TP2 ALGORITHMIQUE ET PROGRAMMATION GR 7 G2 POLYTECH UNIKIN 2021-2022

BAZAYAMA Abel METENA Yves YEKWA John

mardi 07 février 2023

1 Les fondements mathématiques de l'algorithme et l'évaluation des algorithmes

[1] [2]

1.1 Qu'est-ce qu'un algorithme ?

Un algorithme est donc une séquence d'étapes de calcul qui transforme une entrée en une sortie, c'est aussi une proédure, étape par étape, permettant de résoudre un problème dans un intervalle de temps fini. Par exemple, une recette de cuisine est un algorithme permettant d'obtenir un plat à partir de ces ingrédients.

1.2 Qu'est-ce qu'un algorithme efficace ?

Un algorithme efficace est un algorithme qui a un temps d'exécution le plus minimal possible, et qui exploite moins la mémoire.

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

L'efficacité d'un algorithme c'est ce qui nous aide à choisir un algorithme parmi plusieurs, alors cette notion d'efficacité se base sur le temps d'exécution, favorisant le temps d'exécution le plus faible. Souvent on fait la différence entre un temps d'execution polynomial et exponentiel, le polynomial étant plus rapide que l'autre. Prenons un exemple, soient deux algorithmes x et y définis pour résoudre une équation du second degré, le premier (x) donne le résultat après 6 secondes et le deuxième donne le résultat après 6.3 secondes, bien évidement la notion de l'efficacité nous recommande de prendre le premier algorithme (x) car son temps d'execution est minimal par rapport à l'autre.

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

1. La méthode gloutonne
2. La methode probabiliste
3. La methode du diviser pour regner
4. La methode de la force brute
5. La méthode de la programmation dynamique

1. Groupe 7 Deuxième graduat Polytechnique université de Kinshasa
2. Réponse aux questions de la deuxième série des Travaux pratiques du cours d'algorithmique et programmation.

1.5 Commentez en quelques phrases les techniques de conception des algorithmes suivantes :

1. **La methode de la force brute :** c'est est une approche qui consiste à essayer toutes les solutions possibles.
2. **La méthode gloutonne :** ici 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.
3. **La méthode du diviser pour régner :** cette méthode vise à diviser le problème à résoudre en plusieurs sous problèmes semblables au problème principal mais de petite taille. Une fois les petits problemes résolus on combine ses solutions pour en faire la solution du problème principal. On le fait en trois étapes : diviser, régner et combiner.
4. **La méthode probabiliste :** celle-ci 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 'a un ou plusieurs générateurs de nombres aléatoires.
5. **La méthode de la programmation dynamique :** la solution optimale est trouvée en combinant des solutions optimales d'une série de sous-problèmes qui se chevauchent.

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

1. **Un pseudo-code :** est une façon de décrire un algorithme sans référence à un langage de programmation particulier.
2. **Un Ordinogramme :** est une représentation graphique normalisée des opérations et des décisions effectuées par un ordinateur.
3. **Autre présentation :** on peut aussi présenter un algorithme à l'aide d'une implémentation dans un langage de programmation spécifique tel que : Python, java, C, VB, etc.

1.7 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.

L'équipe de développeurs choisit ce genre d'approche pour faciliter la communication entre eux, et la compréhension de l'algorithme.

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

On identifie le meilleur algorithme en faisant une analyse, expérimentale ou théorique, il est important de noter que le meilleur algorithme sera celui qui a un temps d'exécution très bas, qui ne consomme pas trop la mémoire par rapport aux autres.

1.9 En quoi consiste l'analyse d'un algorithme ?

L'analyse d'un algorithme consiste à mesurer son temps d'exécution.

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

1. Méthode expérimentale
2. Méthode théorique

1.11 Quels sont les inconvénients de la méthode expérimentale ?

1. les expériences ne peuvent être faites que sur un nombre limité d'entrées (d'autres entrées pouvant se révéler importantes sont laissées de coté).
2. 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) ;
3. 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.

1.12 En quoi consiste la méthode des opérations primitives ?

Elle consiste à analyser un algorithme particulier en effectuant une analyse de son pseudo-code sans procéder à des expériences sur son temps d'exécution, 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.

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

La complexité d'un algorithme c'est le temps d'exécution propre à un algorithme qui le rend utile ou non compte tenu du problème à résoudre et des moyens disponibles pour ça.

1.14 En quoi consiste la notation asymptotique ?

Elle sert à déterminer le temps d'exécution d'un algorithme en se basant surtout sur le cas le plus défavorable.

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

La fonction constante ($f(n) = c$), la fonction logarithme ($f(n) = logb)n$, la fonction linéaire ($f(n) = n$), la fonction ($f(n) = nlogn$), la fonction quadratique ($f(n) = n^2$), la fonction cubique ($f(n) = n^3$), les autres polynomes ($f(n) = a_0 + a_1n + a_2n^2 + a_3n^3 + ::: + a_dn^d$) et la fonction exponentielle ($f(n) = b^n$).

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

C'est celui qui a un temps d'exécution minimal et qui correspond aux moyens disponibles pour résoudre le problème (complexité).

1.17 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.

Cette méthodologie associe 'a chaque algorithme étudié une fonction $f(n)$, qui caractérise son temps d'exécution en fonction de la taille n de l'entrée. Cette fonction s'obtient par un fitting approprié des données expérimentales.

Exemple : tri par insertion Nous avons fait tourner le programme pour des entrées allant de 1000 à 50000 nombres entiers générés de manière aléatoire.

1.18 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 ?

Prenons l'exemple d'un tri par insertion, soit une $liste = 1, 2, 3, 4, 5, 6, 7, 8, 9$:

Cas favorable : Il consiste a obtenir les informations lors de premières opérations de l'algorithme (1) , c'est un cas où l'entrée est déjà connu ;

Cas moyen : Ici on obtient la réponse après quelques opérations $(4, 5)$;

Cas le plus défavorable : Ici c'est la borne supérieure du temps d'exécution, elle consiste à trouver le résultat vers la fin du temps d'exécution soit (9) ou un élément ne faisant pas partie de notre liste pour notre cas.
Le temps d'exécution associé au cas le plus défavorable a une importance capitale parce que connaitre cette valeur nous permettra donc d'avoir la certitude que l'algorithme ne mettra jamais plus de temps que cette limite.

1.19 Définir en quelques mots le concept de récursivité.

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

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

Récursivité linéaire consiste à un seul appel récursif à la fonction pour résoudre un sous problème ;
Récursivité Binaire consiste à deux appels récursifs pour résoudre deux problèmes distincts ;
Récursivité multiple implique plusieurs appels récursifs à la fonction pour résoudre plusieurs sous problèmes différents.

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

Il doit être un processus dans lequel une fonction appelle elle-même directement ou indirectement, il doit se definir de la manière suivante :

1. Scénario de référence ou de base (c'est-à-dire quand s'arrêter) ;
2. Travailler vers le cas de base ;
3. Appel récursif (c'est-à-dire lui-même)

Exemple

definissons l'opération trouve le chemin de retour comme suit :

1. Si vous êtes à la maison, arrêter de bouger.
2. Faites un pas.
3. Trouve ton chemin à la maison.