**Institut Francophone International**

**COMPLEXITÉ ET ALGORITHMES**

**Rapport de travail domicile**

**Sujet : “La solution du problème de mobile équilibré utilisant l’arbre binaire”**

|  |  |
| --- | --- |
| **Enseignant** | : Prof. DO Phan Thuan |
| **Étudiante** | : DAO Thuy Hong |
| **Promotion** | : 20 |

[**SOMMAIRE** 2](#_Toc460787904)

[**1.** **Introduction :** 4](#_Toc460787905)

[**2.** **Description du problème :** 4](#_Toc460787907)

[**3.** **Introduction de l’arbre binaire :** 6](#_Toc460787909)

[**4.** **Algorithme du problème et mis en place :** 8](#_Toc460787910)

[**4.1** **Traitement de l’entrée :** 8](#_Toc460787912)

[**4.2** **Parcours l’arbre binaire :** 10](#_Toc460787913)

[**4.3 Complexité :** 10](#_Toc460787918)

[**5.** **Résultat :** 11](#_Toc460787919)

[**5.1** **Évalution sur “UVa Online Jugde” :** 11](#_Toc460787920)

[**5.2** **Évalution dans le logiciel Themis :** 12](#_Toc460787921)

**TABLEAU DE FIGURE**

[Figure 1 : Un mobile simple 4](#_Toc460788653)

[Figure 2 : Un mobile complexe 5](#_Toc460788654)

[Figure 3 : L’example de l’entrée 6](#_Toc460788655)

[Figure 4 : Un arbre binaire. 7](#_Toc460788656)

[Figure 5 : Un arbre binaire binaire stocké utilisant le pointeur. 7](#_Toc460788657)

[Figure 6 : La structure du noeud 8](#_Toc460788658)

[Figure 7: L’initiation de noeud de racine 8](#_Toc460788659)

[Figure 8 : La procédure récursive pour créer un arbre binaire 9](#_Toc460788660)

[Figure 9 : Le parcours de l’arbre binaire. 10](#_Toc460788661)

Figure 10 : Le résultat sur https://uva.onlinejudge.org 11

[Figure 11 : Indiquer le chemin du répertoire “Tasks”. 12](#_Toc460788663)

[Figure 12 : I indiquer le chemin du répertoire “Tasks” “Contestants” 13](#_Toc460788664)

[Figure 13: Le résultat. 14](#_Toc460788665)

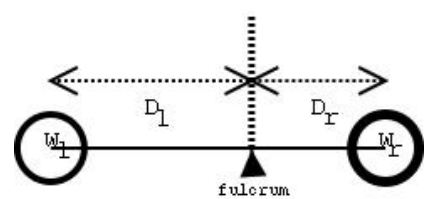
1. **Introduction :**

Dans le cadre du cours complexité et algorithmes, j’ai chois le catégorie de graphe pour mon travail domicile. À partir de mon chois, le professeur m’a donné le problem “Not So Mobile” qui provient du le site <https://uva.onlinejudge.org>.

Dans ce rapport, je vais présenter des travaux que j’ai fait et le résultat que j’ai obtenu. Le rapport comprend cinp partie principales. Dans la partie 2, je vais décrire mon problem et la forme d’entrées et sorties. La connaissance de base de la méthode utilisé pour résoudre le problème va être abordée dans le partie 3. La partie 4 va présenter l'algorithme en détail avec la complexité. La partie 5 va donner quelqué parties de code importantes dans le programme et l’analyser. Le résultat qui obtient après de lancer le programme va être cité dans la partie 6.

1. **Description du problème :**

Le problème aborde le mobile d'équilibre. Un mobile est une structure qui comprend des cordes et des fils afin de pendre des choses. La figure suivante illustre un mobile simple. Il comprend un fil, pendu par une chaîne, avec un objet de chaque côté.

