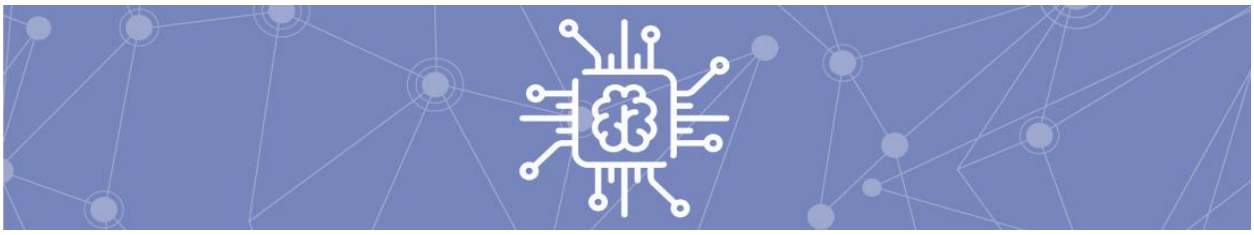
REPUBLIQUE DEMOCRATIQUE DU CONGO
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET UNIVERSITAIRE
UNIVERSITE DE KINSHASA

FACULTE POLYTECHNIQUE

ALGORITHME ET PROGRAMMATION



GROUPE-20 : MUKABI NGOMBO NELSON 2GEI

ASIFIWE NYAMWOGA 2GEI

WATENDA SONGI 2GEI

Assistant. MOBISA

Année académique : 2022-2023

Répondre aux questions suivantes (révision de la matière):

1. Qu'est-ce qu'un algorithme?

   Réponse :

   Un **algorithme** est une suite finie et non ambiguë d'instructions ou de méthode permettant de résoudre une classe de problèmes

2. Qu'est-ce qu'un algorithme efficace ?

   Réponse :

   Un algorithme est efficace s'il utilise avec parcimonie les ressources dont il dispose, c'est-à-dire le temps CPU, la mémoire vive et (objet de recherches récentes) la consommation électrique.

3. Que pouvez-vous dire à propos de l'efficacité d'un algorithme?

   Réponse :

Quelle que soit leur puissance théorique, les machines informatiques réelles sont soumises à des limitations physiques touchant à la puissance de calcul, c'est-à-dire le nombre d'opérations élémentaires pouvant être effectuées chaque seconde, ainsi qu'à la mémoire disponible, c'est-à-dire la quantité d'informations qu'un programme peut avoir à disposition, ou auxquelles il peut accéder à tout moment en un temps raisonnable.

On peut ainsi évaluer le « coût » d'une opération informatique ou d'un calcul, au sens large, par le temps et la mémoire que nécessite son exécution. Une part importante de la recherche en algorithmique consiste à élaborer des algorithmes de plus en plus efficaces, c'est-à-dire ayant un « coût » le plus faible possible. Il apparaît souvent qu'un effort d'analyse important au moment de la conception permet de mettre au point des algorithmes extrêmement puissants vis-à-vis des applications, avec des gains de temps exceptionnels.

L'efficacité d'un algorithme est mesurée notamment par :

- Sa durée de calcul ;
- Sa consommation de mémoire vive (en partant du principe que chaque instruction a un temps d'exécution constant) ;
- la précision des résultats obtenus (par exemple avec l'utilisation de méthodes probabilistes) ;
- sa scalabilité (son aptitude à être efficacement parallélisé) ;
- etc.

4. Citer quelques unes des techniques de conception d'un algorithme?

   Réponse :

Parmi les techniques de conception nous avons :

Invariants ; récursivité ; essais successifs ; méthodes PSEP ; algorithmes gloutons ; diviser pour régner ; programmation dynamique ; Séquence ; Sélection ; Répétition ; les structures de données linéaires (liste chaînées) ; Les arbres ; Les graphes.

5. Commentez en quelques phrases les techniques de conception des algorithmes suivantes: la méthode de la force brute, la méthode gloutonne, la méthode du diviser pour régner, la méthode probabiliste, la méthode de la programmation dynamique

Réponse :

- La recherche exhaustive ou recherche par force brute est une méthode algorithmique qui **consiste principalement à essayer toutes les solutions possibles**. Par exemple pour trouver le maximum d'un certain ensemble de valeurs, on consulte toutes les valeurs.

