Julien De Loor [julien.de-loor1@uqac.ca] (DELJ17089301)

Devoir 2/TP2

Tous les programmes sont en ligne de commandes.
Généralement les programment attendent que l'on appuis sur une touche du clavier à la fin de l'exécution.

Généralement il faut spécifier le fichier en premier argument et la distance ensuite (si il y a une distance dans la question).
La seule exception est la question 6 qui s'utilise que ceci:
question6.exe <key> <msg>

Question 1

Pour la première question j'ai utilisé une file prioritaire (std::priority_queue):
L'algorithme se trouve dans *army.cpp* aux fonctions *affectNaive* et *affectSolution*.

Le pseudo code est le suivant:

Trier les tours de la plus grande estimation à la plus petite dans une file
Pour chaque soldat non affecté:
    Prendre la tour avec la plus grande estimation
    Affecter cette tour
    Supprimer la tour de la file
Fin Pour

Question 2

L'algorithme est quasiment identique à la première question sauf qu'il tiens compte de la contrainte supplémentaire. Il se trouve dans *army.cpp* aux fonction *affectNaiveDist, CanAddToSolutionDist* et *affectSolution*.

Le pseudo code est le suivant:

Fonction TrouverUneTourValide:
    Debut Boucle:
        Prendre la tour avec la plus grande estimation dans la file
        Si la tour satisfait les contraintes de distances selon les autres tours affectés:
            Supprimer la tour de la file
            renvoyer la tour.
        Sinon:
            Supprimer la tour de la file

Fin Boucle
Fin Fonction

Fonction AffecterTourDistance:
Trier les tours de la plus grande estimation à la plus petite dans une file
Pour chaque soldat non affecté:
TrouverUneTour
Affecter cette tour au soldat
Fin Pour
Fin Fonction

Question 3

Pour la question 3 l'algorithme est quasi identique à la question 2 sauf je calcule le nombre réel de zombie tués au moment ou j'affecte les soldat, ensuite si cette valeur est inférieure à l'estimation, je modifie les estimations des tours adjacentes, et je tri à nouveau les tours en fonction des nouvelles estimations. Le code se trouve dans *army.cpp* aux fonction *affectWeatherDist, CanAddToSolutionDist* et *affectSolution*.

Dans ce cas si je n'utilise plus std::priority_queue car je dois re-trier les tours lorsque la valeur réelle est inférieure à l'estimation. (notez que l'algorithme regarde aussi si un soldat est déjà affecté pour pouvoir être réutilisé à la question 4).

Le pseudo code est le suivant:

Fonction TrouverUneTourValide:
Debut Boucle:
Prendre la tour avec la plus grande estimation dans la file
Si la tour satisfait les contraintes de distances selon les autres tours affectés:
Supprimer la tour de la file
renvoyer la tour.
Sinon:
Supprimer la tour de la file
Fin Boucle
Fin Fonction

Fonction AffecterTourMeteoDistance:
Trier les tours de la plus grande estimation à la plus petite dans une file
Pour chaque soldat:
Si le soldat N,a pas de tour affecté:
TrouverUneTour
Affecter cette tour au soldat
Calculer le nombre de zombis tués
Si le nombre de tués est inférieur à l'estimation de la tour:
Calculer les estimations pour les tours adjacentes

Trier à nouveau les tours dans la file
Fin Pour
Fin Fonction

Question 4

Encore une fois pour la question 4 l'algorithme est presque identique à la question précédente. Sauf que le segment de code *TrouverUneTourValide (CanAddToSolutionDist)* a été modifier pour prendre en compte une nouvelle variable sur les tours: le fait que la valeur réelle (membre *doNotUse* de la classe *Tower*) est été inférieur à est l'heure précédente.

Le code se trouve dans *army.cpp* aux fonction *affectWeatherDist, CanAddToSolutionDist* et *affectSolution*.

Fonction TrouverUneTourValide:
Debut Boucle:
Prendre la tour avec la plus grande estimation dans la file
Si la tour satisfait les contraintes de distances et est marqué comme utilisable:
Supprimer la tour de la file
renvoyer la tour.
Sinon:
Supprimer la tour de la file
Fin Boucle
Fin Fonction

Ensuite l'algorithme est exécutable heure par heure:

Pour chaque nouvelle heure:
Réinitialiser les estimations météo aux estimations de départ
Algorithme de la question 3
Marquer toutes les tours comme utilisables
Pour chaque soldats:
Si le soldat a tué mois de zombis que l'estimation:
Marquer la tour du soldat comme inutilisable
Marquer le soldat comme non affectés.
Fin Pour

Question 5

J'ai utilisé un algorithme génétique.
Les paramètres de cet algorithmes sont réglables dans *main.cpp* du projet *question 5*.

Générer une population de N solutions valides aléatoires
Créer un individu Best avec une distance à parcourir valant l'infini
Tant que on n'a pas effectuer un certains nombre d'iteration OU
Tant que on n'a pas effectuer un certains nombre d'iterations sans amélioration:
    Calculer la distance (fitness) à parcourir de chaque individu
    Trier les individu de la population selon leur distance à parcourir
    Si la distance de l'individu avec la distance la plus faible < distance du Best:
        Affecter Best avec cet individu
        Remettre à zéro le nombre d'iterations sans améliorations
    Sinon
        Incrementé le nombre d'iterations sans améliorations.

    Garder les M meilleurs individus.
    Ajouter les M meilleurs des individus précédents.
    Dupliquer les individus pour obtenir N individus.

    Pour chaque individus:
        Effectuer plusieurs permutations aléatoires
        Calculer la distance à parcourir de l'individu.
    Fin Pour
Fin Tant que

Question 6

Pour la question6 j'ai utilisé un algorithme de cryptage symétrique vigenère avec une table XOR.
L'algorithme est le même pour crypter ou decrypter.
Pour des raisons pratique j'ai transposés les caractères de la table ASCII sur une table 6bits contenant [A-Z] [a-z] [0-9] et ' ' afin que le résultat soit toujours visible dans la console.

Pseudo code:

Tant que la longueur de la clée est inferieure à la longueur du message:
    concaténer la clée avec elle même
Fin Tant que

Pour chaque caractères du message:
    m = Trouver l'entier 6bits correspond au caractère
    k = Trouver l'entier 6bits correspond au caractère courant de la clee
    o = k XOR m
    Ajouter à la chaine de retour le caractere ASCII correpondant à o

    Prendre le caractère suivant de la clée
Fin Pour