**ЛАБОРАТОРНАЯ РАБОТА № 2**

**«ПОИСК РЕШЕНИЙ CSP ЗАДАЧ»**

**Цель работы**

Изучение особенностей задач удовлетворения ограничений (CSP – Constraint Satisfaction Problem) и исследование основных методов поиска их решений средствами языка Пролог.

**Постановка задачи**

В одном классе учатся три брата: Алексей, Леонид и Александр. Учитель заметил, что:

— если кто-то из них получает подряд две четверки или две тройки, то дальше он учится кое-как и получает тройку;

— если он получает подряд две пятерки, то совсем перестает заниматься и получает двойку;

— если он получает две разные оценки, то следующей будет большая из

них.

В начале полугодия Алексей получил оценки 4 и 5, Леонид — 3 и 2, Александр — 2 и 4. Какие итоговые оценки получит каждый из рассматриваемых школьников за данное полугодие, если учитель выставил каждому по 30 оценок, а итоговая оценка — ближайшее целое число к среднему арифметическому полученных оценок?

**Ход работы**

Для начала был написан код программы на prolog (код приведен в приложении А), который решает поставленную задачу применяя метод CSP «генерируй и проверяй», далее был создан предикат main, который создает полные решения для всех братьев.

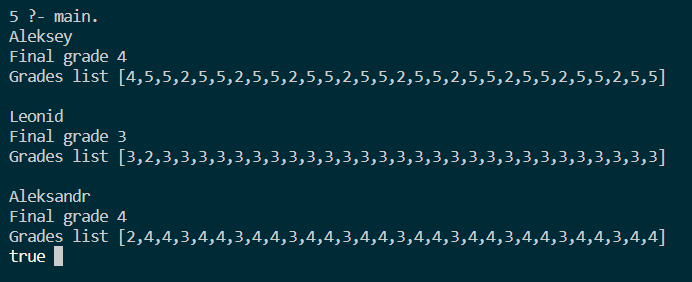


Рисунок 1 – Вызов предиката main

Далее было изучено время срабатывания работы поиска оценки для Алексея с запуском поиска 1000 раз.



Рисунок 2 – Статистика выполнения программы

Как видим результат получился достаточно ресурсозатратным как по времени выполнения, так и по количеству пере присваиваний, возможно его можно улучшить, если использовать предварительную проверку ограничений.



Рисунок 3 – Статистика выполнения программы с использованием предварительной проверки ограничений

В результате реализации предварительной проверки ограничений результат улучшился приблизительно в 10 раз по обоим показателем, что говорит о том, что применение предварительной проверки ограничений было правильным выбором.

**Выводы**

В ходе проделанной работы были изучены особенности CSP задач, рассмотрены методы реализации решений CSP задач на языке prolog, а также была решена соответствующая задача методом «генерируй и проверяй», было изучено время ее выполнения, были предложены возможные варианты ускорения поиска решения поставленной задачи и реализован метод поиска с возвратами и предварительной проверки ограничений, что дало значительное улучшения статистики выполнения программы.

ПРИЛОЖЕНИЕ А

% Оценки братьев

grade(aleksey, [4, 5]).

grade(leonid, [3, 2]).

grade(aleksandr, [2, 4]).

% Правило для получения последних двух элементов списка

last\_two\_elements(List, Result) :-

append(\_, [X, Y], List),

Result = [X, Y].

del(X,[X|T],T). % удаление из списка

del(X,[Н|T],[Н|T1] ):-del(X, T, T1).

insert(X,L1,L):-del(X,L,L1). % добавление в список

% Пример использования:

% Результат: L = [2, 3, 1]

swap([],[]).

swap([X|L],P):-swap(L,L1), insert(X, L1, P).

max(X, Y, Max) :- % поиск максимального из двух чисел

(X >= Y -> Max = X ; Max = Y).

find\_next(G):- % поиск решения

check(X,G),generate(X).

generate(X):- % генерация решения

swap([2,3,4,5], X).

check(X,G):- % проверка решения

[F|\_] = X,

last\_two\_elements(G,Result),

[A,B] = Result,

((A=5,B=5, F=2);

(A=4,B=4,F=3);

(A=3,B=3,F=3);

(A\=B, max(A,B,Max), F=Max)),

append(G, [F], Gn),

length(Gn,Len),

((Len =< 30, find\_next(Gn));

(Len = 30,

average(Gn, Avg),

round\_integer(Avg, FinalGrade),

write("Final grade"),write(' '), writeln(FinalGrade),

write("Grades list"), write(' '), writeln(Gn))).

% Предикат для вычисления среднего арифметического

average(List, Avg) :-

sum\_list(List, Sum),

length(List, Len),

Avg is Sum / Len.

% Предикат для округления до ближайшего целого числа

round\_integer(X, Y) :-

Y is round(X).

% Предикат для вычисления итоговой оценки

final\_grade(Name) :-

grade(Name, Grades),

find\_next(Grades).

main:-

writeln("Aleksey"),final\_grade(aleksey),nl,

writeln("Leonid"),final\_grade(leonid),nl,

writeln("Aleksandr"),final\_grade(aleksandr).

% цикл повторения выполнения Цели заданное число раз (N)

repeat(Goal,1):-Goal.

repeat(Goal,N):-

not(not(Goal)), %стирание предыдущих подстановок

M is N-1,repeat(Goal,M).