BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Pada Tahap Implementasi ini dilakukan implementasi apa yang telah dirancang di tahapan sebelumnya untuk melihat apakah perancangan yang telah dilakukan sebelumnya mampu berfungsi dan berjalan sesuai harapan atau tidak.

Implementasi Algoritma DPLL

SAT Solver mengimplementasikan algoritma DPLL menggunakan beberapa fungsi yang bukan hanya saling bergantung pada satu sama lain namun juga di gunakan untuk menyelesaikan *SAT problem* yang diberikan. Adapun fungsi-fungsi tersebut adalah :

a. Metode *Unit Propagation*

Metode ini adalah metode yang berfungsi memberikan nilai *True* terhadap klausa yang memiliki 1 literal dan juga klausa lain yang memiliki literal yang sama dalam klausa tersebut.

```
unitPropagnation(byref Claw){
tempVar := ""
tempArray := object()
loop % Claw.MaxIndex()
{
        tempArray := StrSplit(Claw[A_Index],A_Space)
        i:=0
        loop
        {
                i++
                if (tempArray.MaxIndex() = "" or i > tempArray.MaxIndex())
                        break
                if tempArray[i] = "0"
                {
                        tempArray.removeat(i)
                        i:=0
                }
        }
        if tempArray.MaxIndex() = 1
                if tempArray[1] is integer
                {
                        tempVar := tempArray[1]
                        break
                }
}
if tempVar is integer
{
        if tempVar is integer
                NtempVar := tempVar * -1
        else
                if InStr(tempVar,"-")
                        NtempVar := StrReplace(tempVar,"-")
                else
                        NtempVar := "-" + tempVar

        i:=0
        loop
        {
                i++
                if (Claw.MaxIndex() = "" or i > Claw.MaxIndex())
                        Break

                tempArray := StrSplit(Claw[i],A_Space)
                Claw[i] := StrReplace(Claw[i],NtempVar " ")

                loop % tempArray.MaxIndex()
                        if tempArray[A_Index] = tempVar
                        {
                                Claw.RemoveAt(i)
                                i:=0
                        }
        }

}
return
}
```

Gambar 4.1Coding Unit Propagation

b. Metode *Pure Literals*

Metode ini adalah metode yang berfungsi memberikan nilai *True* terhadap klausa yang memiliki literal tanpa adanya bentuk negasi dari literal tersebut dalam klausa manapun.

```
PureLiterals(byref Claw){
tempVar:=object()
NtempVar:=object()
Claw2 :=Claw.Clone()
ExLiter( Claw2 ,tempVar)
loop % tempVar.MaxIndex()
        NtempVar[A_Index]:= tempVar[A_Index] * -1
loop % tempVar.MaxIndex()
{
        tempArray:=object()
        B_Index:= A_Index
        point:=0
        loop % Claw.MaxIndex()
        {
                tempArray := StrSplit(Claw[A_Index],A_Space)
                if existIn(tempArray,tempVar[B_Index]) = 1
                {
                        point:=point + 1
                        break
                }
        }
        loop % Claw.MaxIndex()
        {
                tempArray := StrSplit(Claw[A_Index],A_Space)
                if existIn(tempArray,NtempVar[B_Index]) = 1
                {
                        point:=point + 2
                        break
                }
        }
        i:=0
        if point = 1
                loop
                {
                        i++
                        if (Claw.MaxIndex() = "" or i > Claw.MaxIndex())
                                Break
                        if InStr(Claw[i],tempVar[B_Index]) > 0
                        {
                                Claw.RemoveAt(i)
                                i:=0
                        }
                }
        if point = 2
                loop
                {
                        i++
                        if (Claw.MaxIndex() = "" or i > Claw.MaxIndex())
                                Break
                        if InStr(Claw[i],NtempVar[B_Index]) > 0
                        {
                                Claw.RemoveAt(i)
                                i:=0
                        }
                }
}
return
}
```