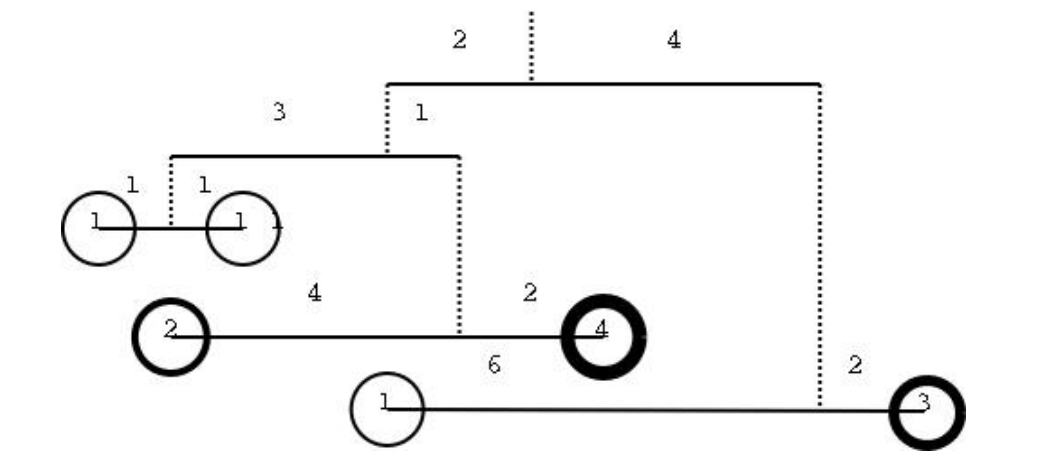


*Figure 1 : Un mobile simple.*

Figure 1

Un mobile simple est équilibré lorsque le produit du poids des objets par leur distance par rapport au point d'appui doit être égal. C’est-à-dire : Wl× Dl = Wr× Dr .

Dans un mobile plus complexe, un objet peut être remplacé par un sous-mobile, comme indiqué sur la figure suivante. Dans ce cas, il n’est pas simple pour vérifier si le mobile est équilibré ou non. Alors, on va écrire un programme qui lit la description d'un mobile d'entrée et puis vérifie l’équilibre de ce mobile.

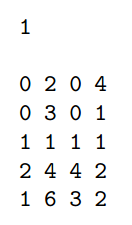


*Figure 2 : Un mobile complexe.*

Figure 2

Dans ce problème, l’entrée et la sortie sont en forme suivante. L’entrée est commence par un nombre entier positif sur une ligne. Ce nombre indique le nombre de cas de test. L'entrée est composée de plusieurs lignes, chaque ligne contient 4 nombres entiers positifs en forme : Wl Dl Wr Dr .

Si Wl ou Wr est égal à zéro alors il y a une suspension de sous-mobile et les lignes suivantes indiquent le sous-mobile. Dan ce cas, le poids du sous-mobile est la somme des poids de tous ses objets. Si les deux Wl et Wr sont zero, deux sous-mobiles sont cités par les lignes suivantes : d'abord la gauche et puis le droit.



*Figure 3 : L’example de l’entrée.*

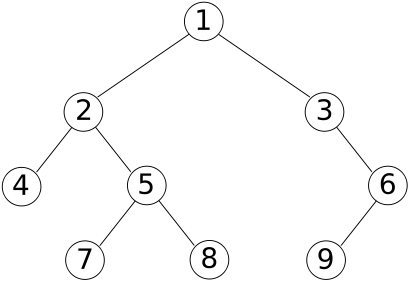
Figure 3

Pour la sortie, pour chaque cas de test, le programme va écrire “YES” si le mobile est équilibré et écrire “NO” dans autre cas. Les sorties de deux cas consécutifs sont séparés par une ligne blanche.

1. **Introduction de l’arbre binaire :**

À partir de la description des entrées, si on considère chaque côté du système de mobile comme un nœud, on pourra utiliser l’abre binaire afin de stocker le poid et le distance de chaque côté. Et puis, on va utiliser le parcours postfixe pour calculer le résultat. Dans cette partie, je vais presenter brièvement la connaissance de base de l’arbre binaire.

