Cartouche du document

Année : ING 1
Matière : Algorithmique
Activité : Examen

Objectifs

Cet examen teste vos aptitudes en algorithmique sur
- les graphes
- la complexité

Tous les documents (électronique et autres) sont autorisés.

La durée de l'examen est de 1h30 heures.

Sommaire des exercices

1 - Graphes
2 - Complexité

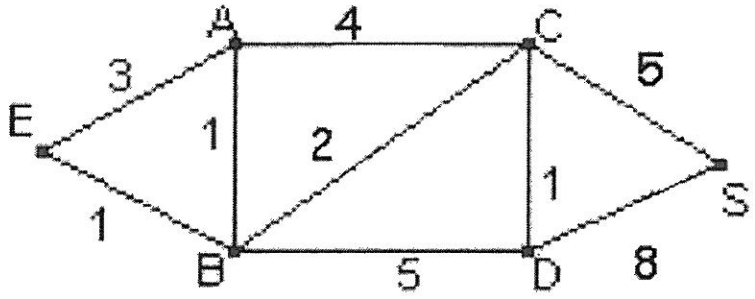
Corps des exercices

1 - Graphes

Question 1)

Enoncé de la question

On considère le graphe suivant :



Un petit graphe

Quand deux sommets a et b sont reliés, alors les arcs ab et ba sont dans le graphe. La valuation des deux arcs sont les mêmes.

On vous demande d'appliquer l'algorithme de Dijkstra du sommet E vers le sommet S. Vous devez formuler votre réponse en donnant les différentes étapes du déroulement de l'algorithme.

Question 2)

Enoncé de la question

On vous demande d'écrire l'opération qui permet d'adapter l'algorithme de Dijkstra de la façon suivante :

Cette opération reçoit deux paramètres :
- Le sommet de départ r
- un réel vmax

Le graphe que doit construire cette opération ne contient que les chemins les plus courts partant de r et dont la valuation est inférieure ou égal à vmax.

On rappelle l'algorithme de *Dijkstra* (Voir année)

2 - Complexité

Question 1)

Enoncé de la question

On considère l'algorihtme de tri suivant :

```
void trier(int [] tableau, int n)
{
   int pas, i, j, memoire;
   pas = 0;

   // Calcul du pas
   while(pas < n)
   {
      pas = 3*pas+1;
   }
      pas
   while(pas!=0) // tant que le pas est > 0
   {
      pas = pas/3;
      for(i=pas; i < n; i++)
      {
         memoire = tableau[i]; // valeur à décaler éventuellement
         j = i;

         while((j > (pas-1)) and (tableau[j-pas] > memoire))
         { // échange des valeurs
            tableau[j] = tableau[j-pas];
            j = j-pas;
         }
```

```
            tableau[j] = memoire;
        }
    }
}
```

Le temps de calcul de ce tri dépend évidemment
- de la taille n du tableau
- du nombre d'opérations élémentaires effectuées (affectations, additions , comparaison et accès à un élément du tableau)

En notant
- t : le temps d'une affectation d'entiers
- top : le temps d'une opération arithmétique
- t[ ] : le temps d'un accès à un élément du tableau
- tc : le temps d'une comparaison
- a(n) : le nombre d'affectations pour un tableau de taille n

déterminer le temps T(n) pour trier ce tableau et donner son ordre de complexité.

Annexes (Arel)

Référence : Dijkstra

Observateur Graphe algoDijkstra (Entier r) : Graphe
// Le paramètre implicite
Observé gr

Références locales
Ensemble s
Entier pivot, nbSommets, j, y
Vecteur pi
Graphe res
vecteur succ