Gambar 4.2Coding Pure Literals

c. Metode *Decide*

Metode ini adalah metode yang berfungsi memberikan nilai *True* terhadap literal secara random.

```
DecidePrep(byref dClaw,byref backside,byref choice){
Claw:=dClaw.Clone()
tempLiter:=object()
choiceVar:=0
ExLiter(Claw,tempLiter)

Random,choiceVar,1,tempLiter.MaxIndex()

backside.push(Claw.Clone())
choice.push(tempLiter[choiceVar])

Decide(Claw,tempLiter[choiceVar])
dClaw:=Claw.Clone()
return
}


Decide(byref Claw,choice,negate:=0){
tempArray:=object()
if negate=1
        choice := choice * -1
Nchoice := choice * -1

i:=0
loop
{
        i++
        if (Claw.MaxIndex() = "" or i > Claw.MaxIndex())
                Break

        tempArray:= StrSplit(Claw[i],A_Space)
        Claw[i] := StrReplace(Claw[i],Nchoice " ")

        loop % tempArray.MaxIndex()
        {

                if tempArray[A_Index] = choice
                {
                        Claw.removeat(i)
                        i:=0
                }
        }
}
return
}
```

Gambar 4.3Coding Decide

d. Metode *Fail*

Metode Fail merupakan metode yang digunakan untuk melihat apakah pada hasil iterasi penyelesaian permasalahan terdapat klausa kosong/*Fail* atau tidak, dimana nantinya digunakan sebagai basis dibutuhkannya backtracking atau tidak. Metode Fail dalam *SAT Solver* bukan merupakan metode yang berdiri sendiri, melainkan berupa kondisi pengecekan pada metode methodCheck().

```
if (Claw.MaxIndex() != "" and BackLog.MaxIndex() != "")  ;Fail & BackTrack Check
{
        tempArray := object()
        loop % Claw.MaxIndex()
        {
                tempArray := StrSplit(Claw[A_Index],A_Space)
                if tempArray[1] = "0"
                {
                        method := 4
                        break
                }
        }
}
```

Gambar 4.4Coding Fail Check

e. Metode *Backtracking*

Metode *Backtracking* merupakan metode untuk melakukan *Backtrack* dengan memanfaatkan metode *Decide* untuk membuat *Save Point* yang digunakan untuk melakukan *Backtrack* (penelusuran ulang) apabila terjadi nya kondisi tertentu yang memerlukan *Backtrack.*

```
BackTrack(byref dClaw,byref backside,byref choice){
Claw:=backside.Pop()
Decide(Claw,choice.Pop(),1)
dClaw:=Claw.Clone()
return
}
```

Gambar 4.5Coding Backtrack

4.2 Pengujian

Tahap pengujian ini dilakukan untuk mengetahui apakah *SAT Solver* yang dibuat dapat berjalan dengan lancar.Selain itu tahap ini juga berfungsi untuk melihat kelebihan dan kekurangan *SAT Solver* dalam penerapan penyelesaian permasalahan *SAT Problem.* Adapun

pengujian dilakukan dengan cara penginputan data nyata yang digunakan untuk melihat kemampuan *SAT Solver*. Hasil dari pengujian tersebut adalah sebagai berikut :

Tabel 4.1Pengujian Penyelesaian *SAT Problem*

No	Banyak Variabel	Banyak Klausa	Waktu	Benchmark	Hasil
1	1	2	0 detik	Satisfied	Satisfied
2	2	3	0 detik	Satisfied	Satisfied
3	6	6	0 detik	Satisfied	Satisfied
4	3	8	0 detik	Not Satisfied	Not Satisfied
5	3	11	0 detik	Satisfied	Satisfied
6	20	70	2 detik	Satisfied	Satisfied
7	20	90	Unlimited	Satisfied	???

Menurut hasil pengujian pada Tabel 4.1Pengujian Penyelesaian *SAT Problem*, *SAT Solver* sudah mampu menyelesaikan beberapa masalah yang termasuk golongan mudah (dapat diselesaikan secara manual dengan cepat dan tidak rumit), namun dikarenakan alasan yang belum diketahui *SAT Solver* tidak dapat menyelesaikan soal nomer 7.

