

Travaux dirigés d'Algorithmique et Structures de données

Travaux Dirigés 1

Exercice 1 : Décomposition d'une somme en euros

Écrire une procédure `DecomposerSomme` qui, à partir d'une somme d'argent passée en paramètre (un nombre entier), écrit à l'écran le nombre minimal de billets de 50 €, 20 €, 10 € et 5 €, et de pièces de 2 € et 1 € qui la composent.

Exercice 2 : Les nombres parfaits

Proposez un algorithme qui calcule les nombres parfaits jusqu'à 10 000. Un nombre entier est dit parfait si il est égal à la somme de ses diviseurs stricts (1 est inclus, le nombre lui-même est exclus). Exemples de nombres parfaits : $6 = 1 + 2 + 3$; $28 = 1 + 2 + 4 + 7 + 14$; ...

Exercice 3 : Les nombres premiers

Écrire la fonction `EstPremier` qui, à partir d'un nombre entier strictement positif passé en paramètre, retourne le résultat booléen VRAI ou FAUX selon que le nombre est premier ou non.

On ne souhaite pas mémoriser la liste de tous les nombres premiers (1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, ...) et comparer si le nombre entier passé en paramètre appartient à la liste ; cette solution serait trop coûteuse en mémoire et en temps de calcul. On préférera tester l'existence d'un diviseur ou s'appuyer sur des propriétés des mathématiques pour vérifier si un nombre est premier ou non.

Travaux Dirigés 2

Exercice 4 : Le zéro d'une fonction

Écrire une fonction « `ZeroFonction` » qui calcule à epsilon près le zéro d'une fonction. On recherche une solution à l'équation $f(x) = 0$; $f(x)$ est une fonction continue sur l'intervalle $[a,b]$, où elle ne s'annule qu'une seule et unique fois ($f(a)$ et $f(b)$ sont de signes opposés). Pour y parvenir, on procédera par dichotomie, c'est-à-dire que l'on divisera l'intervalle de recherche par deux à chaque étape. Soit m le milieu de $[a,b]$. Si $f(m)$ et $f(a)$ sont de mêmes signes, alors le zéro recherché est dans l'intervalle $[m,b]$, sinon il est dans l'intervalle $[a,m]$.

Exercice 5 : L'intégrale d'une fonction

Écrire une fonction « `Intégrale` » qui retourne la valeur de l'intégrale d'une fonction f réelle continue sur l'intervalle sur $[a, b]$. Soit $f(x)$ une fonction continue sur l'intervalle $[a, b]$,

l'intégration consiste à découper cet intervalle en n sous intervalles de longueur Δ . L'intégrale d'un sous intervalle $[x, x + \Delta]$ est approximée au trapèze de base Δ et de côtés $f(x)$ et $f(x + \Delta)$.

Exercice 6 : Proposer un algorithme permettant de calculer la racine carré d'un nombre réel.

Travaux Dirigés 3

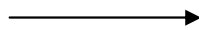
Exercice 7 : Trier trois nombres entiers entrés par l'utilisateur.

Méthode : on cherche à utiliser le moins de comparaisons et de permutations possibles.

Exercice 8 : Le tri à bulles

Méthode : À chaque itération, on permute deux à deux les éléments adjacents du tableau de telle manière que les nombres les plus élevés remontent progressivement sur la droite du tableau.