// Début de l'algorithme
DEBUT
s <-- ensembleVide()
ajouter(s,r)

```
pivot <-- r

// On crée le graphe résultat simplement avec les sommets
res <-- creerGraphe(nbSommets)


nbSommets <-- recNbSommets(gr)
pi <-- creerVecteur(1, nbSommets)
affVal(pi,r,0)

// On met la valuation de tous les sommets autre que r à +infini
j <-- 1
TANTQUE j <= nbSommets FAIRE // Structure itérative
 DEBUT
 SI NON estEgal(j,r) ALORS // Structure conditionnelle
  DEBUT
  affVal(pi,j,MAX_INT)
  FIN
 j <-- j + 1
 FIN

j <-- 1
TANTQUE j <= (nbSommets - 1) FAIRE // Structure itérative
 DEBUT
 succ <-- recSuccesseurs(gr,pivot)
 no <-- 1
 TANTQUE no <= borneSup(succ) FAIRE // Structure itérative
  DEBUT
  y <-- recVal(succ,no)
  SI NON appartient(s,y) ALORS // Structure conditionnelle
   DEBUT
   SI recval(pi,pivot) + recValuation(recArete(gr,pivot,y)) < recVal(pi,y) ALORS //
Structure conditionnelle
    DEBUT
    affVal(pi,y,recval(pi,pivot) + recValuation(recArete(gr,pivot,y)))
    FIN
   FIN
  no <-- no + 1
  FIN

 y <-- 1
 min <-- MAX_INT
 TANTQUE y <= nbSommets FAIRE // Structure itérative
  DEBUT
```

```
    SI NON appartient(s,y) ALORS // Structure conditionnelle
     DEBUT
     SI recVal(pi,y) < min ALORS // Structure conditionnelle
      DEBUT
      ymin <-- y
      min <-- recVal(pi,y)
      FIN
     FIN
    y <-- y + 1
    FIN
   ajouterArete(res,creerAreteValuee(pivot,y,recValuation(recArete(gr,pivot,y))))
   pivot <-- ymin
   ajouter(s,pivot)
   j <-- j + 1
   FIN
  retourner res
  FIN
```

Cartouche du document

Année : ING 1
Matière : algo
Activité : Examen

Objectifs

Cet examen teste vos aptitudes en algorithmique
- à utiliser les types abstraits existants
- à définir des types abstraits
- à construire des algorithmes et en particulier à utiliser des structures de contrôles

Tous les documents (electronique et autres) sont autorisés.

La durée de l'examen est de 1h20 heures.

Sommaire des exercices

1 - Conteneur de nombres premiers
2 - Fusion de listes
3 - Un petit type abstrait
4 - Réflexion : Comparaison de tris

Corps des exercices

1 - Conteneur de nombres premiers

Enoncé :

Nous allons écrire un algorithme permettant d'engendrer les nombres premiers inférieurs à une certaine valeur n entière passée en paramètre. Pour cela nous allons construire une structure linéaire initialisée par le nombre premier 2, à laquelle nous rajouterons les nouveaux nombres premiers en vérifiant qu'ils ne sont pas divisibles par les nombres premiers déjà trouvés.

Question 1)

Enoncé de la question

Quelle type abstrait choisiriez vous pour implémenter cet algorithme. Justifiez votre réponse.

Question 2)

Enoncé de la question

Ecrire l'opération d'extension du type abstrait que vous avez choisi, qui engendre les nombres premiers inférieurs à un paramètre n selon la méthode ci-dessus.

2 - Fusion de listes

Enoncé :

On souhaite fusionner deux listes quelconques L1 et L2 en une liste LF dans laquelle les éléments de L1 et L2 sont intercalés. Ainsi, LF contient le 1^{er} élément de L1, puis le 1^{er} élément de L2, puis le $2^{ème}$ élément de L1, puis le $2^{ème}$ élément de L2, etc. jusqu'à épuisement des éléments de L2. Le résiduel de L1 est ajouté à la fin de la liste LF.

On considèrera que la fusion ne peut pas être effectuée si la liste L1 est vide ou si le nombre d'éléments de la liste L2 est strictement supérieur à celui de L1.

Si L2 est vide, la fusion peut se faire et la liste fusionnée est identique à L1.

Question 1)

Enoncé de la question

Donner la signature exacte de cette opération d'extension. On n'oubliera pas de spécifier les préconditions de cette opération sous forme axiomatique.

Question 2)

Enoncé de la question

Ecrire l'opération d'extension.

3 - Un petit type abstrait

Enoncé :

Il s'agit dans cet exercice de définir le type abstrait rectangle modélisant des rectangles dans l'espace R^2 . Dans cet espace, on ne s'intéresse qu'aux rectangles parallèles à l'axe des x.

On doit pouvoir construire un nouveau rectangle de deux façons différentes
- en donnant les coordonnées du point en haut à gauche et du point en bas à droite
- en donnant les coordonnées du point en haut à gauche, sa longueur et sa hauteur

Avec un objet de ce type, on doit pouvoir :
- récupérer le couple des coordonnées du point en haut à gauche
- récupérer le périmètre d'un rectangle
- tester la position d'un point par rapport à un rectangle (extérieure, intérieure ou sur la frontière)
- récupérer le couple des coordonnées du point en bas à droite
- récupérer la hauteur du triangle
- translater un rectangle
- récupérer la largeur du triangle

Dans cet exercice, on ne vous demande pas d'écrire les algorithmes des opérations d'extensions

Question 1)

Enoncé de la question

Donner la signature exacte des opérations de base. On n'oubliera pas de spécifier les préconditions (s'il y a lieu).

