TP C++ 2
Description des tests Fonctionnels
Jacques FOLLEAS Arnaud DUPEYRAT

Tous les tests effectués ont été remis dans le fichier tests. Chaque commande précedée d'un **~** posséde un dossier individuel contenant les différents tests numérotés.
Dans la suite de ce document, chaque numerotation de test fait directement référence au dossier de la méthode testée.
Ce dossier contient le fichier d'entree .in et le fichier de sortie .out attendus, permettant ainsi la mise en place des tests de non regression.
Tous les tests fonctionnels ont été effectués sur un nombre "important" de données afin de correspondre au mieux à la réalité. Les différents fichiers d'entrées ont été générés le plus souvent avec un ordre de capteur aléatoire.
La commande "ADD" sera testée implicitement par tous les autres tests effectués. Nous pensons alors que cette commande ne nécessite aucun test supplémentaire.

Nos tests fonctionnels sont regroupés en deux parties:

Premierement les tests de **fonctionnement des differentes méthodes** contenu dans notre programme.

1) ~ COMMANDE STATS_C <idCaptor>

Utilisation normale:
- Verifier que l'affichage est correct vis à vis du cahier des charges **(TestSTATCS_C1)**
- La commande doit prendre en compte le capteur mis en paramètre et uniquement celui-là **(TestSTATS_C2)**.

Fonctionnalités paticulières:
- La commande doit afficher des pourcentages nuls si aucune valeur n'a été ajouté pour le capteur referencé par l'ID placé en paramètre. **(TestSTATS_C3)**

2) ~ COMMANDE JAM_DH <d7>

Utilisation normale:
- La méthode doit prendre en compte toute la plage horaire possible pour les différentes valeurs entrées par l'utilisateur. **(Test 1)**
- La méthode doit prendre en compte la différence de couleur représentant les embouteillages, ainsi que la gestion du temps totale **(Test 2) (Test 3)**
- La méthode prend en compte uniquement le jour placé en paramètre **(Test 4)**
- Tester notre méthode sur un nombre important d'évenements avec des données aléatoires. **(Test 6).**
  Ce test a été réalisé sur 3200 valeurs ajoutées le même jour avec aléatoirement 2365 segments embouteillés c'est à dire 'R' ou 'N'. Ainsi le pourcentage obtenu est égal à 73.90. Donc le resultat attendu de la part de notre programme est l'arrondi inférieur soit **73%** d'embouteillage pour ce jour.

Fonctionnalités paticulières

- Vérifier que la méthode nous indique un niveau d'embouteillage nul si aucun capteur n'a été ajouté au traffic. **(Test 5)**

3) ~ COMMANDE STATS_D7 <d7>

Utilisation normale:

- La méthode prend bien en compte uniquement le jour placé en paramètre **(Test 2)**
- La méthode doit prendre en compte, la gestion des évements en faisant la moyenne du temps passé dans un état particulier du trafic, ainsi que les differentes couleurs possibles pour les segments étudiés. **(Test 4)**
- Verifier que la méthode fonctionne sur un nombre important d'évenements avec des données aléatoires. **(Test 3)**
  Test réalisé sur 3200 valeurs ajoutées le même jour, avec aléatoirement 835 'N', 762 'R',795 'V' et 808 'J'. Ainsi les resultats attendus sont les arrondis inferieurs de:
  V 24,84% J 25,25% R 23,8% N 26,1% (en colonne)

Fonctionnalités paticulières

- Verifier que la méthode indique un niveau d'embouteillage nul si aucun capteur n'a été ajouté au traffic. **(Test 1)**

4) ~ COMMANDE OPT <D7> <H_start> <H_end> <seg_count> <seg_1>...<seg_n>

Utilisation normale:

- Verifier que l'affichage est correct vis à vis du cahier des charges.et que la commande retourne le départ optimal **(TestOPT1)**.
- La commande doit prendre en compte tous les segments mis en paramètre **(TestOPT2)**.
- La commande doit afficher "impossible d'arriver à l'heure" si il y a trop de segments à parcourir dans la tranche horaire, et qu'il est impossible d'arriver à l'heure **(TestOPT4)**
- La commande doit afficher "impossible d'arriver à l'heure" si le trafic est trop dense pour faire le trajet dans la tranche horaire mis en paramètre. **(TestOPT5)**

Fonctionnalités paticulières:

- Verifier que la commande, par défaut considére un traffic fluide, si aucune information n'est précisée pour le capteur représentant le capteur étudié **.(TestOPT6)**

Deuxiemement les tests de **performances du programme**:

Test de Temps de Reponse

Le programme doit étre en mesure de traiter 20 000 000 évenements, le test contient un unique capteur dont tous les evénement sont de couleurs 'N' puis trois commandes STAT_C,JAM_DH et STATS_D7. La durée du test doit alors rester inférieur à 30 secondes (mesure réalisée sur les machines du departement)

Test utilisation Memoire

Le programme doit etre en mesure de traiter des informations venant de 1500 capteurs sans dépasser la capacité mémoire permise sur les ordinateurs du département. **(TestPERF2)**
Le programme ne doit pas généré de fuite mémoire..**(TestPERF3).** (Commande UNIX valgrind)