Un arbre binaire est un type special de la graphe. Un arbre binaire est un graphe connexe acyclique, tel que le degré de chaque nœud (ou vertex) soit au plus 3. La racine d'un arbre binaire est le nœud d'un graphe de degré maximum 2. Avec une racine ainsi choisie, chaque nœud aura un unique parent défini et deux fils. La figure au-dessous est illustre un arbre binaire.

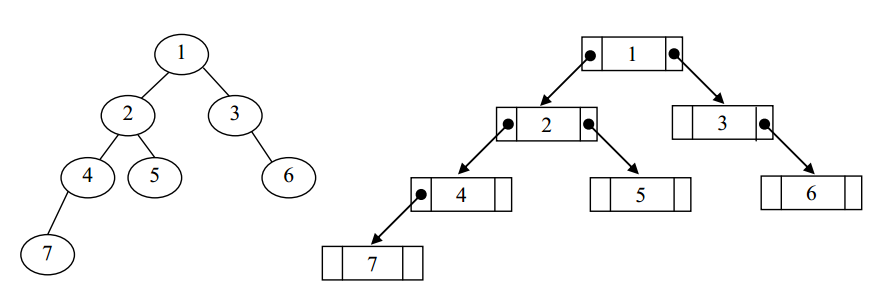


*Figure 4 : Un arbre binaire.*

Figure 4

Pour stocker des arbres binaires, on peut utiliser le pointeur ou le tableau. Ici, je s’interesse seulement le pointeur, donc je vais présenter en détail le façon pour stocker un arbre binaire utilisant le pointeur. Vous pouvez consulter le ligne que je mets dans la partie de référence pour le tableau.

Dans l’arbare binaire stocké utilisant le pointeur, chaque nœud contient quelques données et pointeurs vers son fils droit et son fils gauche. Parfois, il contient également un pointeur vers son unique parent. Si un nœud possède moins de deux fils, l'un des deux pointeurs peut être affecté de la valeur spéciale nulle . La figure suivant présente un arbre binaire binaire stocké utilisant le pointeur.



*Figure 5 : Un arbre binaire binaire stocké utilisant le pointeur.*

Figure 5

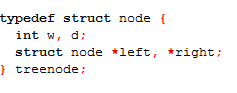
Il existe plusieurs ordres dans lesquels les nœuds peuvent être visités comme : parcours préfixe, infixe et postfixe; parcours en profondeur; parcours en largeur. Je appliquerai le parcours postfixe. Ce type de parcours sera décrit dans la partie suivante.

1. **Algorithme du problème et mis en place :**

Dans cette partie, je vais décrire l’algorithme que je applique et donner des parties de code de mon programme afin de inllustrer l’algorithme. Je utilise le langage C++ pour implémenter l’algorithme.

* 1. **Traitement de l’entrée :**

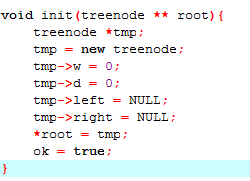
Comme j’ai déja abordé au-dessus, je vais utiliser l’arbre binaire afin de stocker des données dans chaque cas de test. Dans cet arbre, chaque noeud comprend des informations suivantes : poids d’objet W, distance par rapport au point d'appui D, pointeurs qui citent le noeud gauche et le noeud de droit.



*Figure 6 : La structure du noeud.*

Figure 6

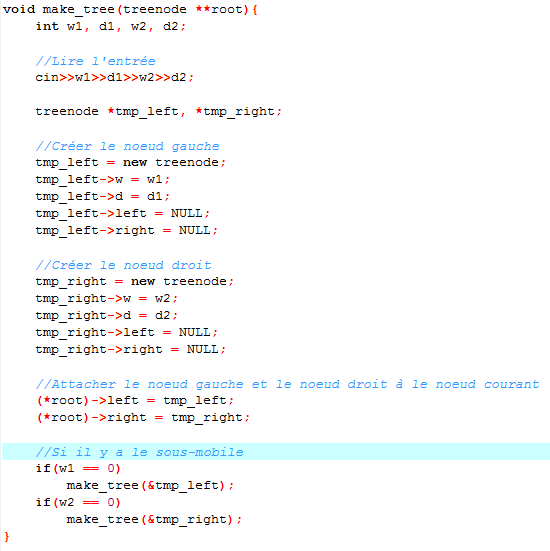
Le processus de creation d’un arbre binaire aura lieu comme ça : d’abord, un noeud de racine est initié.