Ada beberapa kemungkinan mengapa *SAT Solver* tidak mampu menyelesaikan soal nomer 7. Beberapa dari kemungkinan tersebut termasuk kesalahan logika pada metode dan perintah yang digunakan. Kesalahan logika dapat terjadi akibat cara pengerjaan *SAT Solver* yang berbasis pada iterasi dan bukan pada rekursif, sedangkan metode DPLL ini seharusnya besbasiskan pada rekursif. Adapun kemungkinan lainnya adalah memori yang tidak mencukupi akibat banyak nya data yang digunakan sebagai *backup* yang dimanfaatkan dalam melakukan *backtrack*. Oleh karena itu pengujian pun berlanjut menggunakan data *dummy* yang memiliki beberapa spesifikasi khusus. Hasil dari pengujian lebih lanjut tersebut dapat dilihat pada Tabel 4.2 berikut :

Tabel 4.2Pengujian Lanjutan

No	Banyak Variabel	Banyak Klausa	Spesifikasi Khusus	Waktu	Benchmark	Hasil
1	10	20	Berisi formula dengan banyak *backtrack*	2 detik	Not Satisfied	Not Satisfied
2	10	90	Modifikasi soal nomer 7 pada Tabel 4.1Pengujian Penyelesaian *SAT Problem* menggunakan lebih sedikit variabel	2 detik	Satisfied	Satisfied
3	40	40	Soal dengan menggunakan banyak variabel dan *decide*, namun sedikit klausa	3 detik	Satisfied	Satisfied
4	12	24	Modifikasi soal nomer 1 dengan lebih banyak variabel, klausa dan *backtrack*	Unlimited/ 2 detik	Not Satisfied	Tidak selesai /Not satisfied

Tabel 4.2 menunjukkan bahwa tidak terjadi kesalahan sintaks maupun logika pada *SAT Solver*. Hal ini dibuktikan dengan kemampuan *SAT Solver* dalam menyelesaikan soal no 1 sampai nomer 4 pada Tabel 4.2.Namun terdapat keganjalan pada *SAT Solver* dimana *SAT Solver* terkadang tidak mampu menyelesaikan soal nomor 4 walaupun ketika *SAT Solver* mampu menyelesaikan soal tersebut, *SAT Solver* sering menggunakan lebih banyak iterasi dan *backtrack* dibandingkan ketika *SAT Solver* tidak sanggup menyelesaikan soal tersebut.

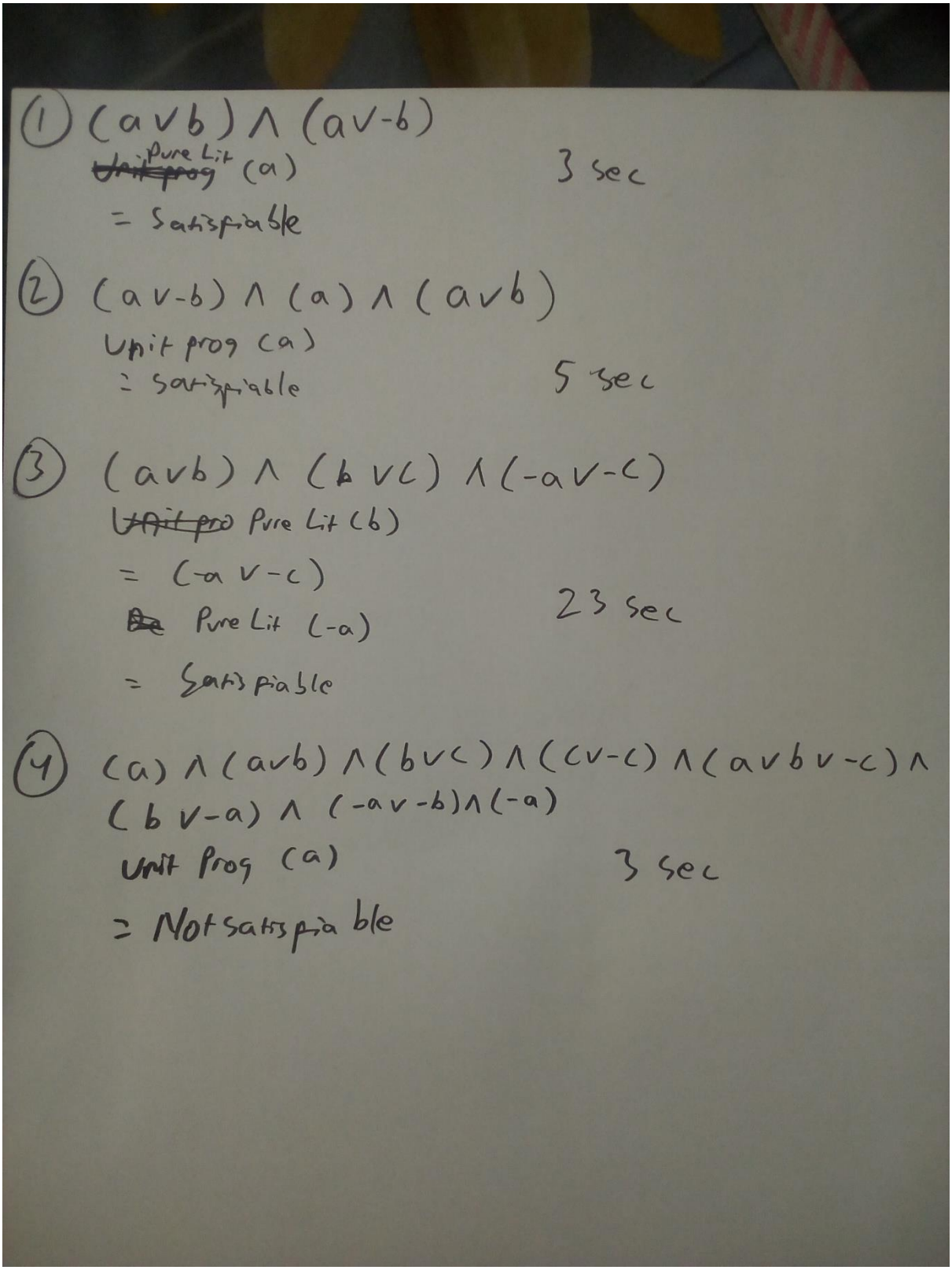
Dari uji coba diatas, dapat disimpulkan bahwa *SAT Solver* tidak mampu menyelesaikan soal nomer 7 pada tabel Tabel 4.1Pengujian Penyelesaian *SAT Problem* bukan akibat dari kesalahan logika pada metode yang digunakan, maupun kurangnya memori. Hal ini dibuktikan dengan kemampuan *SAT Solver* dalam menyelesaikan soal nomer 1, 2 dan 3 dengan jawaban yang akurat sebagai bukti tidak adanya kesalahan logika maupun sintaks pada *SAT Solver*, dan soal nomer 3 juga sebagai bukti bahwa *SAT Solver* tidak banyak mengguakan memori ketika menyimpan *backup* yang digunakan untuk *backtrack*.

