Randriansoa Nathan, Fellouh Corentin TG1

Rapport Projet 3 :

Questions IV :

note point a trouver pour les fichier :
data1.txt = 6
data2.txt = 7
data3.txt = 15

1) Pour le démontrer, on suppose que P est un point d'EPM(SL) et Q est un point d'EPM(SR) tel que P soit dominé par Q.

   Puisque P est dominé par Q, cela signifie que pour toute dimension i, on a P[i] ≤ Q[i].

   Puisque Q est un point dans SR, et que T est le point avec la plus petite coordonnée x, cela signifie que T possède la plus grande coordonnée y de EPM(SR) donc yP ≤ yQ ≤ yT.

   Aussi, la droite des abscisses étant croissante, l'abscisse de P sera toujours inférieur -à l'abscisse de T donc xP ≤ xT ≤ xQ.

   On a donc pour toute dimension i, P[i] ≤ Q[i] et P[i] ≤ T[i].

   Nous avons donc prouvé que si un point P d'EPM(SL) est dominé par un point Q quelconque d'EPM (SR), alors P est également dominé par T.

2) algorithme sous forme de pseudo code :

   variables :
   S ⇒ Listes de points trier
   SR ⇒ Liste: Point
   SL ⇒ Liste: Point
   Pareto = []

   DÉBUT :
   SI (S <= 1) :
   ```
       Retourner  S
   ```
   FIN SI
   TRIER(S)
   L,R = DÉCOUPER(S)
   SL = EPM(L)
   SR = EPM(R)
   t = min(SR)

   FOR point IN SL:
   ```
       SI NON domine(t,point):
   ```

```
                ajouter(SR,point)
            FIN SI
        FIN FOR

        RETURN SR
```

3) La complexité de la fonction EPM s'exprime en O(n log(n)), en effet car EPM est une fonction qui se résout récursivement en divisant à chaque fois la liste originel par deux donc O(log(n)), mais également dans cette fonction nous avions une boucle for qui a une complexité en O(n), donc en rassemblant tout ca on en déduit que le fonction a une complexité de O(n log(n)).

Question V :

1) Implementation en python ⇒ voir fichier index.py dans le même dossier.

2) Après avoir effectué l'implémentation en python, nous avons calculé le front de pareto du fichier data2.txt on remarque que son front de pareto est composé de 7 point maximaux : [(9999, 4017), (9974, 9226), (9943, 9396), (9310, 9984), (7744, 9986), (5154, 9991), (3551, 9997)].

3) Pour le fichier data3.txt, son front de pareto est composé de 15 point maximaux : [(499984, 283573), (499972, 297641), (499949, 355110), (499938, 404426), (499930, 419405), (499926, 470311), (499910, 481428), (499882, 496626), (499832, 498581), (499670, 499473), (498355, 499850), (490889, 499884), (460451, 499954), (431226, 499981), (210380, 499992)]

infos sur le script python :

Les fonction :

- LirePoint() : cette fonction prend en paramètre l'adresse d'accès du fichier contenant les point afin de ressortir une liste de points composés de tuples

- EPM() : Cette fonction prend en paramètre une liste de points et calcul son front de pareto (Ensemble de points maximaux). Elle s'exécute de manière récursive et retourne le front de pareto de la liste de points mis en paramètre.

Fonctions de la Partie graphique :

- graphique() : Fonction de l'interface graphique, affiche le repère orthonormé, l'ensemble de point en noir ainsi que les points et la courbe du front de pareto respectivement en bleu et en rouge

- open_file() : Ouvre une fenêtre de l'explorateur de fichier afin que l'utilisateur sélectionne le fichier txt contenant l'ensemble de point à utiliser.

- regenerate() : S'activera après avoir généré un front de pareto qui demande à l'utilisateur s' il souhaite générer un autre front depuis un autre fichier txt, il remettra toutes le variable a 0 et ouvrira de nouveau l'explorateur de fichier.

- quit() : comme son nom l'indique cette fonction permet de quitter le programme, elle est exécutée par le biais d'un bouton situé sur la fenêtre principale de l'interface graphique et vous demande à travers une petite fenêtre de confirmation si l'utilisateur confirme vouloir quitter le programme

Fonctions "Diviser pour régner" :

- Trier() : Elle prend en paramètre une liste de point composé de tuple (x,y), et la renvoie trier dans l'ordre croissant, une vérification est d'abord effectuée afin de s'assurer que la syntaxe est correcte.

- domine() : Cette fonction permet de vérifier entre deux points si l'un domine l'autre, en prenant en premier argument un point p1 et en deuxième un point p2, des vérifications sont effectuées afin de s'assurer que la syntaxe est correcte.

- taille() : Prend en paramètre une liste de points et renvoie sa longueur.

- découper() : Prend en paramètre une liste qui sera découpée en deux au niveau du point médian et en retournera deux nouvelles liste de point distincts : SL pour la partie a gauche du point médian et SR pour la partie droite. Une vérification est faite afin de vérifier que la syntaxe est correcte.

- ajouter() : Prend en paramètre une liste de points et un point et retournera cette liste avec le nouveau point ajouté à sa fin, des vérification sont faite afin de vérifier que la syntaxe est correcte.

infos sur l'utilisation du programme :

nous utilisons les modules : numpy, matplotlib, tkinter et typing. Nos tests ont été effectués sur la version 3.10.9 et 3.11.1 de python, on ne peut garantir un fonctionnement parfait avec des versions antérieures.

Etapes d'utilisation :

En exécutant le code, une interface graphique s'affiche, vous pouvez y voir distinctement deux boutons différents

- Le premier permet de sélectionner et charger un fichier txt pour calculer son front de pareto
- l'autre sert à quitter le programme avec une confirmation

Pour calculer un front de pareto il suffit de cliquer sur le bouton qui permet d'ouvrir un fichier txt, une fenêtre de l'explorateur windows s'affiche, il ne reste qu'à sélectionner votre fichier de points, une fois le fichier trouvé, un graphique de matplotlib s'affiche où vous pourrez y voir tout vos point ainsi que la courbe du front de pareto.

une fois que vous quittez le graphique, une fenêtre pop up s'affiche vous demandant si vous voulez générer un autre front de pareto à partir d'un autre fichier txt ou non, en cliquant sur "oui" une nouvelle fenêtre de l'explorateur windows s'ouvre et le processus se répète comme ci dessus, en cliquant sur "non" le programme s'arrête.

Vous disposez d'un fichier "incorrect_data_test.txt" pour tester les protections du programme