*Figure 7 : L’initiation de noeud de racine.*

Figure 7

Avec les information dans chaque ligne dans l’entrée, on va créer deux noeuds : noeud gauche et noeud droit et attacher ces noeuds à le noeud de parent courant. Et puis, on va considérer s’il y a le sous-mobiles. Si le poids (W) de noeud est égal zéro, la ligne suivante va être lu pour les information de ce sous-mobile. Au niveau de codage, le processus est implémenté par une procédure récursive. Dans le langage C++, cette procédure est mis en place comme la figure suivante .



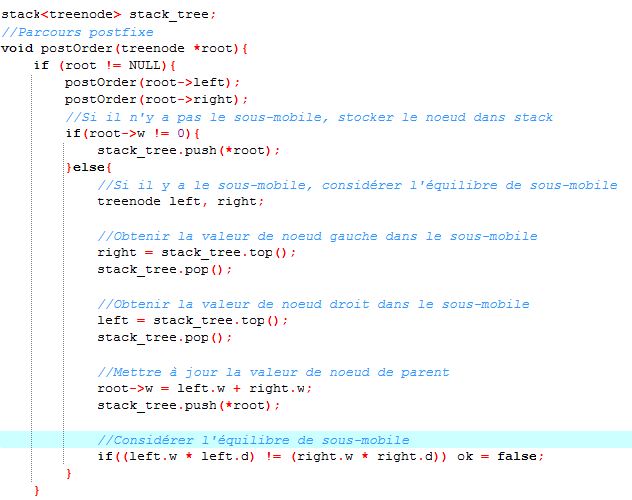
*Figure 8 : La procédure récursive pour créer un arbre binaire.*

Figure 8

* 1. **Parcours l’arbre binaire :**

Pour déterminer si le mobile est équilibré ou non, je vais considérer l’équilibre du sous- mobile en premier. Des sous-mobiles vont être considéré de la gauche du droit, de bas en haut. Pour faire ça, je vais parcourir l’arbre binaire dans ordre postfixe. On doit calculer le somme des poids des objets dans chaque sous-mobile et le stocker.

Au niveau de codage, je utilise stack afin de réaliser ce processus. D’abord, on vérifie la valeur poids du noeud courant. Si cette valeur n’est pas égal zéro, on la mettre dans stack. Si non, on obtiendra deux valeurs dans stack. C’est les valeurs de noeud gauche et noeud droit du sous-mobile. On va tester l’équilibre de ce sous-mobile et calculer le somme des poids des objets dans ce sous-mobile et mettre ce somme dans stack. Le code de cette partie est comme la figure au-dessous.