Adapun hasil komparasi pengujian antara pengerjaan manual dengan menggunakan *SAT Solver* dapat dilihat pada Tabel 4.3Pengujian Komparasi *SAT Problem* :

Tabel 4.3Pengujian Komparasi *SAT Problem*

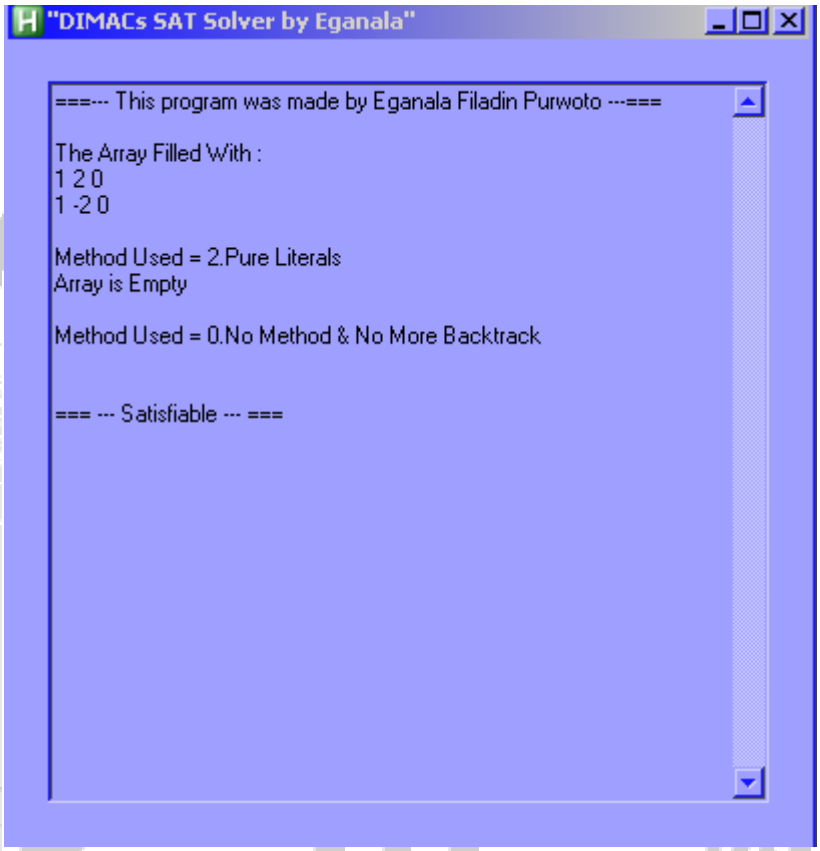
No	Banyak Variabel	Banyak Klausa	Manual		*SAT Solver*		Benchmark
			Hasil	Waktu	Hasil	Waktu	
1	2	2	Satisfied	3 sec	Satisfied	<1 sec	Satisfied
2	2	3	Satisfied	5 sec	Satisfied	<1 sec	Satisfied
3	3	3	Satisfied	23 sec	Satisfied	<1 sec	Satisfied
4	3	8	Not Satisfied	3 sec	Not Satisfied	<1 sec	Not Satisfied

