



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : CONSTRAINT SATISFACTION PROBLEM
NAMA : REZA AZZUBAIR WIJONARKO
NIM : 155150200111182
TANGGAL : 26/04/2017
JENIS : LATIHAN
ASISTEN : - ANNISA FITRIANI NUR
- RISKI PUSPA DEWI D. P..

ACC

A. DEFINISI MASALAH

1. Tentukan komponen-komponen untuk permasalahan constraint satisfaction berikut :

- Sudoku problem
- 4-Queens problem
- Pewarnaan (merah, kuning, hijau, biru) pada gambar berikut :



2. Selesaikan permasalahan 4-Queens pada papan 4x4 secara manual menggunakan algoritma Backtracking!

B. JAWAB

1.

a) Sudoku Problem

	2	6				8	1	
3			7		8			6
4				5				7
	5		1		7		9	
		3	9		5	1		
	4		3		2		5	
1				3				2
5			2		4			9
	3	8				4	6	

Contoh sudoku problem

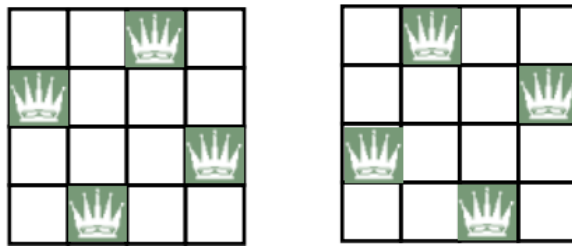
- Variabel : $9 \times 9 = 81$ variable, satu untuk masing masing bidang grid. Untuk gambar kasus sudoku di atas hanya grid kosong yang merupakan variabelnya.
- Domain : 1,2,3,4,5,6,7,8,9
- Constraint : tidak ada dua variabel yang sama dalam baris, kolom, atau blok bisa memiliki nilai yang sma.
- Solution : semua koordinat dalam sudoku terisi dengan domain dan tiddakada konflik constraint.

7	2	6	4	2	9	8	1	5
3	1	5	7	2	8	9	4	6
4	8	9	6	5	1	2	3	7
8	5	2	1	4	7	6	9	3
6	7	3	9	8	5	1	2	4
9	4	1	3	6	2	7	5	8
1	9	4	8	3	6	5	7	2
5	6	7	2	1	4	3	8	9
2	3	8	5	7	9	4	6	1

Contoh solusi sudoku

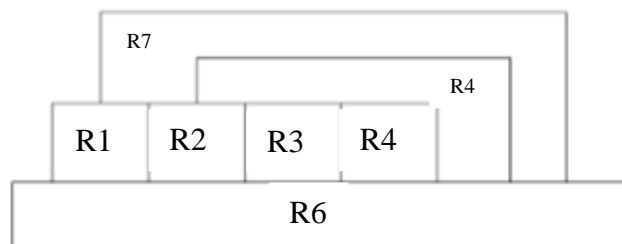
b) 4 queens problem

- Variabel : Kotak $4 \times 4 = 16$ grid yang bisa diisi dengan empat buah queen
- Domain : 4 buah Queen
- Constraint : tidak ada Queen yang sama dalam satu garis horizontal, vertikal dan diagonal.
- Solution : 4 queen yang ada sudah menempati posisi dalam kotak dan tidak menimbulkan konflik constraint (saling memakan)

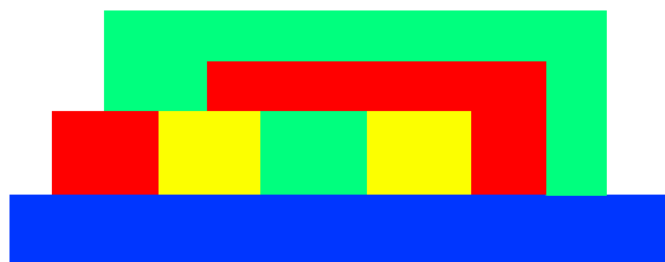


Solusi 4-queen dengan backtracking

c) Pewarnaan (merah, kuning, hijau, biru) pada gambar berikut



- Variabel : 7 buah kotak bisa dinamai dengan R1, R2, R3, R4, R5, R6, dan R7
- Domain : Merah, Kuning, Hijau, Biru
- Constraint : Tidak ada kotak yang memiliki warna sama dengan tetangganya, jadi setiap warna tidak akan saling bersebelahan.
- Solution : Semua kotak akan mempunyai warna dari salah satu domain dengan tidak ada konflik constraint.



