UNIVERSITE DE KINSHASA

FACULTE POLYTECHNIQUE

DEPARTEMENT DE GENIE ELECTRIQUE ET INFORMATIQUE

TRAVAIL PRATIQUE DU COURS D'ALGORITHMIQUE ET PROGRAMMATION

Par :
MFWAMBA NDAYA (2GEI)
SHIMBA ILUNGA (2GEI)
BADIBANGA BADIBANGA (2GEI)

Dirigé par l'assistant **MOBISA**

Année académique 2021-2022

QUESTIONS

Question 1 Qu'est ce qu'un algorithme ?

Question 2 Qu'est ce qu'un algorithme efficace ?

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

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

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

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

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 des codes.

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

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

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

Question 11 Quelles sont les incovenients de la méthode expérimentale ?

Question 12 En quoi consiste la méthode des opérations primitives ?

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

Question 14 En quoi consiste la notation asymptotique ?

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

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

Question 17 Pour évaluer expérimentalement un algorithme, on doit lui implementer et lui fournir des entrées différentes question de mésurer le temps d'exécution correspondant à chaque entrée. C'est en dessinant la courbe du temps d'exécution en focntion 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.

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 à une importance particulière ?

Question 19 Définir en quelques mots le concept de récursivité.

Question 20 En quoi,consiste la recursivité linéaire, la recursivité binaire et la recursivité multiple.

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

REPONSES

Question 1 Un algorithme est définit comme étant la description d'une suite d'étapes permettant d'obtenir un résultat à partir d'élements fournis en entrée.

Question 2 Un algorithme est dit *efficace* lorsque les valeurs d'entrée de cette fonction sont petites ou croissent lentement par rapport à une croissance de la taille de l'entrée.

Question 3 Des algorithmes différents peuvent servir à résoudre le même problème. Mais le choix est le plus souvent porter vers l'algorithme le plus efficace. Cela se justifie, comme définit à la question précédente, par le temps minimal que met l'algorithme pour s'exécuter. C'est par des analyses théoriques ou expérimentales que les algorithmes peuvent être identifié comme efficaces.

Question 4 la méthode de la force brute, la méthode gloutonne, la méthode de diviser pour regner, la méthode probabiliste, et l'approche par la programmation dynamique.

Question 5 Les différentes méthodes sont décries de la manière suivante :

- ➢ La méthode de la force brute : est une approche qui consiste à essayer toutes les solutions possibles.
- ➢ La méthode gloutonne : On construit la solution de manière incrémentable en optimisant de manière aveugle un critère local.
- ➢ L'approche diviser pour regner : ici le prolème à résoudre est de diviser en sous-problème 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. Bref, cette approche implique trois étapes à chaque niveau de la récursivité : diviser, régner et combiner.
- ➢ La méthode probabiliste : fait appel aux nombres aléatoires.
- ➢ L'approchepar programmation dynamique : la solution optimale est trouvée en combinant des solutions optimales d'une série de sous-problèmes qui se chevauchent.

Question 6 **Un pseudo-code,** aussi connu sous l'appellation de *langage de description d'algorithmes* est une façcon de décrire un algorithme sans faire référence à un langage de programmation particulier.
Un organigrame, aussi appelé *algorigramme, logigramme ou ordinogramme* est une représentation graphique normalisée des opérations et des décisions effectuées par un ordinateur.
L'algorithme est représenté par une implémentation directe sur un langage de programmation.

Question 7 Ces quelques raisons peuvent être avancées :

- – L'écriture en pseudo-code 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.
- – l'algorigramme permet aux developpeurs des logiciels de visualiser facilement les blocs du programme, les boucles, les tests et les erreurs.

Question 8 C'est au moyen des analyses théorique et expérimentale.

Question 9 L'analyse d'un algorithme consiste à mésurer son temps d'exécution.

Question 10 L'analyse expérimentale et l'analyse théorique

Question 11 La méthode expérimentale présente trois inconvenients majeurs à savoir :

- Les expériences ne peuvent être faites que sur un nombre limité d'entrée(d'autres entrées importantes pouvant être mis de côté)
- il est difficil 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'implementer et d'exécuter un algorithme en vue d'étudier ses performances. Cette limitation est celle qui requiert le plus de temps lors d'une étude exoérimentale d'un algoritthme.

Question 12 Elle consiste en une suite d'opérations de bas niveau qui sont indépendantes du langage de programmation. Nous pouvons citer par exemple : ppel et retour d'une méthode, effectuer une opération arithmétique,Affectation d'une variable,etc.

Question 13 Elle consite comparer l'efficacité des algorithmes pouvant résoudre le même problème. Ce, en évaluant la quantité des réssources nécessaires permettant de traiter les entrées.

Question 14 Les notations asymptotiques consistent à 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.

Question 15 Ces fonctions sont les suivantes :
- La fonction constante ;
- la fonction logarithme ;
- la fonction linéaire ;
- la fonction $\mathbf{nlogn}$;
- la fonction quadratique ;
- la fonction cubique ;
- les autres polynômes et
- la fonction exponentielle.

Question 16 C'est l'algorithme dont le temps d'exécution de son cas le plus defavorable a un ordre de grandeur inférieur.

