », « Stratpage ». Enfin le contrôleur est composé de la classe « App ».
Ci-dessous un diagramme de classes permettant de mieux comprendre la structure du code et les composantes des classes.

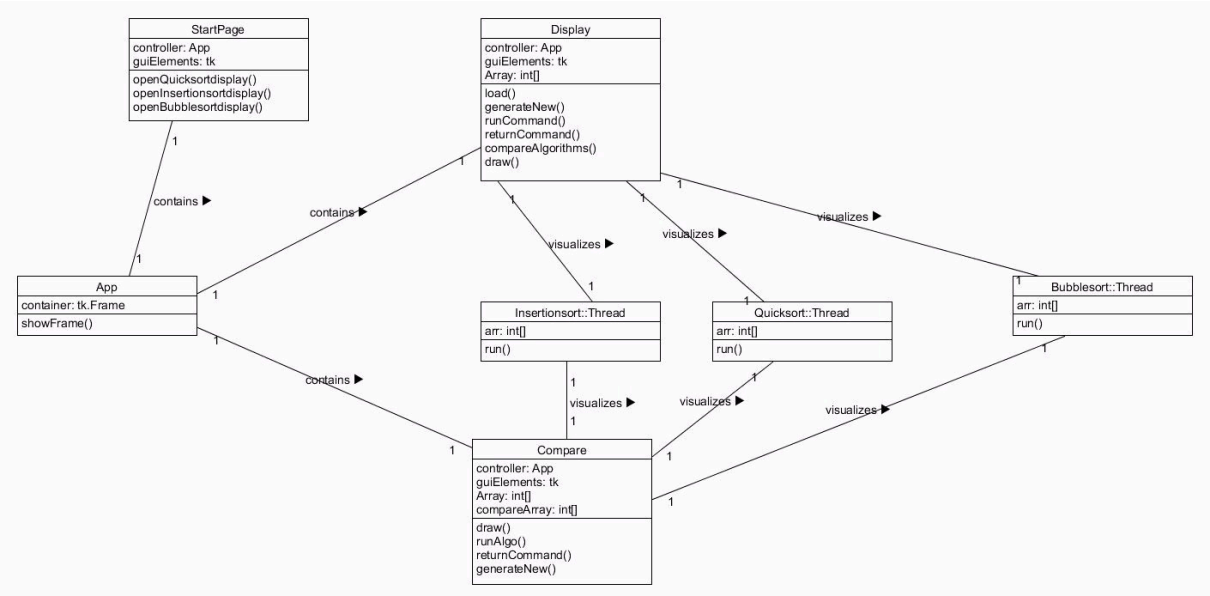

Diagramme de classe représentant l'architecture général du programme

Les classes regroupant les algorithmes :

Elles ne sont constituées que de l'implémentation des algorithmes choisis. Ces classes hérite de la classe thread. En effet, les algorithmes sont implémentés dans des threads séparés en parallèle sinon la boucle dans la fonction draw et les algorithmes se bloquent mutuellement.

Display et ressource.py

Dans cette classe nous utilisons Tkinter afin d'afficher une fenêtre répartie en trois zones. Une comportant les boutons permettant de lancer l'animation à partir d'une liste. La liste est générée aléatoirement grâce à la méthode « generatenew » ou peut être généré à partir d'un fichier Excel grâce à la méthode « load ». Une autre zone comportant une description de l'algorithme ainsi que son pseudocode. La dernière zone affiche l'animation générée grâce à la méthode « draw ».

La méthode « load » permet de générer une liste via Excel, dans cette méthode on s'assure que les données sont bien des entiers avec un boucle for. La méthode « generate new » génère une liste d'entiers aléatoire compris entre 1 et 100 pour rendre l'animation possible. Enfin la méthode « draw » permet entre autre de lancer l'animation en dessinant successivement des rectangle dont la hauteur est proportionnelle à une valeur de la liste à trier.

Pour réaliser l'animation nous avons choisi d'afficher un histogramme de valeur aléatoires comprises entre 0 et 100. Une fois l'application lancée, chaque rectangle de l'histogramme bouge dynamiquement en fonction du déroulement de l'algorithme choisi. A la demande de l'enseignant et pour rendre l'animation plus claire nous avons affiché sous chaque rectangle le nombre qui lui est associé.

Le fichier ressource ne contient que des chaines de caractères contenant les différentes descriptions ainsi que le pseudocode de chaque algorithme.

Startpage

Cette classe affiche simplement la page de démarrage avec les différents boutons permettant d'accéder à un algorithme. Chacune des pages de l'interface graphique comporte un bouton qui renvoie l'utilisateur sur cette page s'il le souhaite.

Compare

Cette classe permet de comparer deux algorithmes en faisant apparaître deux canvas. Chacun d'eux est associé à un algorithme choisi par l'utilisateur, ce dernier peut ensuite lancer les deux algorithmes en parallèle afin de savoir lequel est le plus rapide pour une liste donnée.

App et config.py

La classe « app » gère l'architecture générale de l'affichage notamment en générant les différentes pages notamment grâce à un container qui empile les différentes fenêtres les unes sur les autres. Celle qui doit être affichée sera donc en haut de la pile.
Le fichier « config » est juste là pour faire le lien avec les liste et les algorithme, afin de traiter les mêmes listes dans tout le code. Elle dispose également d'un numéro de référence pour chaque algorithme. Cette classe ne comporte aucune méthode.

Difficultés

Nous avons rencontré quelques problèmes sur la manière de réaliser l'animation mais après quelques recherches sur internet, il nous a semblé judicieux d'utilisation la bibliothèque tkinter qui possède de nombreuses fonctionnalités liées aux affichages d'animations. L'un des membre du binôme connaissait déjà cette bibliothèque ce qui a ensuite facilité l'implémentation de l'animation.

A la fin du projet nous avons rencontré quelques problèmes d'affichage car certains fonts n'étaient pas compatibles entre Linux et Mac OS. Les fonts permettant de choisir la police de caractères du texte des descriptions ou des boutons. Pour gérer ce problème nous avons choisi de coder la suite du projet sur une seul machine sous linux.

Bilan

Il s'agissait ici de coder un programme permettant à un utilisateur d'étudier et de comparer les algorithmes de tri. En utilisant les fonctionnalités et les bibliothèques basiques de python il a été possible de réaliser ce programme de manière simple. Chaque algorithme suit le même modèle. Il est donc toujours possible de rajouter d'autres algorithmes en utilisant

ledit modèle proposé. L'animation peut au passage être modifiée si besoin bien qu'elle soit adaptée pour la plupart des algorithmes actuels.

Il est toujours possible de rajouter quelques fonctionnalités visuelle au programme mais le produit fini répond aux normes du cahier des charges et est prêt à l'utilisation.

Les contraintes temporelles ne nous ont pas permis d'implémenter certaines fonctionnalités plus ou moins importantes. Tout d'abord nous n'avons pu décrire que trois algorithmes, néanmoins le modèle utilisé ne change pas et il est alors simple d'ajouter autant d'algorithmes que nécessaire. Nous souhaitions également pouvoir lancer l'application avec différentes langues, cela est simple à implémenter mais nous avons manqué de temps.