Dan adapun *screenshot*hasil pengerjaan untuk tabel komparasi adalah sebagai berikut :



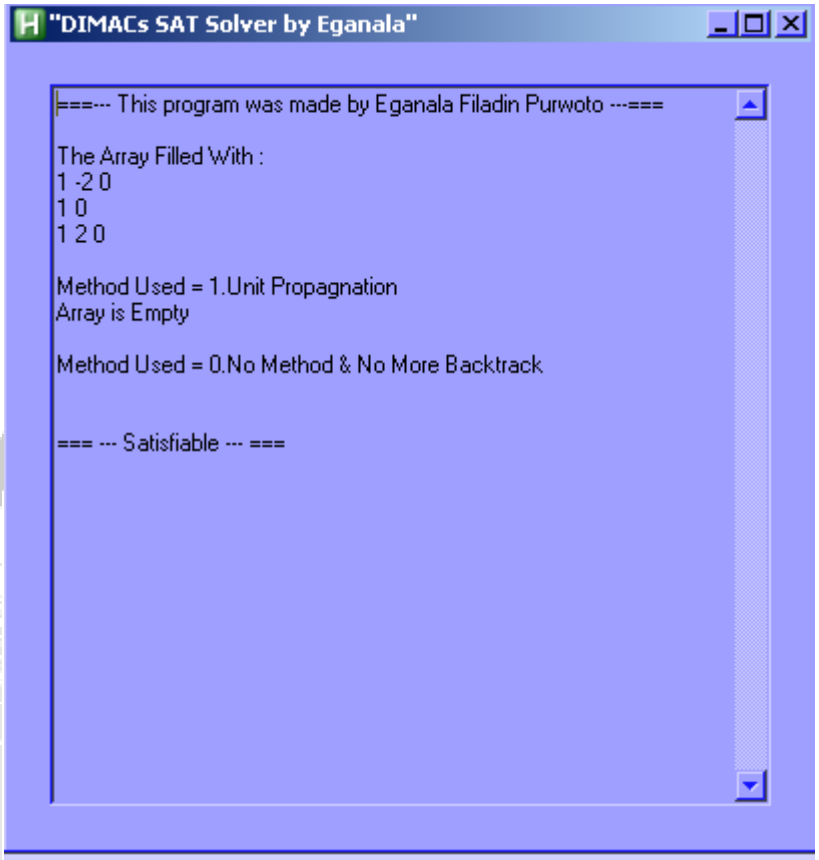
Gambar 4.6*Screenshot* Komparasi Manual

Gambar 4.6 diatas merupakan foto dari soal dan pengerjaan manual yang digunakan dalam tes komparasi. Permasalahan yang sama digunakan dalam penghitungan menggunakan *SAT Solver*.



