UNIVERSITE DE KINSHASA

FACULTE POLYTECHNIQUE

Deuxième Graduat GCEIM

TP D'ALGO & PROGRAMMATION

PROJET 2 (Questions théoriques)

Dirigé par :

PROF OLAMBA

Assistant MOBISA

GROUPE 12 :

MAVINGA MAVINGA 2GC

MPUMBU MUYAYA 2GC

NGALAMULUME KABUE 2GC

ANNEE ACADEMIQUE 2021-2022

TP D'ALGORITHMIQUE ET PROGRAMMATION GROUPE 12 G2 2023

Fiche 2: Les fondements mathématiques de l'algorithmique et l'évaluation des algorithmes

Répondre aux questions suivantes (révision de la matière):

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

R/ De manière informelle un algorithme se définit comme étant une procédure de calcul qui prend en entrée une valeur, ou un ensemble de valeurs, et qui donne en sortie une valeur ou un ensemble de valeurs. Un algorithme est donc une séquence d'étapes de calcul qui transforme une entrée en une sortie.

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

R/ C'est un algorithme qui a un temps d'exécution minimal.

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

R/ Il arrive que des algorithmes différents conçus pour résoudre le même problème diffèrent fortement entre eux en termes d'efficacité. Ces différences bien souvent sont plus importantes que celles dues au matériel et au logiciel. Elles proviennent donc de la nature même des algorithmes. C'est en analysant expérimentalement ou théoriquement cette valeur intrinsèque des algorithmes que nous pourrons identifier les algorithmes les plus efficaces.

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

R/ La méthode de la force brute, la méthode gloutonne, la méthode du diviser pour régner, la méthode probabiliste, et l'approche par la programmation dynamique.

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.

R**/ La méthode de la force brute** est une approche qui consiste à essayer toutes les solutions possibles. Par exemple, pour trouver le minimum, d'un ensemble de nombres, on essaye tous les nombres jusqu'à ce que l'on obtienne le minimum.

Dans **la méthode gloutonne**, on construit une solution de manière incrémentale en optimisant de manière aveugle un critère local. Un algorithme glouton est un algorithme qui étape par étape fait le choix d'un optimum local. Dans certains cas, cette approche permet d'arriver à un optimum global, mais dans le cas général c'est une heuristique.

Dans l'approche du diviser pour régner, le problème à résoudre est diviser en sous-problèmes semblables au problème initial, mais de taille moindre. Ensuite les sous-problèmes sont résolus de manière récursive et enfin les solutions des sous-problèmes sont combinées pour avoir la solution du problème original. Le paradigme du diviser pour régner implique trois étapes à chaque niveau de récursivité, à savoir : diviser, régner et combiner.

La méthode probabiliste fait appel aux nombres aléatoires. Un algorithme est dit probabiliste lorsqu'il fait des choix aléatoires au cours de son exécution. Un tel algorithme fait appel à un ou plusieurs générateurs de nombres aléatoires.

L'approche par programmation dynamique : la solution optimale est trouvée en combinant des solutions optimales d'une série de sous-problèmes qui se chevauchent. On matérialise un algorithme par la production d'un pseudo-code, d'un organigramme ou d'une implémentation dans un langage de programmation.

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

R/ Le pseudo-code appelé également Langage de Description d'Algorithmes (LDA) est une façon de décrire un algorithme sans référence à un langage de programmation particulier.

Un organigramme, également appelé algorigramme, logigramme ou ordinogramme est une représentation graphique normalisée des opérations et des décisions effectuées par un ordinateur.

On peut aussi représenter un algorithme **par une implémentation en 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.

R/ Selon le contexte, les objectifs de la démarche, les compétences du programmeur et les attentes du public cible, l'équipe concevra et présentera les algorithmes selon l'une des trois façons ci-dessus mentionnées.

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

R/ On identifiera le meilleur comme étant celui qui a le temps d'exécution le plus petit.

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

R/ Elle consiste à étudier son temps d'exécution en fonction de la taille de l'entrée.

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

R/ L'analyse expérimentale et l'analyse théorique.

10. Quels sont les inconvénients de la méthode expérimentale?

R/ Les expériences ne peuvent être faites que sur un nombre limité d'entrées (d'autres entrées pouvant se révéler importantes sont laissées de côté);

Il est difficile de comparer les temps d'exécution expérimentaux de deux algorithmes sauf si les expériences ont été menées sur les mêmes environnements (Hardware et Software);

On est obligé d'implémenter et d'exécuter un algorithme en vue d'étudier ses performances. Cette dernière limitation est celle qui requiert le plus de temps lors d'une étude expérimentale d'un algorithme.