- Les algorithmes gloutons constituent une alternative dont le résultat n'est pas toujours optimal. Plus précisément, **ces algorithmes déterminent une solution optimale en effectuant successivement des choix locaux, jamais remis en cause**.

- Le **diviser pour régner** est **une méthode algorithmique** basée sur le principe suivant : On prend un problème (généralement complexe à résoudre), on **divise** ce problème en une multitude **de** petits problèmes, l'idée étant que les "petits problèmes" seront plus simples à résoudre que le problème original.

- un algorithme probabiliste, ou algorithme randomisé, **est un algorithme qui utilise une source de hasard**. Plus précisément le déroulement de l'algorithme fait appel à des données tirées au hasard.

- Un algorithme de programmation dynamique **résout chaque sous-sous-problème une seule fois et mémorise sa réponse dans un tableau, évitant ainsi le recalcul de la solution chaque fois qu'il résout chaque sous-sous-problème**

6. Qu'est-ce qu'un pseudo-code? Qu'est-ce qu'un ordinogramme? De quelle autre façon peut-on présenter un algorithme?

Réponse :

- le **pseudo-code**, également appelé **LDA** (pour **Langage de Description d'Algorithmes**) est une façon de décrire un algorithme en langage *presque naturel*, sans référence à un langage de programmation en particulier.

- Un **organigramme de programmation** (parfois appelé **algorigramme** , **logigramme** ou plus rarement **ordinogramme**) est une représentation graphique normalisée de l'enchaînement des opérations et des décisions effectuées par un programme d'ordinateur.

- Les autres façons de représenter un algorithme est d'utiliser un Langage de Programmation

7. Pour quelles raisons une équipe de développeurs de logiciels choisit-elle de représenter les algorithmes par du pseudo-code, des organigrammes ou des bouts de code.

   Réponse :

   Les raisons sont les suivantes :

   - Pour le cas de l'utilisation du pseudo-code, elle permet de décrire facilement un algorithme avec un vocabulaire simple et sans connaissance à priori du langage de programmation ; Elle permet souvent de bien prendre toute la mesure de la difficulté de la mise en œuvre de l'algorithme, et de développer une démarche structurée dans la construction de celui-ci.
   - Pour le cas de L'organigramme, elle permet de visualiser facilement les blocs du programme, les boucles, les tests et les erreurs.

8. En général pour un problème donné, on peut développer plusieurs algorithmes. Comment identifier le meilleur algorithme de cet ensemble?

   Réponse :

   On l'identifie en faisant l'analyse théorique ou l'analyse expérimentale de chaque algorithme et voir celui qui a un meilleur temps d'exécution.

9. En quoi consiste l'analyse d'un algorithme?

   Réponse :

   L'analyse algorithmique permet de prédire l'évolution en temps calcul nécessaire pour amener un algorithme à son terme, en fonction de la quantité de données à traiter.
   Elle consiste aussi à déterminer la quantité de ressources en temps et en espace nécessaires à son exécution

10. Quelles sont les deux méthodes d'analyse d'un algorithme?

    Réponse :

    -L'analyse théorique
    - l'analyse expérimentale

11. Quels sont les inconvénients de la méthode expérimentale?

    Réponse :

    -L'inconvénient est que les expériences sont faites que sur un nombre limité d'entrées ;
    -Il est dur de comparer les temps d'exécutions expérimentaux de deux algorithmes sauf si les expériences ont été menées sur les mêmes environnements ;
    -l'implémentation et l'exécution de l'algorithme en vue d'étudier ses performances.

12. En quoi consiste la méthode des opérations primitives?

    Réponse :

    ce sont des opérations de bat-niveau qui consiste à exécuter une à la suite de l'autre les opération d'un algorithme de façon continue.On ne peut pas arbitrairement changer cette séquence.

13. Qu'est-ce que la complexité d'un algorithme?

    Réponse :

    La complexité d'un algorithme est la quantité de ressources nécessaires pour traiter des entrées. On la voit comme une fonction de n, la taille de l'entrée

14. En quoi consiste la notation asymptotique?

Réponse :

Les **notations asymptotiques** consiste d'analyser l'ordre de grandeur du temps d'exécution d'un **algorithme** en identifiant son comportement à mesure que les données d'entrée de l'**algorithme** augmentent. On appelle également cela le taux de croissance d'un **algorithme**.