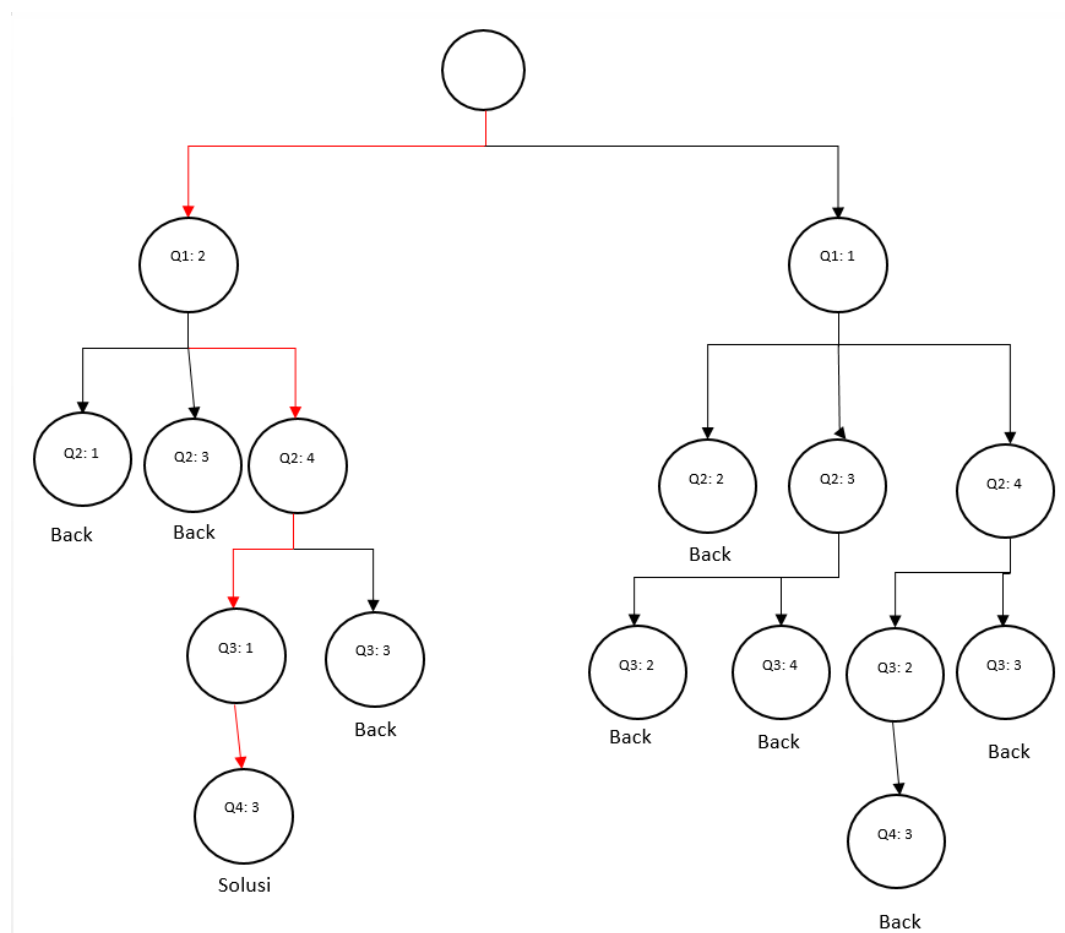
Hasil Pewarnan

2. Penyelesaian permasalahan 4-queen pada papan 4x4 menggunakan algoritma backtracking.

Penomoran bidang pada papan

Q1	1	2	3	4
Q2	1	2	3	4
Q3	1	2	3	4
Q4	1	2	3	4

Penggambaran algoritma backtracking



Visualisasi Solusi

	Q1		
			Q2
Q3			
		Q4	



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : CONSTRAINT SATISFACTION PROBLEM
NAMA : REZA AZZUBAIR WIJONARKO
NIM : 155150200111182
TANGGAL : 26/04/2017
JENIS : TUGAS
ASISTEN : - ANNISA FITRIANI NUR
- RISKI PUSPA DEWI D. P..

ACC

A. DEFINISI MASALAH

Modifikasilah program diatas sehingga dapat menyelesaikan permasalahan coloring map pada peta pulau Sumatera di bawah ini.