On jusfiera pourquoi elles ont été rangées dans la catégorie des opérations de base.

Question 2)

Enoncé de la question

Donner la signature exacte des opérations d'extension. On n'oubliera pas de spécifier les préconditions (s'il y a lieu).

On jusfiera pourquoi elles ont été rangées dans la catégorie des opérations d'extension.

Pour l'opération translater, on écrira sous forme axiomatique (postconditions) l'objet après transformation

4 - Réflexion : Comparaison de tris

Enoncé :

En comparant les complexités des trois tris vus en travaux dirigés, donner les avantages et inconvénients de chacun d'eux.

Cartouche du document

Année : ING 1
Matière : Algorithmique
Activité : Examen

Objectifs

Cet examen teste vos aptitudes en algorithmique sur :

- les graphes ;
- la complexité.

Tous les documents (électronique et autres) sont autorisés. Votre ordinateur est autorisé seulement en local.

La durée de l'examen est de 1h30.

Sommaire des exercices

1 - Graphes
2 - Algorithmes gloutons

Corps des exercices

1 - Graphes

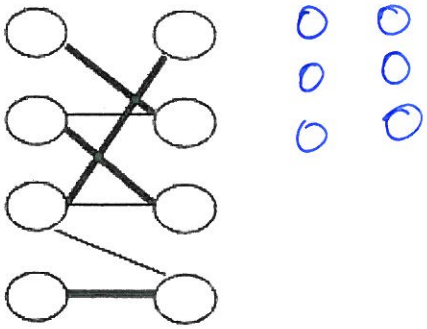
Énoncé :

Dans cet exercice, on s'intéresse à un graphe biparti. Un graphe biparti est un triplet

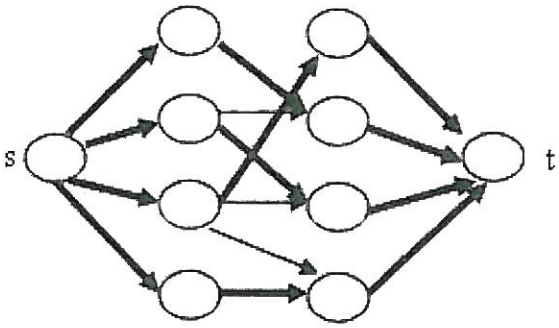
B = { U, V, E } où :

- U est un ensemble de sommets appelés garçons ;
- V est un ensemble de sommets appelés filles ;
- E est un ensemble d'arêtes dont l'une des extrémités est un garçon et l'autre une fille.

On supposera dans la suite que card(U) = card(V).



Graphe biparti
Les sommets à gauche représentent les garçons, ceux de droite les filles.
Les arêtes représentent les associations possibles. Les arêtes en gras sont celles du couplage.



Graphe équivalent
sous forme de réseau de transport

Un *couplage parfait* ou plus simplement *couplage* est un sous-graphe de B où chaque garçon est associé exactement à une fille et réciproquement.

Question 1)

Énoncé de la question

Donner un exemple de graphe biparti qui ne possède pas de couplage.

Question 2)

Énoncé de la question

Détailler les étapes qui permettent de transformer un graphe B en problème de flot (réseau de transport).

Question 3)

Énoncé de la question

Écrire l'algorithme qui, à partir d'un graphe biparti, renvoie un couplage s'il existe.

2 - Algorithmes gloutons

Énoncé :

Le problème de la couverture d'ensembles consiste à couvrir les éléments d'un ensemble à partir de sous-ensembles d'éléments, en utilisant le moins de sous-ensembles possible. Par exemple, imaginons que l'on dispose d'un ensemble de personnes disponibles pour travailler sur un problème qui nécessite un ensemble de compétences. Chaque personne possède un certain nombre de compétences, et le but est de créer une équipe contenant le moins de personnes possible qui couvre l'ensemble des compétences nécessaires au problème.

Question 1)

Énoncé de la question

Écrire l'opération d'extension du type abstrait Ensemble qui retourne le nombre d'éléments communs entre deux ensembles.

Question 2)

Énoncé de la question

Utiliser cette opération afin d'écrire un algorithme glouton qui résoud le problème de la couverture d'ensemble, selon la méthode suivante :

- choisir une personne qui maximise l'intersection entre ses compétences et les compétences non encore couvertes ;
- mettre à jour les compétences nouvellement couvertes ;
- mettre à jour les personnes disponibles ;
- recommencer tant qu'il reste des compétences non couvertes.

Question 3)

Énoncé de la question

Déterminer la complexité de cet algorithme. Vous ne ferez pas tous les calculs mais justifierez clairement votre réponse.