11. En quoi consiste la méthode des opérations primitives?

R/ s'appuie sur des opérations de base telles que :

- Assigner une valeur à une variable;

- Effectuer une opération arithmétique (exemple : additionner deux nombres);
- Comparer deux nombres;
- Indexer un tableau;
- Suivre la référence d'un objet;
- Sortir d'une méthode.

En fait, une opération primitive correspond à une instruction de bas-niveau avec un temps d'exécution constant. Pour déterminer le temps d'exécution d'un algorithme, il suffit de compter le nombre d'opérations primitives exécutées et pour chaque opération primitive de multiplier ce nombre par son temps d'exécution constant. Il y aura une corrélation entre ce temps d'exécution présumé et le temps d'exécution sur une machine spécifique. A chaque opération primitive correspond un temps d'exécution constant et il n'y a qu'un nombre limité d'opérations primitives. Ainsi, le nombre d'opérations primitives exécutées par un algorithme sera proportionnel au temps d'exécution de cet algorithme.

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

R/ La complexité d'un algorithme est une mesure du temps et de l'espace nécessaires pour exécuter un algorithme, en fonction de la taille de son entrée.

13. En quoi consiste la notation asymptotique?

R/ Elle consiste à décrire la croissance relative de la complexité d'un algorithme en fonction de la taille de son entrée.

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

R/ $\boldsymbol{n}, \boldsymbol{\log n}, \boldsymbol{n\ \log n}, \boldsymbol{n^2}, \boldsymbol{n^3}, \boldsymbol{2^n}$

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

R/ Le bon algorithme, mieux l'algorithme efficace est celui qui résout le problème en un temps minimal.

16. 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.

R/ Le sens exact de « taille d'une entrée » dépend du problème à résoudre. Pour de nombreux problèmes tels que le filtrage, le calcul de la transformée de Fourrier discrète, le sens le plus naturel pour « taille d'entrée » est le nombre d'éléments constituant l'entrée, par exemple la longueur n du tableau à filtrer.

Pour beaucoup d'autres problèmes tels que la multiplication de deux entiers, la meilleure mesure de la taille de l'entrée est le nombre total de bits nécessaires à la représentation de l'entrée dans la notation binaire habituelle. Parfois il est plus approprié de décrire une entrée avec deux nombres au lieu d'un seul. Par exemple si l'entrée d'un algorithme est un repère orthonormé, on pourra décrire la taille de l'entrée par l'axe des abscisses et l'axe des ordonnés.

17. 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?

R/ Le cas le plus défavorable décrit la situation où l'algorithme prend le plus de temps pour accomplir sa tâche (maximal). Le temps d'exécution associé au cas le plus défavorable est une borne supérieure du temps d'exécution associée à une entrée quelconque. Connaitre cette valeur nous permettra donc d'avoir la certitude que l'algorithme ne mettra jamais plus de temps que cette limite.

Le cas le plus favorable décrit la situation où l'algorithme prend le moins de temps pour accomplir sa tâche (minimal).

Le cas moyen décrit la performance de l'algorithme en termes de la complexité moyenne des entrées possibles. Cela implique de prendre en compte toutes les entrées possibles et de les pondérer en fonction de leur probabilité d'occurrence. Le but est de déterminer la performance moyenne de l'algorithme.

18. Définir en quelques mots le concept de récursivité.

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

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

R/ Si un appel récursif fait un et un seul autre appel récursif lors de son exécution on parle de récursivité linéaire.

Si un appel récursif déclenche deux autres appels récursifs au cours de son exécution, on parle de récursivité binaire.

Si un appel récursif déclenche au moins trois autres appels récursifs au cours de son exécution, on parle de récursivité multiple.

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

R/ Un problème récursif doit pouvoir se définir de façon à ce que la solution d'une partie du problème soit utilisée pour résoudre une partie plus petite du même problème. Cela crée une décomposition répétitive du problème en sous-problèmes plus petits jusqu'à ce qu'ils atteignent une taille minimale qui peut être résolue directement.

Un exemple de problème récursif est la recherche d'un élément dans une liste triée. La solution consiste à déterminer la valeur médiane de la liste et à la comparer à l'élément recherché. Si l'élément est égal à la valeur médiane, la recherche est terminée. Si l'élément est plus petit, la recherche est continuée dans la première moitié de la liste. Si l'élément est plus grand, la recherche est continuée dans la seconde moitié de la liste. Cette démarche peut être répétée jusqu'à ce l'élément soit trouvé ou que la liste soit vide. Voilà l'exemple.