FACULTÉ POLYTECHNIQUE

Deuxième Graduat

TP D'ALGORITHME ET PROGRAMMATION

Réalisé par:

- ASIMWE NDRUUDJO GLOIRE (2GC)
- IMANI UKURANGO CHRISTIAN (2GC)
- HESHIMA MASIMIKE DAVID (2GM)

ANNÉE ACADÉMIQUE 2021 - 2022

Question 1

Qu'est-ce qu'un algorithme?

Réponse: Un algorithme est une suite définie d'instructions afin de résoudre un problème ou atteindre un but particulier.

Question 2

Qu'est-ce qu'un algorithme efficace?

Réponse: Un algorithme est dit efficace s'il est capable d'accomplir une tâche dans un temps raisonnable et utilise un minimum de ressources telles que la mémoire et la puissance de calcul.

Question 3

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

Réponse: L'efficacité d'un algorithme peut être mesurée de plusieurs manières, telles que la vitesse d'exécution, l'utilisation de la mémoire, le nombre d'opérations effectuées, etc. Il est important de choisir la mesure appropriée en fonction du contexte et des objectifs spécifiques. De plus, l'efficacité peut varier selon les entrées et les conditions de l'algorithme. Il est également important de prendre en compte le coût de développement et de maintenance de l'algorithme dans l'évaluation de son efficacité globale.

Question 4

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

Réponse: Voici quelques techniques utilisées pour concevoir un algorithme :

1. Décomposition en sous-tâches
2. Brute-force ou méthode d'essai et erreur
3. Recherche dichotomique
4. Programmation dynamique
5. Diviser pour régner
6. la méthode probabiliste
7. Greedy Algorithm (algorithme glouton)
8. Algorithme de tri (comme Tri par sélection, Tri rapide, Tri à bulles,...)

Question 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: Voici les commentaires:

1. **La méthode de la force brute** implique d'essayer toutes les solutions possibles jusqu'à trouver la solution optimale. C'est une méthode simple mais peu efficace pour les grands ensembles de données.

2. **La méthode gloutonne** consiste à prendre les meilleurs décisions à chaque étapes sans se soucier de l'impact sur les étapes ultérieurs. Elle peut donner des résultats optimaux dans certains cas, mais pas toujours.

3. **La méthode du diviser pour régner** implique de diviser le problème en sous-problèmes plus petits et de les résoudre indépendamment avant de les combiner pour trouver la solution au problème d'origine. C'est souvent utilisé pour les problèmes de tri ou de recherche.

4. **La méthode probabiliste** implique l'utilisation de probabilités pour prendre des décisions et trouver des solutions. C'est souvent utilisé dans les algorithmes de recherche heuristique et les algorithmes de machine learning.

5. **La méthode de la programmation dynamique** implique de décomposer en sous-problèmes récurrents et de mémoriser les solutions pour les sous-problèmes pour réutiliser plus tard. C'est souvent utilisé pour les problèmes de optimisation avec des dépendances entre les sous-problèmes.

Question 6

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

Réponse:

Un pseudo-code est une représentation formelle et générique d'un algorithme, écrite en utilisant une syntaxe de programmation simplifiée. Il sert à communiquer clairement les étapes de l'algorithme à des personnes qui ne connaissent pas le code.

Un ordinogramme (Organigramme) est une forme graphique d'un algorithme, qui représente les étapes sous forme de boîtes connectées par des flèches. Chaque boîte représente une étape de l'algorithme et les flèches représentent les dépendances entre les étapes.

Il existe d'autres façons de représenter un algorithme, telles que des algorithmes en langage naturel, des algorithmes formels, des algorithmes flowchart, etc. Tout dépend de la personne qui les utilise et de l'objectif poursuivi.

Question 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: Parce que cela leur permet de développer leurs concepts avec plus de précision et de souplesse.

Question 8

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

Réponse: Pour identifier le meilleur algorithme d'un ensemble, l'évaluation et comparaison de chaque algorithme sont nécessaires. vous devez évaluer chaque algorithme en fonction de son efficacité et de son temps d'exécution, ainsi que d'autres facteurs tels que la mémoire et la facilité de maintenance.

Question 9

En quoi consiste l'analyse d'un algorithme?