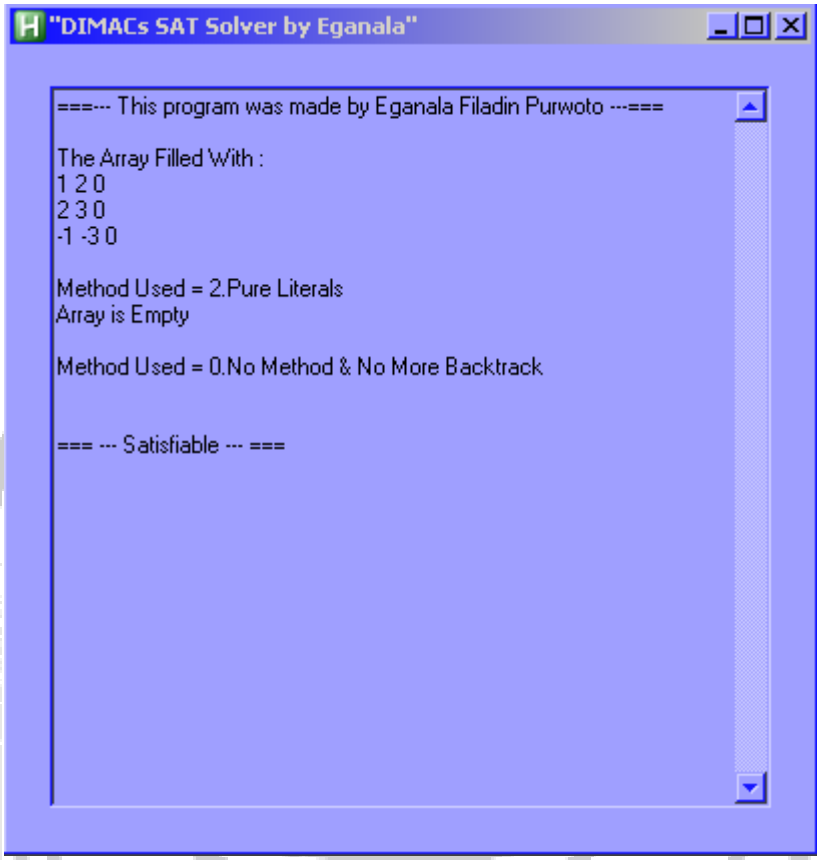
Gambar 4.7*Screenshot* Komparasi *SAT Solver* Soal 1

Gambar 4.7 diatas adalah hasil screenshot dari pengerjaan soal 1 yang sebelum nya dikerjakan manual. Soal di atas dapat diselesaikan hanya menggunakan 1 iterasi saja menggunakan metode nomer 2, yaitu pure literals dengan hasil akhir satisfiable.



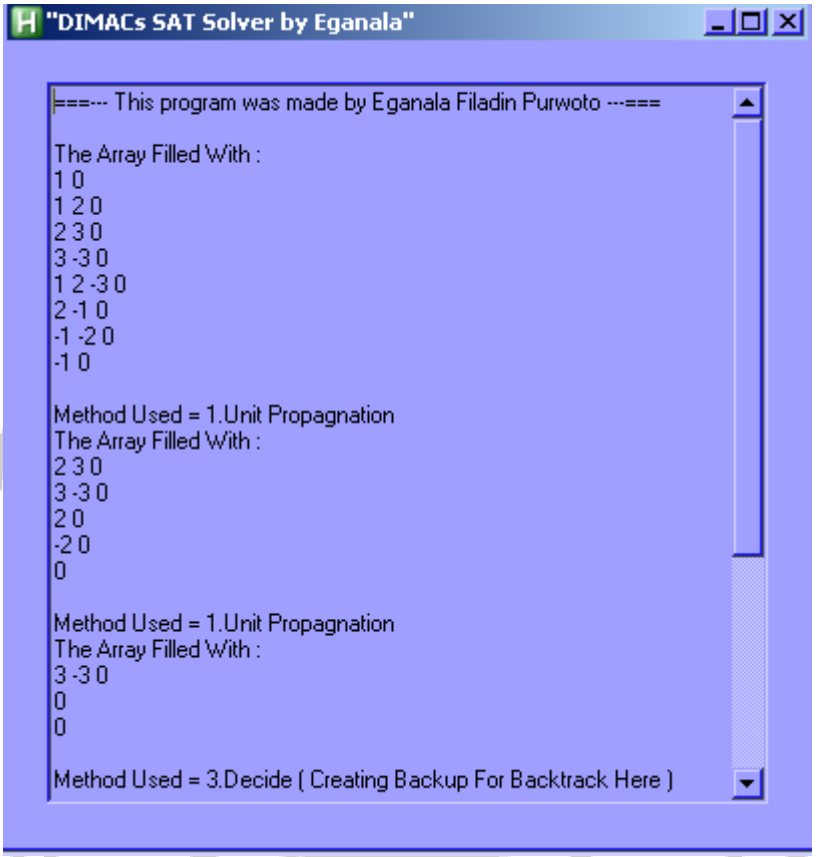
Gambar 4.8*Screenshot* Komparasi *SAT Solver* Soal 2