*Figure 9 : Le parcours de l’arbre binaire.*

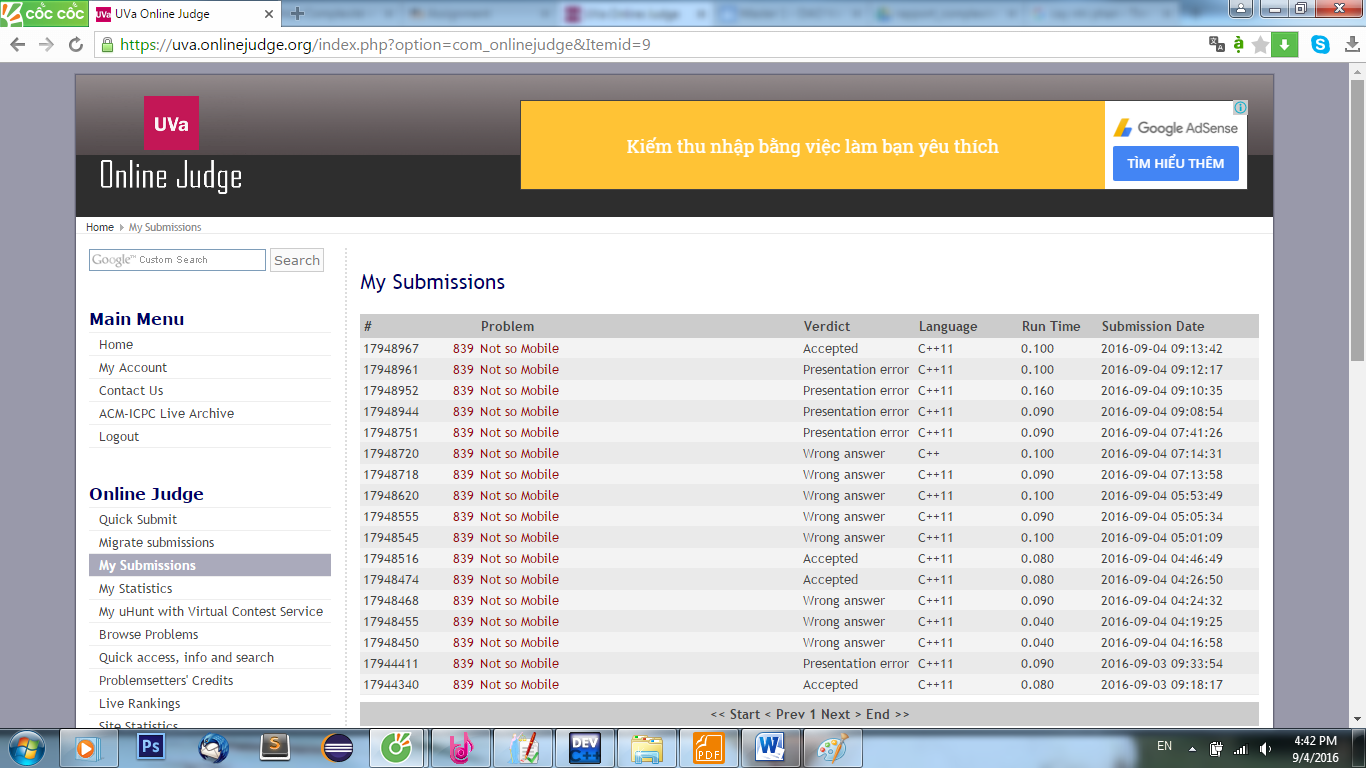
Figure 9

## Complexité :

Dans tous les cas, l’opération de parcourir l'arbre (postOrder) de la racine jusqu'à une feuille : le temps d'exécution est donc proportionnel à la profondeur de l'arbre qui vaut n dans le pire des cas, d'où une complexité maximale en O(n) (n est la quantité de noeud). À l’égard de l’opération “make\_tree”, la complexité est aussi comme l’opération “postOrder”. Alors, la complexité de l’alrogithme est O(n).

1. **Résultat :** 
   1. **Évalution sur “UVa Online Jugde” :**

“UVa Online Jugde” (<https://uva.onlinejudge.org>) est un site qui fournit des centaines de problèmes. Nous pouvons soumettre nos sources dans une variété de langues, en essayant de résoudre l'un des problèmes disponibles sur ce site. Mon problème est un des problèmes dans ce site, donc j’ai soumis mon programme sur <https://uva.onlinejudge.org> afin de l’évaluer et il est accepté. Le résultat comme suivant :