15. Quelles sont les fonctions qui apparaissent le plus lors de l'analyse théorique des algorithmes?

Réponse :

Parmi les fonctions nous avons :
-les fonctions constantes
- les fonctions logarithmiques
- les fonctions linéaires
- les fonctions nlogn
- les fonctions quadratiques
- les fonctions cubiques
- les fonctions polynômes ($n \geq 4$)
- les fonctions exponentielles.

16. Quel est l'algorithme le plus efficace parmi un ensemble d'algorithmes permettant de résoudre un problème

Réponse :

C'est l'algorithme qui possède un moindre temps d'exécution

17. Pour évaluer expérimentalement un algorithme on doit l'implémenter et lui fournir des entrées différentes question de mesurer le temps d'exécution correspondant à chaque entrée. C'est en dessinant la courbe du temps d'exécution en fonction de la taille de l'entrée que l'on peut identifier la fonction correspondant à l'évolution du temps d'exécution en fonction de la taille d'entrée. La notion de la taille d'une entrée est très importante. Pourriez-vous la définir en quelques mots et donner quelques exemples de taille d'entrée pour des problèmes simples.

Réponse :

La taille d'entrée prend tout son sens en analyse expérimentale étant donné que plus grande elle est, plus longue sera la mesure du temps d'exécution.
Elle est principalement l'élément qui déterminera le temps d'exécution d'un algorithme ; le facteur multiplicateur qui rallonge le temps d'exécution. On a comme cas le plus parlant de tri par insertion, plus grande est l'entrée et plus long sera le tri ; la recherche de données dans la recherche binaire, etc.

18. Dans l'analyse d'un algorithme on distingue généralement le cas le plus défavorable, le cas le plus favorable et le cas moyen (probabiliste). Expliquez en quoi consiste chaque cas. Pourquoi le cas le plus défavorable a une importance particulière?

Réponse :

Le cas le plus mieux est le cas optimal. C'est-à-dire que dans ce cas on ne peut pas résoudre le problème en moins de temps.

Le cas défavorable est la pire des situations, c'est-à-dire que dans ce cas on ne peut pas résoudre le problème en plus de temps.
Et le cas modéré se situe entre le cas le plus mieux et le cas défavorable.

19. Définir en quelques mots le concept de récursivité.

Réponse :

La récursivité est un processus par lequel une fonction s'appelle elle-même au cours de son exécution.

20. En quoi consistent la récursivité linéaire, la récursivité binaire et la récursivité multiple.

Réponse :

-La récursivité linéaire consiste en ce que pour chaque invocation du corps, l'algorithme fasse au plus un nouvel récursif. (Récursivité type factoriel)

-La récursivité binaire consiste en ce que la fonction fasse deux appels récursifs

-La récursivité multiple consiste en ce que la fonction fasse plusieurs appels récursifs au moins plus de 2 fois.

21. De quelle façon un problème récursif doit-il pouvoir se définir? Donnez un exemple.

Réponse :

Elle se définie en une fonction s'appelant elle-même avec un cas de base pour terminer la boucle infinie
Pour exemple :

En mathématiques [modifier | modifier le code]

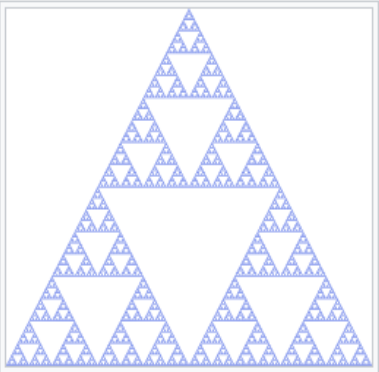
Suite définie récursivement [modifier | modifier le code]

Article détaillé : Définition récursive d'une suite.

Fonctions récursives [modifier | modifier le code]

Articles détaillés : Algorithme récursif et Fonction récursive.

Une fonction peut être définie en fonction d'elle-même. Un exemple familier est la suite de Fibonacci vue comme une fonction $F : \mathbb{N} \to \mathbb{N}$, à savoir $F(n) = F(n-1) + F(n-2)$. Pour qu'une telle définition ait un sens, elle doit conduire à des valeurs immédiatement évaluables: dans le cas de la suite de Fibonacci on pose $F(0) = 0$ et $F(1) = 1$.



Le triangle de Sierpiński — une récurrence de triangles formant une fractale.