Participants :

- BASHONGA LUKELEJI Céleste 2GM
- BUALUTI BUKELE Ephraim 2GEI
- KIENGA LUKEMBISA Guy-Landié 2GC

T.P. Numéro 002 D'ALGORITHMIQUE ET PROGRAMMATION

1)R)

- C'est 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).
- C'est un outil permettant de résoudre un problème de calcul bien spécifié.

2)R)

Un algorithme efficace est celui qui résout le problème donné dans un minimum de temps possible.

3)R)

- Il doit être capable de trouver des solutions qui conviennent ou qui soient meilleures.
- Il doit avoir des applications concrètes.

4)R)

Parmi les méthodes de conception d'algorithme, nous pouvons citer : La méthode de la force brute, la méthode gloutonne, la méthode du diviser pour régner, la méthode probabiliste, l'approche par la programmation dynamique, décomposition en sous-problèmes, recherche dichotomique, Backtracking, heuristique et greedy algorithm.

5)R)

- On peut comprendre que la méthode de la force brute peut s'avérer plus lente car elle doit effectuer l'essai de toutes les solutions possibles mais la méthode gloutonne semble optimale et plus adaptée. Un algorithme glouton est un algorithme qui étape par étape fait le choix d'un optimum local.
- 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.

6)R)

- Pseudo-Code : est une façon de décrire un algorithme sans référence à un langage de programmation particulier.
- Ordinogramme : C'est une représentation graphique normalisée des opérations et des décisions effectuées par un ordinateur. Les mots organigramme, algorigramme, logigramme et ordinogramme sont synonymes.

En outre, un Algorithme peut être présenté avec : l 'écriture formelle, le diagramme de flux, présentation Orale, programmation de Code.

7)R)

Une équipe de développeurs de logiciel peut choisir de représenter les algorithmes de différentes manières pour des raisons telles que :

Participants :

- BASHONGA LUKELEJI Céleste 2GM
- BUALUTI BUKELE Ephraim 2GEI
- KIENGA LUKEMBISA Guy-Landié 2GC

- Clarification : les pseudo-codes peut aider à clarifier les idées et les concepts sous-jacent a un algorithme en utilisant une syntaxe proche du langage naturel.
- Facilitation de la communication : les organigrammes peuvent être utilisés pour visualiser la structure et les flux de contrôle d'un algorithme, facilitant ainsi la communication entre les membres de l'équipe.
- Vérification de la syntaxe et tests : les équipes peuvent choisir d'utiliser différentes représentations selon les besoins du projet et la phase du développement.

8)R)

On reconnaitra le bon algorithme par sa capacité de résoudre un problème en un temps minimal.

9)R)

Ces deux méthodes sont : L'analyse théorique et l'analyse expérimentale.

10)R)

Elle présente les inconvénients ci-après :

- 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 même 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)R)

La méthode des opérations primitives consiste :

- Assigner une valeur à une variable ;
- Effectuer une opération arithmétique ;
- Comparer deux nombres ;
- Indexer un tableau ;
- Suivre la référence d'un objet ;
- Sortir d'une méthode.

12)R)

La complexité d'un algorithme est une mesure de temps et de l'espace nécessaire pour exécuter un algorithme en fonction de la taille de son entrée.

13)R)

La notion asymptotique consiste est une méthode pour évaluer la complexité d'un algorithme en termes de la croissance relative de la quantité de travail qu'il effectue en fonction de la taille de son entrée.

Participants :

- BASHONGA LUKELEJI Céleste 2GM
- BUALUTI BUKELE Ephraim 2GEI
- KIENGA LUKEMBISA Guy-Landié 2GC

T.P. Numéro 002 D'ALGORITHMIQUE ET PROGRAMMATION

14)R)

Les fonctions qui apparaissent le plus lors de l'analyse théorique des algorithmes : les fonctions trigonométriques, les fonctions exponentielles, les fonctions logarithmiques, les polynômes, les fonctions racines carrées, les fonctions fractionnaires, les fonctions logistiques et les fonctions de Bessel.

15)R)

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

16)R)

Le sens exact de "taille d'une entrée" dépend du problème à résoudre. Pour de nombreux problèmes tels que le tri, 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 trier. 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 graphe, on pourra d'écrire la taille de l'entrée par le nombre de sommets et le nombre d'arcs. Ainsi pour chaque problème nous indiquerons la mesure utilisée pour exprimer la taille de l'entrée.

17)R)

Même pour des entrées ayant la même taille, le temps d'exécution d'un algorithme peut dépendre de l'entrée particulière ayant cette taille.

Pour répondre à cette question, prenons le cas du tri par insertion où le tableau en entrée est déjà trié. Dans ce cas pour j=2,3…n, on trouve $A[j] \leq clé\ en\ ligne\ 5\ quand\ i\ prend\ sa\ valeur$ initiale de j-1.

Donc, $t_j = 1$ pour j=1,2,3…n. le temps d'exécution associé est donc :

$$T(n) = C_1 n + C_2(n-1) + C_4(n-1) + C_5(n-1) + C_8(n-1)$$

$$=(C_1 + C_2 + C_4 + C_5 + C_8)n - (C_2 + C_4 + C_5 + C_8)$$

Si le tableau est trié dans l'ordre décroissant, alors c'est le cas le plus défavorable. On doit comparer chaque élément du sous tableau trié A [1… j-1] et donc $t_j = j\ pour\ j = 2,3, \dots n.$

Nous avions pu retenir que :

- Le temps d'exécution associé au cas le plus défavorable est une borne supérieure du temps d'exécution associé à une entrée quelconque. Connaître cette valeur nous permettra donc d'avoir la certitude que l'algorithme ne mettra jamais plus de temps que cette limite .

Participants :

- BASHONGA LUKELEJI Céleste 2GM
- BUALUTI BUKELE Ephraim 2GEI
- KIENGA LUKEMBISA Guy-Landié 2GC

- Pour certains algorithmes, le cas le plus défavorable survient assez souvent. Par exemple la recherche dans une base de données d'une information qui ne s'y trouve pas.
- Il n'est pas rare que le cas moyen soit aussi mauvais que le cas le plus défavorable.

18)R)

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

19)

- Récursivité linéaire : Elle est conçue pour que chaque invocation du corps fasse au plus un nouvel appel récursif. Une conséquence de la définition de la récursivité linéaire est que toute trace de récursivité apparaitra comme une seule séquence d'appels.
- Récursivité binaire : On parle de récursivité binaire lorsqu' une fonction effectue deux appels récursifs.
- Récursivité multiple : Par généralisation de la récursivité binaire, c'est un processus dans lequel une fonction peut effectuer plus de deux appels récursifs.

20)R)

- Un problème récursif doit se définir tel que la solution d'une partie du problème soit utilisée pour résoudre une partie plus petite du problème. Cette répétition de décomposition en sous problèmes permet d'atteindre une taille minimale du problème qui peut être résolue directement et bien.

Un exemple est celui de la recherche d'un élément dans une liste triée.