Cartouche du document

Année : ING 1
Matière : algo
Activité : Examen de rattrapage

Objectifs

Cet examen teste vos aptitudes en algorithmique
- à utiliser les types abstraits existants
- à définir des types abstraits
- à construire des algorithmes et en particulier à utiliser des structures de contrôles

Tous les documents (electroniques et autres) sont autorisés.

La durée de l'examen est de 1h30.

Vos réponses à l'examen seront rédigées dans un unique fichier texte que vous envoyez par mail à yannick.lenir@eisti.fr pour les étudiants de Pau et à hdm@eisti.fr pour les étudiants de Cergy. Dans les deux cas votre mail doit avoir pour sujet "Examen Ratt Algo".

Sommaire des exercices

1 - Conteneur pour calculs statistiques : 6 points
2 - Fusion de piles : 6 points
3 - Un petit type abstrait : 8 points

Corps des exercices

1 - Conteneur pour calculs statistiques : 6 points

Enoncé :

On interroge au hasard n personnes (n est connu à l'avance). Pour chaque personne, on lui demande son age et son sexe.

Question 1)

Enoncé de la question

Expliquer pourquoi et comment deux vecteurs sont parfaitement adaptés pour stocker ces informations.

Question 2)

Enoncé de la question

Ecrire l'opération d'extension qui reçoit ces deux vecteurs et qui retourne les moyennes des ages par

sexe (ie un couple de valeurs : (age moyen des femmes, age moyen des hommes))

2 - Fusion de piles : 6 points

Enoncé :

On dispose de deux piles non vides. On suppose que chaque pile est ordonnée. On souhaite injecter la deuxième pile dans la première. Le nouveau contenu de la première pile doit toujours être trié.

Question 1)

Enoncé de la question

Donner la signature exacte de l'opération d'extension qui permet de réaliser cette opération. On n'oubliera pas de spécifier les préconditions de cette opération sous forme axiomatique.

Question 2)

Enoncé de la question

Ecrire l'opération d'extension.

3 - Un petit type abstrait : 8 points

Enoncé :

Il s'agit dans cet exercice de définir le type abstrait Polyligne modélisant des lignes brisées dans l'espace R^2.

On doit pouvoir :
- 1) créer une nouvelle polyligne en l'initialisant avec un point d'origine.
- 2) la transformer en ajoutant un à un des points.
- 3) calculer la longueur d'une ligne brisée.
- 4) évaluer si la ligne brisée est fermée ou ouverte.
- 5) récupérer sous forme de vecteurs l'ensemble de points de la ligne brisée.
- 6) créer une ligne brisée par concaténation de deux lignes brisées.

Dans cet exercice, on ne vous demande pas d'écrire les algorithmes des opérations d'extension.

Question 1)

Enoncé de la question

Donner la signature exacte des opérations de base. On n'oubliera pas de spécifier les préconditions (s'il y a lieu).

On jusfiera pourquoi elles ont été rangées dans la catégorie des opérations de base.

Question 2)

Enoncé de la question

Donner la signature exacte des opérations d'extension. On n'oubliera pas de spécifier les préconditions (s'il y a lieu) et les postconditions.

On jusfiera pourquoi elles ont été rangées dans la catégorie des opérations d'extension.

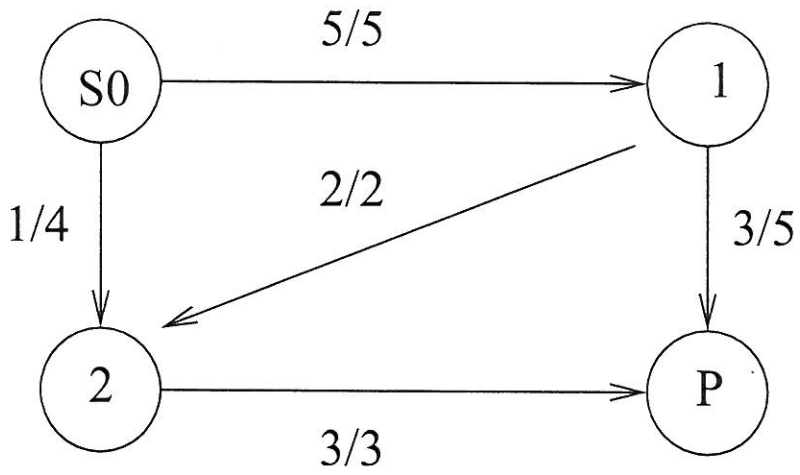
Exam ALGO2 2008

EISTI

14 avril 2008

1 Ford Fulkerson

1. Dans la fonction de marquage de l'algorithme de Ford Fulkerson, nous utilisons la fonction `recPredecesseurs(int sommet)`, comme opération d'extension du type abstrait Graphe. Ecrire cette opération d'extension.
2. Soit le marquage suivant sur un graphe lors du déroulement de l'algorithme de Ford Fulkerson :



Faire tourner l'algorithme de Ford Fulkerson à partir de cette configuration, afin de trouver le flot maximum de ce graphe.