Question 17 La taille d'une entrée désigne la quantité de données ou la complexité d'un problème doit être résolu par un alogorithme. Il est en outre important dans l'évaluation du temps d'exécution. Plus la taille de l'entrée est grande, plus le temps d'exécution sera elevé.
Exemple :
- pour un algorithme de tri, la taille d'entrée peut être définie par le nombre d'élements dans la liste à trier.
- Pour un algorithme de recherche dans un tableau, la taille d'entrée peut être définie par la taille du tableau.
- pour un algorithme de calcul de la somme des n premiers nombres, la taille d'entrée peut être définie par la valeur de n.

Question 18 Premièrement le cas le plus défavorable correspond à la situation où les hypothèses les plus pessimistes sont considerées quant aux entrées de l'algorithme. Par exemple, le pire cas d'une recherche binaire est celui dont la valeur recherchée ne se trouve pas dans letableau.
Deuxièmement, le cas le plus favorable correspond à la situation où les hypothèses optimistes sont considerées, comme par exemple, le meilleur cas d'une recherche binaire est celui où la valeur recherchée se trouve dans la première case du tableau.
Enfin, le cas moyen(probabiliste) est une analyse qui prend en compte la probabilité statistique

des différentes situations. Par exemple, la moyenne arithmétique des temps d'exécution de l'algorithme pour un grand nombre des données aléatoires.
Le cas le plus défavorable revêt une importance particulière en raison de la necessité de garantir la performance minimale de l'algorithme dans toutes les situations possibles. C'est pourquoi, les algorithmes sont souvent conçus et optimisés en fonction du pire cas.

Question 19 La recursivité est un concept en programmation où une fonction s'appelle elle-même pour résoudre une tâche complexe en la décomposant en sous-tâches plus simples, jusqu'à ce qu'une condition de base soit atteinte. Cette technique de programmation permet de résoudre des problème en utilisant une approche diviser pour regner et peut être uilisée pour implémenter des algorithmes tels les tri par fusion et la recherche dans un arbre.

Question 20

- La récursivité linéaire consiste en un seul appel récursif à la fonction pour résoudre un sous-problème . Cette technique est souvent utilisée pour décomposer un problème en sous-tâches plus petites, chacune étant résolue par seule récursion.
- La récursivité binaire implique deux appels récursifs à la fonction pour résoudre deux sous-problèmes distincts. Cette technique est souvent utilisée pour diviser un problème en deux parties égales et traiter chacune elles séparément, en utilisant des récursions supplémentaires pour les résoudre.
- La récursivité multiple implique plusieurs appels récursifs à la fonction pour résoudre plusieurs sous-problèmes differents. Cette technique est souvent utilisée pour décomposer un problème complexe en plusieurs tâches plus petites, chacune étant résolue par une récursion distincte.

Question 21 Un problème de récursivité doit pouvoir se définir en présentant des tâches repétitives.
Exemple :La recherhe binaire.
C'est un algorithme classique, qui est utilisé pour localiser efficacement une valeur cible dans une séquence d'élements. Nous considérons trois cas :

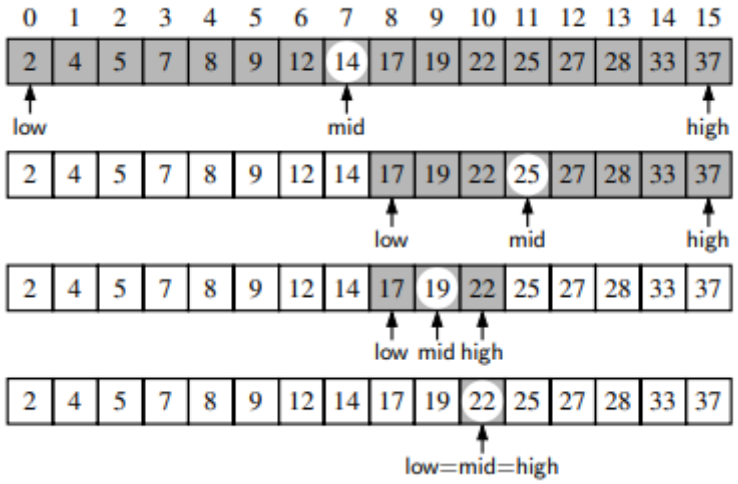

FIGURE 1 –

- Si la cible =donne[milieu], alors nous avons trouvé l'élement recherché ;

- si la cibe < donne [milieu], alors nous avons trouvé sur la première moitié de la séquence, c-à-d sur l'intervalle des indices de bas à milieu -1 ;
- si la cibe > donne [milieu], alors nous avons trouvé sur la seconde moitié de la séquence, c-à-d sur l'intervalle des indices de bas à milieu +1 ;

Alors l'implémentation en python se présente de la manière suivante :

```
1
2  def recherche_binaire(donne,cible,bas,haut):
3      """retourne True si la cible se trouve dans la portion de liste
    indique dans python"""
4
5      if bas > haut:
6          return False
7
8       else:
9           milieu=(bas+haut)//2
10          if cible ==donne[milieu]:
11              return True
12          elif cible <donne[miliru]:
13              return recherche_binaire(donne,cible,bas,milieu-1)
14          else:
15              return recherche_binaire(donne,cible,milieu+1)
16
```

Listing 1 – implémentation en python