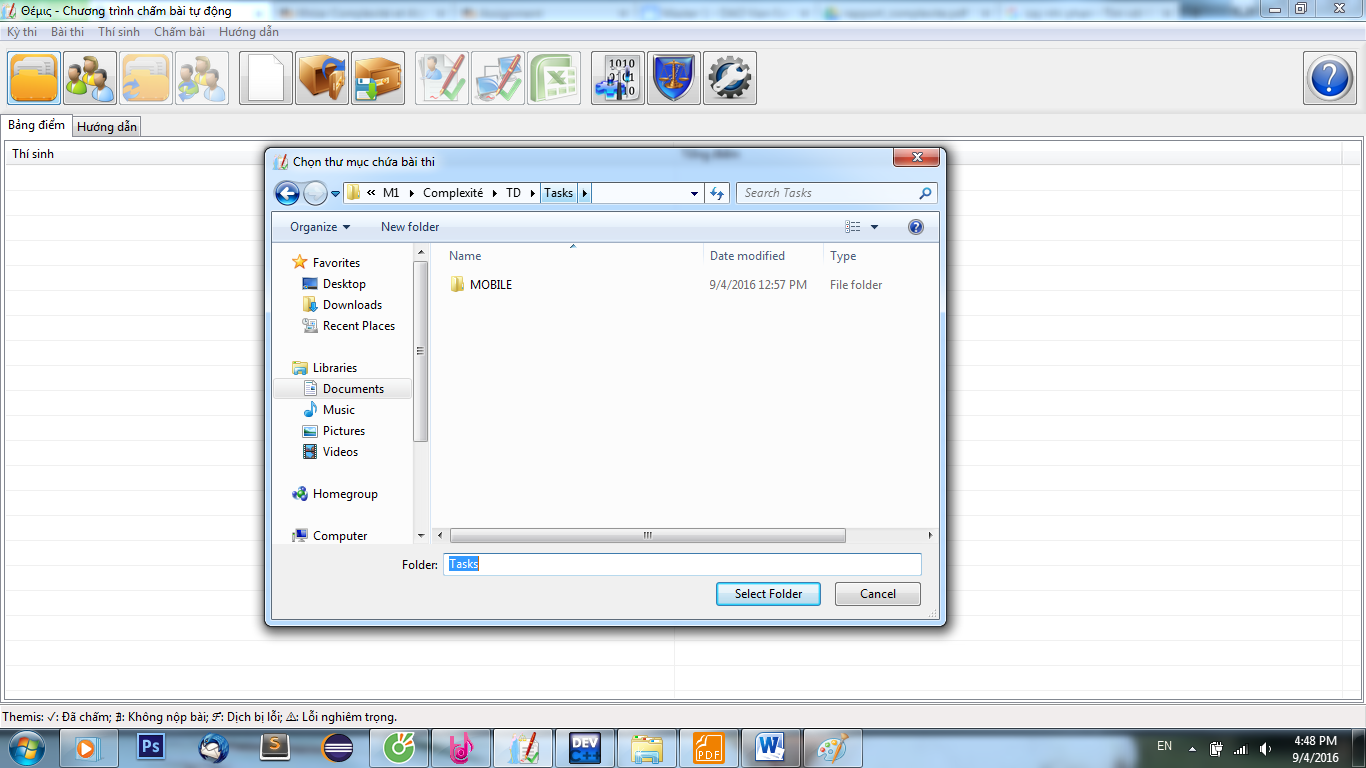
*Figure 10 : Le résultat sur* [*https://uva.onlinejudge.org*](https://uva.onlinejudge.org)

