U.S.T.H.B - 2025-2026
Faculté d'Informatique
N. Bensaou & M.N Djouada

Master Informatique Visuelle
Algorithmique et Complexité (M1 - MIV)

Projet TP : Analyse et conception des algorithmes

1 Présentation du Problème

Problème SAT (Satisfaisabilité)

Soit F une formule logique exprimée en **forme normale conjonctive (CNF)**, c'est-à-dire une conjonction de clauses où chaque clause est une disjonction de littéraux.

Formulation formelle :

- $F = C_1 \wedge C_2 \wedge \cdots \wedge C_k$, où chaque clause C_i est de la forme :

$$C_i = (l_{i1} \vee l_{i2} \vee \cdots \vee l_{im_i}),$$

avec l_{ij} un littéral (une variable x ou sa négation $\neg x$).

Objectif : Trouver une affectation des variables de F telle que F soit évaluée à vrai, c'est-à-dire qu'il existe au moins un littéral vrai dans chaque clause C_i.

Sortie :

- "Oui" si une telle affectation existe, avec éventuellement l'affectation trouvée.
- "Non" sinon.

Exemple pour le problème SAT

Considérons la formule suivante en forme normale conjonctive (CNF) :

$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3).$$

Une affectation possible qui satisfait F est :

$$x_1 = \text{vrai},\ x_2 = \text{vrai},\ x_3 = \text{vrai}.$$

Avec cette affectation, chaque clause contient au moins un littéral évalué à vrai, donc F est satisfaisable.

Problème 3-SAT

Le problème 3-SAT est une version particulière du problème SAT où chaque clause contient **exactement trois littéraux**.

Formulation formelle :

- $F = C_1 \wedge C_2 \wedge \cdots \wedge C_k$, où chaque clause C_i est de la forme :

$$C_i = (l_{i1} \vee l_{i2} \vee l_{i3}),$$

avec l_{ij} un littéral (une variable x ou sa négation $\neg x$).

Objectif : Déterminer s'il existe une affectation des variables de F telle que F soit évaluée à vrai, c'est-à-dire qu'il existe au moins un littéral vrai dans chaque clause C_i.

Sortie :

- "Oui" si une telle affectation existe, avec éventuellement l'affectation trouvée.
- "Non" sinon.

Exemple pour le problème 3-SAT

Considérons la formule suivante en 3-SAT (chaque clause contient exactement 3 littéraux) :

$$F = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3).$$

Une affectation possible qui satisfait F est :

$$x_1 = \text{faux}, x_2 = \text{faux}, x_3 = \text{vrai}.$$

Avec cette affectation, chaque clause contient au moins un littéral évalué à vrai, donc F est satisfaisable.

Problème de la Somme des Sous-Ensembles (SUBSETSUM)

Soit un ensemble fini d'entiers $S = \{a_1, a_2, \ldots, a_n\}$ et une valeur cible $T \in \mathbb{Z}$. Le problème de la somme des sous-ensembles consiste à déterminer s'il existe un sous-ensemble de S dont la somme des éléments est exactement égale à T.

Formulation formelle :

- Trouver un sous-ensemble $S' \subseteq S$ tel que :
$$\sum_{a_i \in S'} a_i = T.$$

Objectif : Déterminer si un tel sous-ensemble existe.

Sortie :

- "Oui" s'il existe un sous-ensemble S' satisfaisant la contrainte.
- "Non" sinon.

Exemple pour le problème SUBSETSUM

Considérons l'instance suivante : $S = \{3, 5, 7, 10\}$, $T = 15$. Un sous-ensemble possible dont la somme vaut T est : $S' = \{5, 10\}$. Une solution existe et la réponse au problème est "Oui".

Objectifs de ce travail :

- Implémenter les solutions pour résoudre les trois problèmes.
- Implémenter les réductions de SAT vers 3SAT et SAT vers SUBSETSUM et l'étudier pratiquement à travers des benchmarks.
- Analyser et comparer les complexités en temps et en espace de chaque algorithme de manière empirique.

2 Questions

- **Développement d'Algorithmes**

1. Etant donnée une instance I du problème SAT, écrire un programme qui trouve la solution de cette instance.
2. Etant donnée une instance du problème SAT, écrire un programme qui vérifie si une `solution` potentielle s satisfait toutes la clauses I.
3. Etudier la complexité pratique, temporelle et spatiale, de deux programmes précédents en spécifiant les structures de données utilisées pour représenter l'instance et les solutions.
4. Pareil pour le problème du 3SAT :

   - Implémenter l'algorithme qui trouve la solution d'une instance donnée.
   - Implémenter l'algorithme qui vérifie si une solution potentielle donnée satisfait une instance donnée.
   - Etudier la complexité pratique, temporelle et spatiale, des deux programmes en spécifiant les structures de données utilisées.

5. Pareil pour le problème SUBSETSUM :
   - Implémenter l'algorithme qui trouve la solution d'une instance donnée.
   - Implémenter l'algorithme qui vérifie si une solution potentielle donnée satisfait une instance donnée.
   - Etudier la complexité pratique, temporelle et spatiale, des deux programmes en spécifiant les structures de données utilisées.
   - Implémenter l'algorithme de résolutiob basé sur la programmation dynamique et étudier sa complexité.
6. Implémenter un algorithme de réduction (linéaire) SAT vers 3SAT (ainsi que les solutions) et étudier sa complexité pratique.
7. Implémenter un algorithme de réduction (linéaire) SAT vers SUBSETSUM (ainsi que les solutions) et étudier sa complexité pratique.

- **Analyse Empirique de la Complexité**
  1. **Comparaison de la Performance** :
     - Générer des instances de tailles différentes pour les trois problèmes.
     - Enregistrer et analyser le temps d'exécution et l'utilisation de la mémoire de chaque approche pour chaque taille d'entrée.
     - Créer des graphes pour montrer la croissance empirique des performances (temps/espace) en fonction de la taille de l'entrée.
  2. **Exploration d'applications du monde réel** :
     - Utiliser les bibliothèques :
       * SATLIB : `https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`.
       * KPLIB : `https://github.com/likr/kplib`.
       * SUBSET_SUM : `https://people.sc.fsu.edu/~jburkardt/datasets/subset_sum/subset_sum.html`.

       pour tester vos programmes.

3 Livrables

- **Code** : Remettre code de chaque solution et l'application.
- **Rapport** :Inclure un rapport (max 15 pages) qui couvre :
  - L'analyse de la complexité et les résultats empiriques (graphiques inclus). En spécifiant l'environnement d'expérimentation (Matériel et Logiciel).
  - La discussion sur les optimisations et les conditions qui réduisent la complexité pour l'application.

Ce travail doit être remis au plus tard Le 02/01/25. Ce travail se fait en ***groupes de 6 étudiants.***.