שימוש בפותרן לבעיית הספיקות

כיצד לשלב את הפותרן בפרויקט אקליפס?

בתחילת העבודה מומלץ ליצור פרויקט java בסביבת אקליפס ולבצע את הפעולות הבאות:

1. להוסיף את כל קבצי הקוד המצורפים לעבודה לספריית הקוד של הפרויקט. ספריית הקוד בפרויקט אקליפס מקבלת את השם src כברירת מחדל.
2. שימו לב כי בקבצי הקוד שקיבלתם:
   a. ישנו קובץ שנקרא **org.sat4j.core.jar**. זהו הקובץ המכיל את הפותרן. אינכם צריכים לעבוד איתו ישירות, אבל צריך שיהיה בספריית הקוד שלכם.
   b. ישנו קובץ שנקרא **SATSolver.java**. זהו הקובץ בו נמצאות כל הפונקציות שאתם צריכים עבור העבודה עם הפותרן. פונקציות אלו מתוארות בסעיף הקודם "עיקרי הממשק של ה-SAT Solver".
3. כדי שיהיה אפשר לעבוד עם הפותרן, יש להוסיף אותו ל-Build Path של הפרויקט. למשל כך:
   a. באקליפס, לחצו עם המקש הימני של העכבר על הקובץ org.sat4j.core.jar שהוספתם לפרויקט.
   b. בחרו באפשרות "Build Path" ואז לחצו על האפשרות "Add to Build Path".
   c. כדי לוודא שהפותרן אכן משולב בפרויקט, תוכלו לכתוב פונקציית main עם קוד מאחת הדוגמאות שבסעיף הקודם ולוודא שהדוגמה אכן עובדת.

עיקרי הממשק של ה-SAT Solver:

- <u>אתחול:</u> יש לבצע קריאה לפונקציה SATSolver.init(int nVars) כאשר הערך במשתנה nVars מציין שמשתני ה-CNF שיופיעו בנוסחת ה-CNF יילקחו אך ורק מתוך הקבוצה $\{x_1, x_2, \dots, x_{nVars}\}$. למשל, לאחר אתחול הפותרן בקריאה: SATSolver.init(34), יהיה אפשר להתייחס רק למשתני CNF מתוך הקבוצה $\{x_1, \dots, x_{34}\}$.
- <u>הוספת פסוקית:</u> כדי להוסיף פסוקית בודדת לפותרן, יש לקרוא לפונקציה
  SATSolver.addClause(int[] clause)
  כאשר המערך clause מייצג פסוקית. למשל, שורות הקוד הבאות:

```
int[] clause = {5,2,-6,7,12};
SATSolver.addClause(clause);
```

  יוסיפו את הפסוקית $(x_5 \vee x_2 \vee \neg x_6 \vee x_7 \vee x_{12})$ לפותרן.

- <u>הוספת פסוקיות:</u> כדי להוסיף כמה פסוקיות לפותרן, יש לקרוא לפונקציה
  SATSolver.addClauses(int[][] clauses)
  כאשר המערך הדו-ממדי clauses מייצג את הפסוקיות. למשל, שורות הקוד הבאות:

```
int[][] clauses = { {5,-2,6}, {4,-17,99} };
SATSolver.addClauses(clauses);
```

  יוסיפו את הפסוקיות $(x_5 \vee \neg x_2 \vee x_6)$ ו- $(x_4 \vee \neg x_{17} \vee x_{99})$ לפותרן.

- <u>מציאת השמה מספקת:</u> כדי לפתור את נוסחת ה-CNF שהצטברה עד כה ב-SATSolver, יש לקרוא לפונקציה

SATSolver.getSolution()

פונקציה זו מחזירה ערך לפי אחת משלוש האפשרויות הבאות:

1. **מערך בוליאני שאינו ריק** - במידה שישנה השמה מספקת. אורך המערך יהיה כמספר המשתנים פלוס אחד. מערך זה מייצג השמה מספקת כפי שהוסבר במבוא לחלק 2 של העבודה, בסעיפי התזכורות.
2. **מערך בוליאני ריק** – במידה שהנוסחה אינה ספיקה (לא קיימת לה השמה מספקת).
3. **ערך null** – במידה שהפותרן לא מצא פתרון, עקב מגבלת זמן (timeout של 3 דקות).

דוגמאות:

1. התכנית הבאה מגדירה נוסחת CNF בעלת שלוש פסוקיות: $\left((x_1) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3)\right)$, מבקשת השמה מספקת מהפותרן, ומדפיסה פלט בהתאם לתוצאה: "SAT" אם הנוסחה מסתפקת, "TIMEOUT" אם הפותרן לא מצא פתרון עקב מגבלת זמן ו- "UNSAT" אם הנוסחא לא מסתפקת .

```
int nVars = 3;
SATSolver.init(nVars);

int[] clause = {1};
SATSolver.addClause(clause);

int[][] clauses = {{-1,-2}, {2,3}};
SATSolver.addClauses(clauses);

boolean[] assignment = SATSolver.getSolution();
if (assignment == null)
      System.out.println("TIMEOUT");

else if (assignment.length == nVars+1)
      System.out.println("SAT");
else
      System.out.println("UNSAT");
```

הפלט של תכנית זו הוא "SAT".

2. התכנית הבאה מגדירה נוסחת CNF בעלת ארבע פסוקיות:

$$\left((x_1) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)\right)$$

מבקשת השמה מספקת מהפותרן, ומדפיסה פלט בהתאם לתוצאה: "SAT" אם הנוסחה מסתפקת, "TIMEOUT" אם הפותרן לא מצא פתרון עקב מגבלת זמן ו- "UNSAT" אם הנוסחא לא מסתפקת .

```
int nVars = 3;
SATSolver.init(nVars);

int[] clause = {1};
SATSolver.addClause(clause);

int[][] clauses = {{-1,-2}, {2,3}, {-1,-3}};
```

```
SATSolver.addClauses(clauses);

boolean[] assignment = SATSolver.getSolution();
if (assignment == null)
      System.out.println("TIMEOUT");

else if (assignment.length == nVars+1)
      System.out.println("SAT");
else
      System.out.println("UNSAT");
```

הפלט הצפוי הוא "UNSAT".