Figure 10

* 1. **Évalution dans le logiciel Themis :**

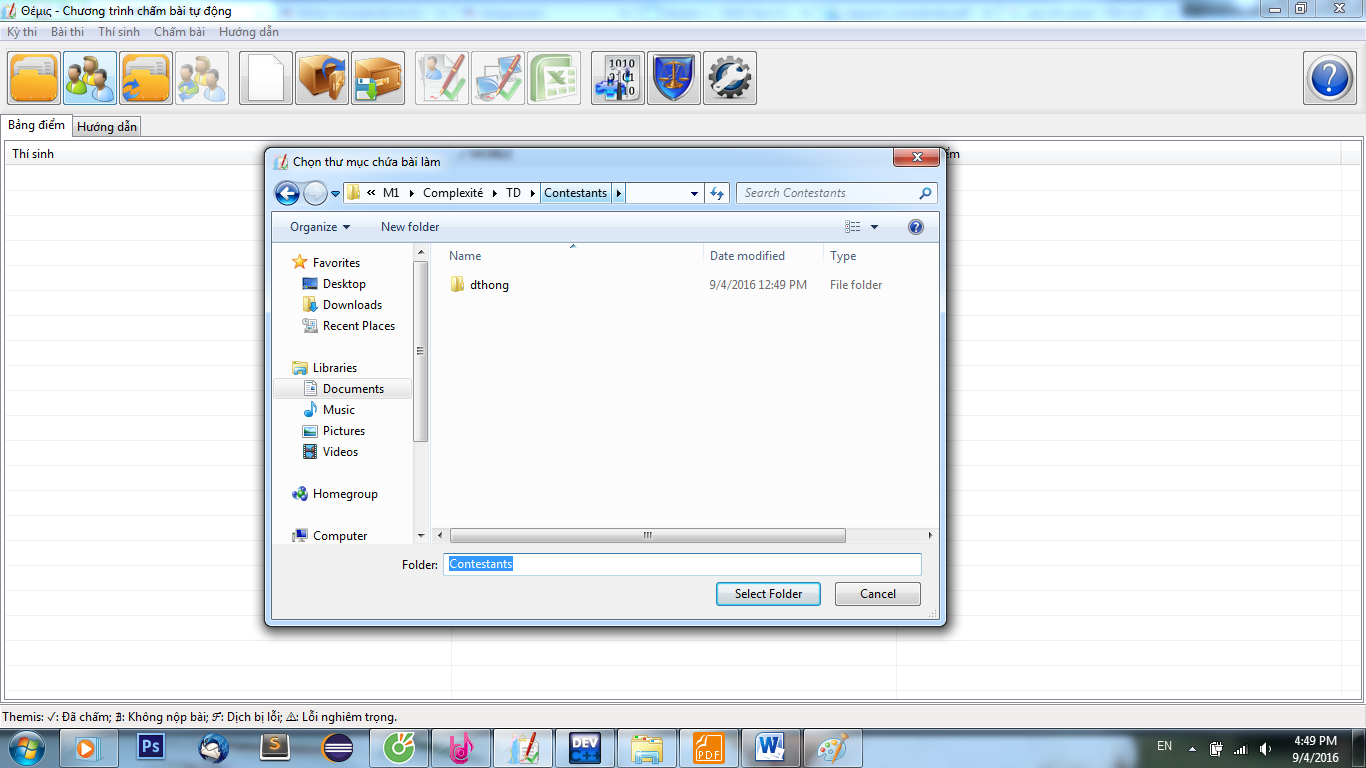
Après d’être accepté sur <https://uva.onlinejudge.org>, j’ai créé 20 cas de test pour évaluer mon programme dans Themis. Themis est un logiciel de correction automatique basé sur des cas de test. Lé auteurs de ce logiciel sont le professeur Le Minh Hoang et le professeur Do Duc Dong.

Pour lancer ce logiel, on doit créer les répertoires “Tasks” et “Contestants”. Le répertoire “Tasks” contient des cas de test et le répertoires “Contestants” contient des sources de code. On appuye le bouton F2 pour indiquer le chemin du répertoire “Tasks” et le bouton F3 pour indiquer le chemin du répertoire “Tasks” “Contestants”.



