TP D'ALGORITHMIQUE ET PROGRAMMATION 21-22/PARTIE THEORIQUE

REPONSES

1. Un algorithme est une suite d'étapes de calcul qui transforme une entrée en une sortie.
2. Un algorithme est dit efficace lorsque son temps d'exécution (le temps qu'il met pour résoudre un problème) est minimal
3. L'efficacité d'un algorithme repose sur le temps minimal qu'il peut prendre pour exécuter une opération donnée. On distingue les algorithmes à temps d'exécution polynomial et ceux à temps d'exécution exponentiel.
4. Il existe de nombreuses techniques pour concevoir un algorithme ; nous avons notamment : *la méthode gloutonne, la méthode de force brute, la méthode du diviser pour régner, la méthode probabiliste et l'approche par la programmation dynamique.*
5. La méthode de la force brute est une approche qui consiste à essayer toutes les solutions possibles ;

   La méthode gloutonne, on construit une solution de manière incrémentale en optimisant de manière aveugle un critère local ;

   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 Enon les solutions des sous-problèmes sont combinés pour avoir la solution du problème original ;

   La méthode probabiliste fait appel aux nombres aléatoires. Un algorithme est dit probabiliste lorsqu'il fait des choix aléatoires au cours de son exécution ;

L'approche par programmation dynamique : la solution optimale est trouvée en combinant des solutions optimales d'une série de sous-problèmes qui se chevauchent ;

6. Le pseudo-code (Langage de Description d'Algorithmes (LDA)) est une façon de d´écrire un algorithme sans référence à un langage de programmation particulier ;

   Un organigramme, ´également appelé logigramme, logigramme ou ordinogramme est une représentation graphique normalisée des opérations et des décisions exécutées par un ordinateur ;

   Une autre façon de présenter un algorithme est soit par un pseudo-code, soit par un organigramme ou encore par une implémentation en langage de programmation.

7. Une équipe de développeurs choisissent ces trois méthodes suivant :le contexte, les objectifs de la démarche, les compétences du programmeur et les attentes du public cible.
8. Pour identifier le meilleur algorithme parmi un ensemble, on se réfère à leur temps d'exécution ; le meilleur algorithme sera celui ayant un temps d'exécution minimal.
9. L'analyse d'un algorithme consiste à étudier son temps d'exécution en fonction de la taille de l'entrée en mesurant de manière expérimentale le temps d'exécution.
10. Les deux méthodes d'analyse d'un algorithme sont : la méthode expérimentale et l'analyse théorique
11. Les inconvénients ou limitations de la méthode expérimentale sont :
    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.
12. La méthode des opérations primitives consiste à décomposer un algorithme en opérations élémentaires, appelées opérations primitives, et à mesurer le nombre de ces opérations en fonction de la taille de l'entrée. Les opérations primitives peuvent être des opérations telles que des additions, des multiplications, des comparaisons, des accès à la mémoire, etc.
13. La complexité d'un algorithme est la mesure du temps nécessaire à l'exécution d'un algorithme en fonction de la taille de sin entrée.
14. La notation asymptotique est une façon de décrire le temps d'exécution d'un algorithme en fonction de la taille n de son entrée.
15. Les fonctions qui apparaissent le plus sont : la fonction constante, logarithme et linéaire.
16. L'algorithme le plus efficace est celui qui résout un problème en un temps minimal
17. 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

18. 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.

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

20. On parle de récursivité linéaire si un appel récursif fait un et un seul appel récursif lors de son exécution

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.

21. Un problème récursif peut 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.

EQUIPE DE TRAVAIL (Tous 2GC)

- KABAMBA MUTELA Joris
- KANYIKI NGANDU Kally
- ENTAMBE EYOFELA Odri