Lecture du tableau



10	4	35	3	8
4	10	3	8	35
4	3	8	10	35
3	4	8	10	35
3	4	8	10	35

Exercice 9 : Le Quick Sort

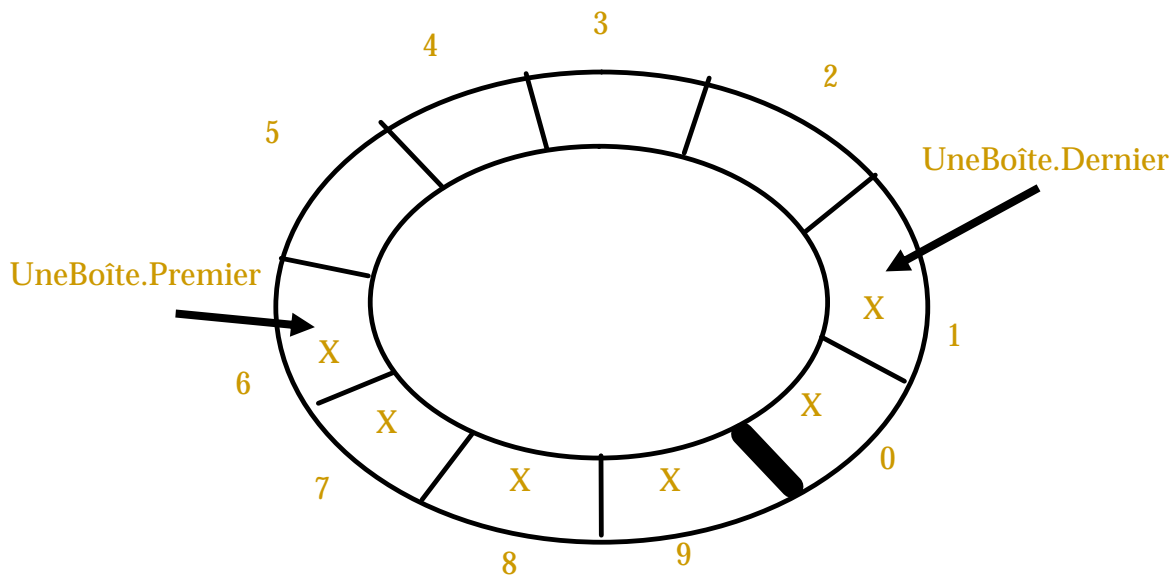
Méthode : Cet algorithme est récursif. Après avoir choisi un pivot, on permute deux à deux les éléments du tableau de telle manière que les nombres supérieurs au pivot se retrouvent placés à sa droite et que les nombres inférieurs au pivot se retrouvent placés à sa gauche.

16	18	10	4	9
9	4	10	18	16
4	9	10	16	18
4	9	10	16	18

Travaux Dirigés 4

Exercice 11 : Boîte aux lettres de message

On souhaite gérer une boîte aux lettres de message par un tableau circulaire :



Chaque message est une chaîne de caractères. La boîte aux lettres est une structure composée du champ `NombreMaximum` (nombre maximum de messages qui est égal à 10 dans l'exemple ci-dessus), du champ `Premier` (indice du premier message reçu non lu), du champ `dernier` (indice du dernier message reçu) et du champ `T` qui est un tableau de `NombreMaximum` de chaînes de caractères.

Question 1 : Ecrire la structure `BoîteAuxLettres`

Chaque fois qu'un message est reçu, si la `FILE` n'est pas pleine alors le message est sauvegardé ; il est de ce fait ajouté dans la `FILE`. Lorsque le propriétaire de la boîte aux lettres désire lire un message, si la `FILE` n'est pas vide alors le premier message reçu non lu est retiré de la `FILE`. Bien sûr, une gestion intelligente des indices `UneBoîte.Premier` et `UneBoîte.Dernier` permet de rendre le tableau circulaire.

Question 2 : Ecrire en pseudo-code les sous-programmes `Créer`, `Vide`, `Pleine`, `Ajout` et `Retrait`.

`Créer (UneBoîte : BoîteAuxLettres)`

booléen `Vide (UneBoîte : BoîteAuxLettres)`

booléen `Pleine (UneBoîte : BoîteAuxLettres)`

`Ajout (UneBoîte : BoîteAuxLettres, LeMessage : chaîne de caractères)`

chaîne de caractères `Retrait (UneBoîte : BoîteAuxLettres)`

On appelle **hauteur** (ou profondeur) d'un arbre son nombre de niveaux. Par exemple, la hauteur de l'arbre ci-dessus est de 4. On atteint le dernier niveau en parcourant le chemin 1, 3, 6, 7.

Quelle **définition récursive** entre un sommet père et son fils gauche et son fils droit peut-on formuler pour calculer la hauteur d'un arbre binaire ? À partir de la définition récursive, écrire l'algorithme récursif qui calcule la hauteur d'un arbre binaire.

