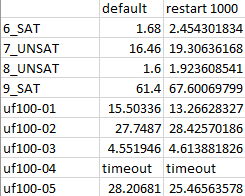
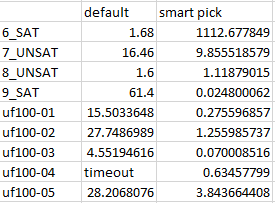
[What I implemented]

1. Basic function for DPLL algorithm with clause learning that I learned in course.
   1. Unit propagation
   2. Propositional variable decision (set propositional variable as false)
   3. Clause learning
2. Optimizations
   1. Restart after particular number of conflicts
   2. Smart decision (choosing literal that occurs max in clauses and set it as true)

[Design decision, rationale behind my implementation]

1. Data structures
   1. [assignment]: it is python dictionary that maps propositional variable to assignment element.
   2. [assignment element]: it contains truth value for propositional variable and imply\_clause for indicating imply clause if it is implied by unit propagation.
   3. [f\_a]: For each unit propagation, I have to find unit clause in F|A. But calculating F|A every time was very inefficient. Therefore, I maintained data structure f\_a which is python dictionary that maps original clause to clause under F|A.
      1. It is updated when assignment (by adding or popping assignment) or CNF formula (by adding learned clause) change.
      2. When assignment changes, for efficiency, change only clauses which is containing target literal. (use literal\_map for efficiency)
   4. [unit clauses]: set of unit clauses. It is used for unit propagation
      1. It is updated when f\_a change.
   5. [conflict clause]: current conflict clause. Some or None. It is used for finding conflict clause if it exists.
      1. It is updated when f\_a change.
   6. [literal\_map]: It is python dictionary that maps literal to clauses that contain the literal.
      1. It is updated when CNF formula change
2. Optimizations
   1. Restart after particular number of conflicts
      1. performance
   2. Smart decision (choosing literal that occurs max in clauses and set it as true)
      1. performance
   3. Restart and Smart decision
      1. performance