2 Algorithmes gloutons

Le problème de la coloration d'un graphe consiste à colorier avec des couleurs différentes l'ensemble des sommets adjacents d'un graphe non orienté. Le but est bien évidemment de colorier l'ensemble des sommets du graphe en utilisant le moins de couleurs possibles.

1. Ecrire l'opération d'extension du type abstrait Graphe, correspondant à l'algorithme glouton qui permet de donner une solution à ce problème, selon le principe suivant :
   - choisir un sommet non encore colorié
   - lui affecter une couleur non encore utilisée par ses voisins immédiats
   - répéter les deux opérations précédantes tant qu'il reste des sommets non coloriés.
2. Déterminer la complexité de cet algorithme. Vous ne ferez pas tous les calculs mais justifierez intuitivement votre réponse.
3. Trouver un exemple de graphe pour lequel votre solution n'est pas optimale.

Type abstrait Graphe

Concept

Ce type permet de modéliser les graphes.
Les sommets sont numérotés de 1 à n.

Opérations de base

Constructeur Graphe : creerGraphe(Entier nbSommets) : Graphe
Transformateur Graphe : ajouterArete(Arete a) : Graphe
Transformateur Graphe : marquer(Entier noS) : Graphe
Transformateur Graphe : demarquer(Entier noS) : Graphe
Observateur Graphe : estMarque(Entier noS) : Booleen

Observateur Graphe : recAretes() : Vecteur
Observateur Graphe : recNbSommets() : Entier
Observateur Graphe : recNbAretes() : Entier
Observateur Graphe : recArete(Entier noSD, Entier noSA) : Arete

Axiomes

Pré-conditions
definie(creerGraphe(nb)) ==> nb > 0
definie(ajouterArete(g,a)) ==> recOrigine(a) >= 1 ET recOrigine(a) <= recNbSommets(g)
definie(ajouterArete(g,a)) ==> recDestination(a) >= 1 ET recDestination(a) <= recNbSommets(g)
definie(marquer(g,noS)) ==> noS >= 1 ET noS <= recNbSommets(g)
definie(demarquer(g,noS)) ==> noS >= 1 ET noS <= recNbSommets(g)
definie(recSuccesseurs(g,noS)) ==> noS >= 1 ET noS <= recNbSommets(g)

Post-conditions
estEgal(recNbSommets(creerGraphe(nb)),nb)
estEgal(recNbAretes(creerGraphe(nb)),0)

Opérations d'extension

// Les prédecesseurs sont rangés par ordre croissant de numéros
Observateur Graphe : recPredecesseurs(Entier noS) : Vecteur

// Les successeurs sont rangés par ordre croissant de numéros
Observateur Graphe : recSuccesseurs(Entier noS) : Vecteur

Examen Algorithmique Séminaire - RATTRAPAGE
ING 1

23 avril 2008

Durée : 1h00

Tous documents sont autorisés.
Les algorithmes devraient être écrits en pseudo-code.

Exercice 1

Écrire un algorithme qui permet de vérifier si un entier naturel non nul est parfait ou pas. Rappelons qu'un entier naturel est dit parfait s'il est égal à la somme de ses diviseurs, lui-même excepté (par exemple : 6 est parfait car $6 = 1 + 2 + 3$).

Exercice 2

On se donne un tableau qui contient une suite d'entiers distincts $a_1, a_2, ..., a_n$. On suppose que cette suite est *circulairement triée* (dans l'ordre croissant), c'est-à-dire qu'il existe un entier i, $1 \leq i \leq n$ tel que $a_{i+1}, a_{i+2}, ..., a_n, a_1, a_2, ..., a_i$ soit triée dans l'ordre croissant. Par exemple, les suites (3, 4, 7, 0, 1, 2), (0, 1, 3), (2, 3, 1) sont de ce type.

$1 \leq i \leq n$

a) Écrire une fonction qui retourne l'indice de début d'une telle suite avec une complexité $O(n)$.

b) Écrire une fonction qui atteint le même résultat avec une complexité $O(\log n)$.

On fournira une explication détaillée de chaque algorithme ainsi qu'une justification du temps de calcul.