Travaux Dirigés 5

Exercice 13 : Le tri topologique – diagramme de Gantt

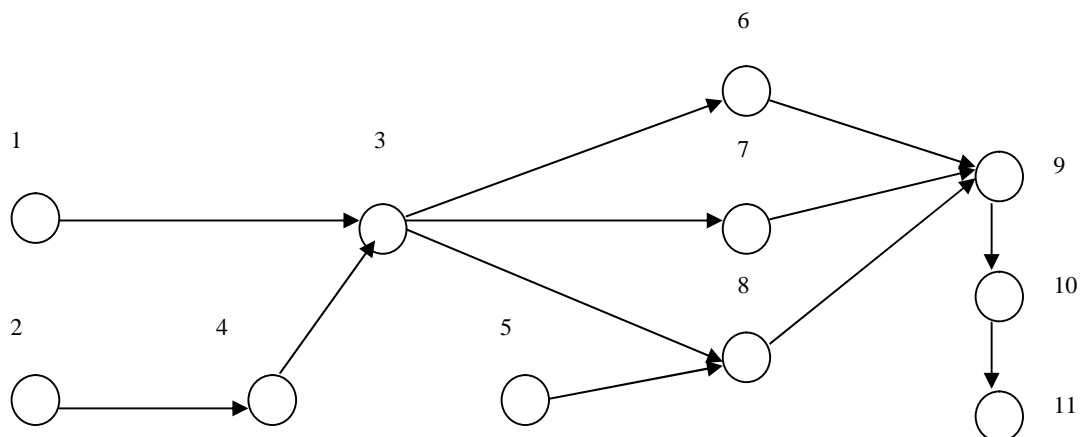
Dans une organisation du travail où certaines tâches doivent être exécutées avant d'autres, on peut schématiser l'ordonnancement des tâches par un graphe où les sommets sont des tâches et où il existe un arc entre deux tâches t_i et t_j si (et seulement si) t_i doit être terminée avant d'exécuter t_j . Choisir une structure de données de type matrice d'adjacence pour représenter le graphe d'ordonnancement des tâches et écrire une procédure sous forme libre (pseudo-code) qui réalise un tri topologique (un diagramme de Gantt). La méthode consiste à trouver un ordre des tâches tel que toute tâche t_i soit exécutée avant toute tâche t_j si il existe un arc allant de t_i à t_j .

Méthode : L'algorithme consiste à extraire un sommet du graphe au fur et à mesure que la tâche associée est traitée ; il peut être décomposé en plusieurs étapes.

(1) Critère d'arrêt de l'algorithme : lorsque le graphe ne comporte plus de sommets (idem le graphe est vide).

(2) Calculer ou remettre à jour le nombre de prédécesseurs des sommets du graphe.

(3) Tant que le graphe n'est pas vide, à chaque itération, déterminer le nombre de tâches à traiter (correspond à un sommet sans prédécesseur). Pour chaque sommet i à traiter, écrire la tâche t_i associée et enlever le sommet i du graphe. Enlever chaque arc allant de t_i à t_j , ce qui revient à mettre un 0 à la place d'un 1 dans la matrice d'adjacence.



Correspondant aux activités et aux délais de la préparation du curry d'agneau :

Sommets	Nom de l'activité	Délai de réalisation
1	Rouler la viande dans le curry	6 minutes
2	Couper les oignons	7 minutes
3	Ajouter la viande dans la cocotte	1 minute
4	Faire revenir les oignons dans la cocotte	10 minutes
5	Râper les pommes	8 minutes
6	Ajouter la noix de coco	1 minute
7	Ajouter le yaourt	1 minute
8	Ajouter les pommes	1 minute
9	Mélanger	5 minutes
10	Rectifier l'assaisonnement	2 minutes
11	Laisser cuire	20 minutes

Appliquer l'algorithme sur la préparation du curry d'agneau. La matrice d'adjacence correspondante est :

	« 1 »	« 2 »	« 3 »	« 4 »	« 5 »	« 6 »	« 7 »	« 8 »	« 9 »	« 10 »	« 11 »
« 1 »	0	0	1	0	0	0	0	0	0	0	0
« 2 »	0	0	0	1	0	0	0	0	0	0	0
« 3 »	0	0	0	0	0	1	1	1	0	0	0
« 4 »	0	0	1	0	0	0	0	0	0	0	0
« 5 »	0	0	0	0	0	0	0	1	0	0	0
« 6 »	0	0	0	0	0	0	0	0	1	0	0
« 7 »	0	0	0	0	0	0	0	0	1	0	0
« 8 »	0	0	0	0	0	0	0	0	1	0	0
« 9 »	0	0	0	0	0	0	0	0	0	1	0
« 10 »	0	0	0	0	0	0	0	0	0	0	1
« 11 »	0	0	0	0	0	0	0	0	0	0	0

Exercice 14 : Recherche du chemin le plus long (chemin critique) dans un graphe

Dans une séquence d'activités où certaines tâches doivent être exécutées avant d'autres, on peut schématiser l'ordonnancement des tâches par un graphe où les sommets sont des tâches et où il existe un arc entre deux tâches t_i et t_j si (et seulement si) t_i doit être terminée avant d'exécuter t_j . Proposez un algorithme pour calculer le chemin le plus long dans un graphe depuis un sommet source jusqu'à un sommet destination.