*Figure 11 : Indiquer le chemin du répertoire “Tasks”.*

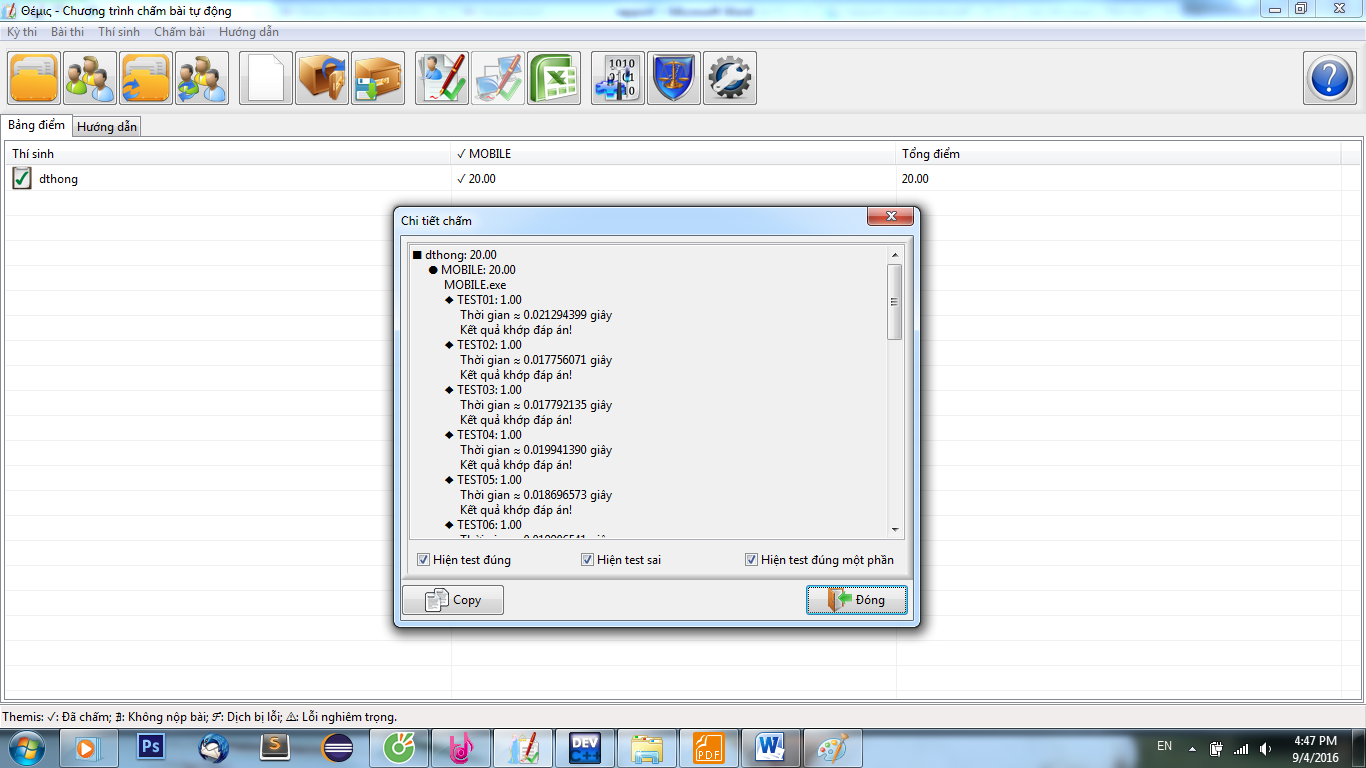
Figure 11



*Figure 12 : I indiquer le chemin du répertoire “Tasks” “Contestants”.*

Figure 12

Et puis, on doit apuyer le bouton F9 pour lancer la function de correction. Je configure la note que on va obtenir si le programme passe un cas de test est 1. Après de lancer sur Themis, je gagne 20, donc mon programme a passé tous les cas de test. Le résultat est illustré dans la figure 13.



*Figure 13 : Le résultat.*

Figure 13

**Référence :**

[1]<https://vi.wikipedia.org/wiki/C%C3%A2y_t%C3%ACm_ki%E1%BA%BFm_nh%E1%BB%8B_ph%C3%A2n>

[2] <https://fr.wikipedia.org/wiki/Arbre_binaire_de_recherche>

[3]<https://uva.onlinejudge.org/index.php?option=com_frontpage&Itemid=1>

[4] <https://dsapblog.wordpress.com/2013/12/24/themis/>

[5]Nguyễn Duy Phương - PTIT, Cấu trúc dữ liệu và giải thuật.

[6] <https://fr.wikipedia.org/wiki/Arbre_binaire>