Université Polytechnique des Hauts-De-France
Institut des Sciences et Techniques ISTV

Licence 1 : TP d'Algorithmique Semestre2 - 2019
Génétique

Sac et reproduction

Un petit TP pour initier à la génétique au travers la reproduction de sacs…
il faut choisir parmi NBOBJETS des objets à placer dans un sac pour partir en randonnée. Les objets ont différents poids et sont plus ou moins intéressants. Il s'agit donc de trouver le sac le plus intéressant et le moins lourd.
Un sac contient donc : un contenu, modélisé par un tableau de NBOBJETS entiers compris (0 ou 1); un poids (la somme des poids des objets qu'il contient); un intérêt (la somme des intérêts des objets); une note (calculée à partir de l'intérêt et du poids).
Ainsi le sac dont le contenu est (1, 0, 0, 1, 0 ,1, 1, 0) contient les produits 0, 3, 5 et 6

Le but est de créer une population de sac; puis de les noter, de faire se reproduire entre eux les meilleurs (les enfants remplacent les sacs les moins intéressants dans la population de sacs), de faire des mutations. Il suffit ensuite répéter une série de reproductions sur plusieurs générations.

Reprenez le code présent sur moodle.

- Les types, appelés TabInt et TabReels représentent respectivement un tableau à une dimensions d'entiers et de réels.
- La constante entière NBOBJETS est initialisée à 8 et la constante NBSACS est initialisée à 40
- La constante POIDS est un tableau de 8 réels donnant le poids des produits
- La constante INTERETS est un tableau de 8 réels donnant le poids des produits
- La fonction randint(0,n) retourne un nombre entier aléatoire entre 0 et n
- Le type composé (classe) Sac qui contient comme attribut : genome, un tableau d'entiers; le poids, l'intérêt, et sa note en réels.
- Le type TabPopulation est un tableau de Sac

1. Définir la procédure **`calculProprietes(sac:Sac)`** qui calcule le poids, l'intérêt et la note du sac. La note est obtenue ainsi : `sac.note = sac.interet - sac.poids`
   Ainsi la note d'un sac croît avec son intérêt et décroit avec son poids (lors d'utilisation de plusieurs critères, il faut en général les « normaliser » pour qu'ils soient sur le même intervalle de valeur; mais c'est le cas dans les données présentes dans le TP, donc on ne normalise pas ici l'intérêt et le poids)
2. Définir la procédure **`permuter(tab: TabPopulation, i: int, j: int)`** qui permute les sacs situés aux positions i et j du tableau tab.
3. Définir la procédure **`triPopulationInsertion(population:TabPopulation)`** qui tri les sacs contenus dans le tableau population par ordre de note décroissante, en utilisant le tri par insertion.
4. Définir la procédure **`croisementSacs(parent1:Sac, parent2:Sac, enfant1:Sac, enfant2:Sac)`** qui crée le contenu des sacs enfants enfant1 et enfant2 à partir des sacs parents parent1 et parent2. La 1ère moitié des parents 1 et 2 est recopiée dans les enfants 1 et 2, puis la seconde moitié des parents 2 et 1 est recopiée dans les enfants 1 et 2.
   Par exemple : si parent1.contenu = (1,0,0,0,1,1,0,0) et parent2.contenu = (0,0,1,1,0,1,0,1) alors enfant1.contenu = (1,0,0,0,0,1,0,1) et enfant2.contenu = (0,0,1,1,1,1,0,0)
5. Définir la procédure **`mutationPopulation(population:TabPopulation)`** qui prend 1 sac au hasard dans la population, et fait muter un élément de son contenu au hasard *(par exemple sélectionne au hasard le 10e sac et transforme le 4e élément de son contenu (qui passe de 0 à 1 ou inversement).*
6. Définir la procédure **`reproductionDeSacs(nb:int, population:TabPopulation)`** qui effectue nb fois la boucle : tri de la population, reproduction, et mutation.
7. Utilisez la procédure **`vie()`** fournie dans le code pour tester vos fonctions. Faites varier le nombre de reproductions pour observer l'accroissement de la valeur des sacs.