Gambar 4.8 juga merupakan screenshot hasil pengerjaan *SAT Solver*, namun untuk permasalahan nomer 2, dimana hanya diperlukan 1 iterasi dan 1 metode saja, yaitu metode unit propagation, untuk menyelesaikan permasalahan tersebut dengan hasil akhir *satisfiable*.



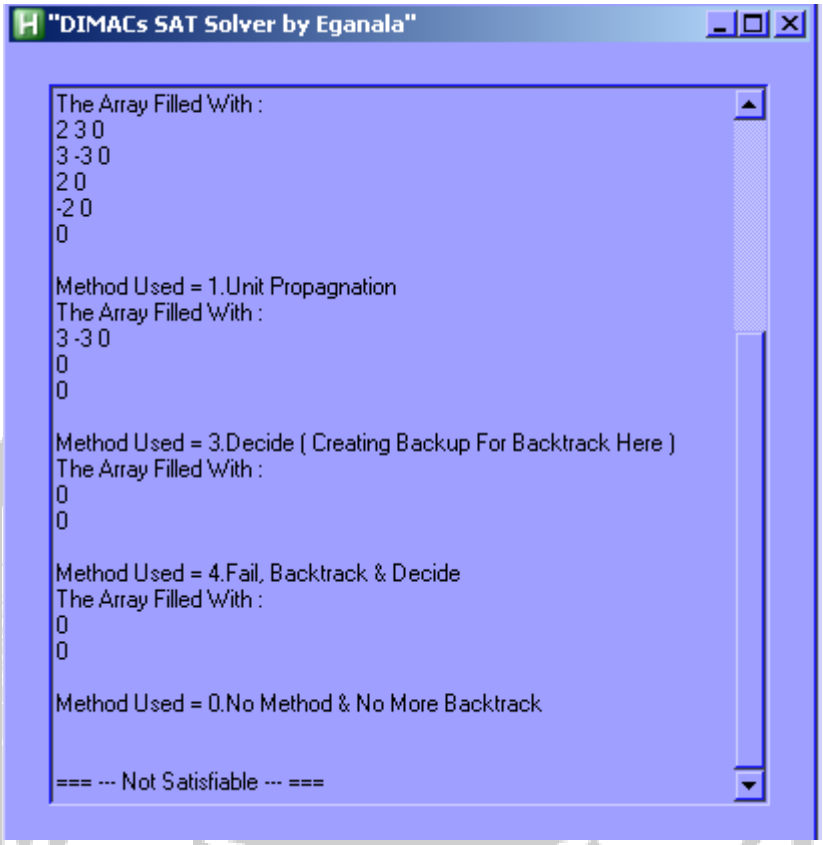
Gambar 4.9 *Screenshot* Komparasi *SAT Solver* Soal 3

Sama seperti gambar Gambar 4.8, Gambar 4.9 juga merupakan screenshot hasil pengerjaan *SAT Solver*, menggunakan permasalahan nomer 3. Sama seperti soal 1 dan 2, di sini permasalahan dapat terselesakan dengan menggunakan 1 iterasi dan 1 metode yaitu *pure literal* saja dengan hasil akhir *satisfiable*.



Gambar 4.10*Screenshot* Komparasi *SAT Solver* Soal 4-1

Gambar 4.10 juga merupakan screenshot hasil pengerjaan *SAT Solver* untuk nomer 4, namun kali ini sedikit berbeda, dikarenakan soal kali ini membutuhkan lebih dari 1 iterasi dan juga lebih dari 1 jenis metode yang digunakan dalam penyelesaiaanya, yaitu unit propagation dan decide, dimana hasil akhirnya dapat dilihat pada Gambar 4.11.



Gambar 4.11*Screenshot* Komparasi *SAT Solver* Soal 4-2

Gambar 4.11 diatas merupakan kelanjutan dari Gambar 4.10. Berdasarkan Gambar 4.11 dapat disimpulkan bahwa no 4 bukan merupakan formula/permasalahan yang *satisfiable*.