OBJECTIFS DE COURS

L'objectif de ce cours est de trouver un bon algorithme pour un problème donné. La résolution de ce problème revient à comparer les performances de deux algorithmes effectuant les mêmes tâches, deux paramètres essentiels parmi les ressources coûteuses sont à prendre en compte : le temps d'exécution (Complexité temporelle), et l'espace mémoire requis (Complexité spatiale) qui permet de représenter la quantité de mémoire nécessaire pour l'exécution d'un programme.

La complexité temporelle, pour rendre l'étude du coût temporel d'un algorithme indépendante de tout aspect physique, il faut prendre pour unité de mesure une opération élémentaire pertinente pour le problème étudié. Calculer *le coût (la complexité)* d'un algorithme, est de déterminer *le nombre d'opérations élémentaires effectuées par cet algorithme***.** Cette comparaison doit être indépendante de l'ordinateur utilisé, et sur un même ordinateur du langage de programmation utilisé pour rendre la comparaison indépendante de tout aspect technique.

Parmi les problèmes qui permet d'engendrer des valeurs de complexité très grand, c'est les problèmes qui nécessitent d'explorer des espace de recherche très grands, on utilisons la récursivité. Les problèmes de tri et de recherche permet de présenter des meilleurs exemples de ce type de complexité. La récursivité est un des concepts de programmation les plus importants. Le principe de l'approche récursive est de ramener le problème à résoudre à un sous-problème correspondant à une instance «réduite» du problème lui-même.

Autrement dit, La recherche d'algorithme ayant une complexité plus petite que les meilleurs algorithmes connus est un thème de recherche important dans toutes les branches de l'informatique. La réalisation de cet objectif nécessite des connaissances d'approfondir qui concerne le domaine de l'analyse d'algorithmique et les techniques d'optimisation de la complexité présentés par l'organisation suivante :

Le premier chapitre est consacré à la théorie de la complexité et mesure de performance. L'objectif du cette partie de cours est d'analyser le coût des algorithmes récursifs et de présenter l'optimisation de cette récursivité de la forme de la récursivité enveloppé vers la récursivité terminale qui est une forme de récursion optimisé**.**

En fin, l'algorithme de la récursivité terminale sera optimisé vers la forme d'un algorithme itératif. Donc, une comparaison d'évaluation de complexité d'une solution itérative (simple) est plus efficace qu'une solution récursive équivalente et encore l'étude des classes de complexité des problèmes. Pour ce fait, le deuxième chapitre est pour définir la notion de récursivité, grandeur des fonctions et la complexité asymptotique. Le troisième chapitre pour objectif de présenter les méthodes de calcul de complexité des algorithmes itératifs et récursifs, le chapitre quatre et cinq pour définir la stratégie diviser pour régner et les algorithmes de tri et de recherche. Les approches récursives se trouvent dans les parcours d'arbres, les recherches en largeur et en profondeur d'abord. Donc le chapitre six est pour l'exploration des graphes et les algorithmes de recherche en profondeur, application du backtracking dans les arbres de jeux.

Nous clôturons la présentation de ce cours par le septième chapitre qui permet de présenter la notion d'optimisation combinatoire, quelques définitions de base et les méthodes heuristiques de résolution d'un problème d'optimisation. Nous focalisons notre étude sur les algorithmes de recherche locale, Glouton, A*, Hill Climbing,