Réponse: Elle consiste à étudier les propriétés formelles et les performances d'un algorithme, telles que sa complexité temporelle et sa complexité en mémoire, afin de déterminer ses limites et son efficacité. Cela peut aider ç déterminer le temps nécessaire pour exécuter l'algorithme et la quantité de mémoire nécessaire pour son exécution, ainsi qu'à identifier des opportunités d'optimisation.

Question 10

Quelles sont les deux méthodes d'analyse d'un algorithme

Réponse: Les deux principales méthodes d'analyse d'un algorithme sont l'analyse théorique et l'analyse expérimentale.

Question 11

Quels sont les inconvénients de la méthode expérimentale?

Réponse: elle dépend du temps d'exécution d'un algorithme en fonction des paramètres d'une machine. Les principales inconvénients sont liés au manque de généralité et aux coûts d'exécution qui sont généralement supérieurs à d'autres méthodes.

Question 12

En quoi consiste la méthode des opérations primitives?

Réponse: C'est une méthode de résolution de problèmes mathématiques qui consiste à identifier les principales opérations(additions, soustractions, multiplications et divisions) nécessaires pour convertir les données en une forme plus simple et plus pratique à manipuler.

Question 13

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

Réponse: La complexité d'un algorithme est une mesure du temps ou la mémoire nécessaire pour exécuter un algorithme en fonction de la taille de l'entrée.

Question 14

En quoi consiste la notation asymptotique?

Réponse: Elle consiste à donner le temps d'exécution des algorithmes à l'aide des fonctions polynômes, linéaires, quadratiques, logarithmiques.

Question 15

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

Réponse: Nous avons: la fonction constante, la fonction logarithme, la fonction linéaire, la fonction $n \log n$, la fonction quadratique, la fonction cubique, les autres polynômes et la fonction exponentielle.

Question 16

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

Réponse: L'algorithme le plus efficace est celui qui a un temps d'exécution minimal et pendre un faible espace mémoire.

Question 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'une entrée dans l'évaluation expérimentale d'un algorithme représente la quantité de données ou de ressources que l'algorithme doit traiter pour exécuter une tâche. Cela peut être la longueur d'une chaîne de caractères, le nombre d'éléments dans un tableau, le nombre de nœuds dans un graphe, etc.
Exemples de taille d'entrée pour des problèmes simples:

- Recherche d'un élément dans un tableau: la taille d'entrée serait le nombre d'éléments dans le tableau.
- Trier un tableau: la taille d'entrée serait le nombre d'éléments dans le tableau.
- Calcul de la factorielle d'un nombre: la taille d'entrée serait le nombre lui-même.

Question 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: Chacun de ces cas offre une perspective différente sur la façon dont l'algorithme peut fonctionner. Le cas le plus défavorable est celui où l'algorithme est le plus lent et le plus inefficace. Il est important de comprendre ce cas car il peut nous aider à déterminer le temps et les ressources nécessaires pour exécuter l'algorithme. De plus, il peut nous aider à identifier les points faibles de l'algorithme et à trouver des solutions pour les corriger. Le cas le plus favorable est celui où l'algorithme est le plus rapide et le plus efficace. Cette analyse nous permet de déterminer le meilleur scénario possible pour l'exécution de l'algorithme. Le cas moyen (probabiliste) est un cas intermédiaire entre les deux premiers. Il est important car il nous permet de déterminer la probabilité que l'algorithme fonctionne correctement dans des conditions réelles. Les cas le plus défavorable a une importance particulière car c'est le plus pire scénario possible pour l'exécution de l'algorithme. Cette analyse nous permet de comprendre les limites de l'algorithme et de trouver des solutions pour le rendre plus efficace.

Question 19

Définir en quelques mots le concept de récursivité.

Réponse: La récursivité est un principe qui consiste à s'appeler soi-même pour résoudre des problèmes complexes et à s'appliquer à une gamme de tâches.

Question 20

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

Réponse:

- La récursivité linéaire se réfère à une fonction récursive qui s'appelle elle-même une seule fois pour résoudre une tâche divisée en sous-tâches plus petites.
- La récursivité binaire implique une fonction qui s'appelle elle-même deux fois pour résoudre une tâche en sous-tâches encore plus petites.
- la récursivité multiple se produit lorsqu'une fonction s'appelle plusieurs fois de manière récursive pour résoudre une tâche complexe.

Question 21

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

Réponse: Un problème récursif peut être défini comme un problème dans lequel la solution de base dépend d'un ou plusieurs sous-problèmes similaires.
Comme exemple on a le factoriel.