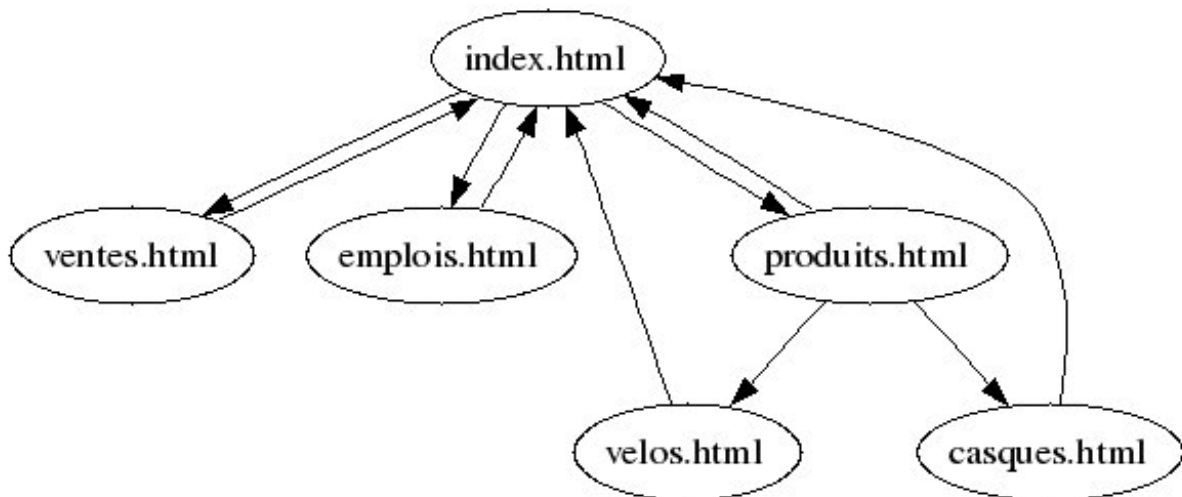
Méthode : Si le graphe ne comporte pas de circuits, il est possible d'appliquer tout simplement un algorithme de parcours en profondeur.

Exercice 15 : L'algorithme PageRank

Cet algorithme est utilisé par le moteur de recherche « Google ». Suite à une requête constituée de mots-clés, le moteur de recherche propose un ensemble de liens vers des pages internet dont l'ordre a été calculé par l'algorithme PageRank. Pour simplifier le fonctionnement de cet algorithme, plus une page internet est référencée, plus sa qualité est jugée comme étant importante et plus la page se situe en tête de la liste des pages présentées. Le tri s'effectue par ordre d'intérêt décroissant dépendant des **indices de popularité**.

Internet peut être vu comme un graphe orienté $G = \langle S, A \rangle$ où S (ensemble de sommets) est l'ensemble des pages et A (ensemble d'arcs) est l'ensemble des liens entre les pages. Un arc part de la page où se trouve le lien et va vers la page indiquée.

Soit, par exemple :



Question 1 : À partir de quelle structure de données, de type matrice, peut-on représenter un graphe internet ?

La mesure de l'**indice de popularité** de l'algorithme PageRank est basée intuitivement sur les principes suivants :

- les pages recevant le plus de liens sont les pages les plus importantes,
- un lien en provenance d'une page importante est plus significatif qu'un lien en provenance d'une page peu importante,
- si une page a beaucoup de liens, chaque lien a moins de valeurs.

Ce qui mathématiquement, peut s'écrire :

Étant donné une page internet u , si E_u est la liste des pages ayant un lien vers u , v une page appartenant à E_u avec N_v le nombre total de pages référencées par v , alors l'indice de popularité (**PageRank**) de u est :

$$PR(u) = \sum_{v \in E_u} \frac{PR(v)}{N_v}$$

Question 2 : Quels calculs faut-il appliquer sur la structure de données de la question 1 pour prendre en compte le fait que si une page a beaucoup de liens, chaque lien a moins de valeurs (ce qui revient à calculer et à introduire les N_v) ?

Pour l'exemple ci-dessus :

	« 1 »	« 2 »	« 3 »	« 4 »	« 5 »	« 6 »
« 1 »	<u>0</u>	<u>1/3</u>	<u>1/3</u>	<u>1/3</u>	<u>0</u>	<u>0</u>
« 2 »	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
« 3 »	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
« 4 »	<u>1/3</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1/3</u>	<u>1/3</u>
« 5 »	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
« 6 »	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>

Question facultative : Comment faut-il désormais procéder pour obtenir l'indice de popularité $PR(u)$ de chaque page (cela amène nécessairement à réitérer les calculs pour prendre en compte qu'un lien en provenance d'une page importante est plus significatif qu'un lien en provenance d'une page peu importante) ?