Domain yang digunakan adalah maksimal 3 warna. Daerah yang digunakan adalah provinsi yang telah diberi nomor. Tentukan terlebih dahulu constrain yang dibutuhkan. Modifikasi pula metode backtracking diatas dengan menambahkan MRV (Most Restricted Variable) Heuristic.

B. JAWAB

Area.java

```
1 package praktikumkb4;
2 public class Area {
3     Area pre;
4     String nama, warna;
5     Edge[] tetangga;
6     int h, level;
7
8     public Area(String nama, String warna) {
9         this.nama = nama;
10        this.warna = warna;
11    }
12
13    public Area(Area pre, String nama, String warna) {
14        this.pre = pre;
15        this.nama = nama;
16        this.warna = warna;
17    }
18
19    public void setTetangga(Edge[] tetangga) {
20        this.tetangga = tetangga;
21        h=this.tetangga.length;
22    }
23
24    @Override
25    public String toString(){
26        return String.format("Nama\t: %-20sWarna:
27%s", nama, warna);
28    }
29
30    public boolean equals(Object obj) {
31        if (this == obj) {
32            return true;
33        }
34        if (obj == null) {
35            return false;
36        }
37        if (getClass() != obj.getClass()) {
38            return false;
39        }
40        Area other = (Area) obj;
41        return nama.equalsIgnoreCase(other.nama) &&
42        warna.equalsIgnoreCase(other.warna);
43    }
44 }
```

Edge.java

```
1 package praktikumkb4;
2 public class Edge {
3     final Area target;
4     public Edge( Area tujuan ) {
5         this.target = tujuan;
6     }
7 }
```

CSP.java

```
1 package praktikumkb4;
2 import java.util.LinkedList;
3 public class CSP {
4     LinkedList<Area> rute = new LinkedList();
5     public void start(Area[] data, String[] warna, String
6 wAwal) {
7         LinkedList<Area> Dup = new LinkedList();
8         LinkedList<Area> Stack = new LinkedList();
9         boolean selesai = false;
10        int i = 0;
11        Area awal = nextMRV(data, i);
12        awal.warna = wAwal;
13        awal.level = i;
14        Stack.add(awal);
15        Dup.add(awal);
16        while (!Stack.isEmpty() && !selesai) {
17            Area curr = Stack.pop();
18            boolean able = true;
19            i = curr.level + 1;
20            Area next = nextMRV(data, i);
21            if (next != null) {
22                String nextWarna = warna[0];
23                for (Edge tetangga : next.tetangga) {
24                    for (int j = 0; j < warna.length; j++)
25                    {
26                        if
27 (!tetangga.target.warna.equalsIgnoreCase(nextWarna)) {
28                            able = true;
29                        } else {
30                            able = false;
31                            if(j+1<warna.length){
32                                nextWarna = warna[j+1];
33                            } else {
34                                nextWarna = warna[0];
35                            }
36                        }
37                    }
38                }
39                if (able) {
40                    next.warna = nextWarna;
41                    next.pre = curr;
42                    next.level = i;
43                    checkDup(Stack, Dup, next);
44                }
45                } else {
46                    selesai = true;
47                    makePath(curr);
48                }
49            }
50            if(selesai==false){
51                System.out.println("Tidak dapat diselsaikan
52 dengan "+warna.length+" warna");
53            }
54        }
55    }
```



```

56     private void checkDup(LinkedList<Area> asli,
57     LinkedList<Area> Dup, Area cek) {
58         if (!Dup.contains(cek)) {
59             asli.push(cek);
60             Dup.push(cek);
61         }
62     }
63
64     private void makePath(Area goal) {
65         rute.push(goal);
66         Area path = goal;
67         while (path.pre != null) {
68             rute.push(path.pre);
69             path = path.pre;
70         }
71     }
72
73     public void getPath() {
74         if(!rute.isEmpty()){
75             System.out.println("-----
76             -----");
77             System.out.println("Urutan Solusi Langkah");
78             System.out.println("-----
79             -----");
80             while (!rute.isEmpty()) {
81                 System.out.println(rute.pop());
82             }
83             System.out.println("-----
84             -----");
85         }
86     }
87
88     private Area nextMRV(Area[] data, int i) {
89         try {
90             for (int j = i + 1; j < data.length; j++) {
91                 if (data[i].h < data[j].h) {
92                     Area temp = data[i];
93                     data[i] = data[j];
94                     data[j] = temp;
95                 }
96             }
97             return data[i];
98         } catch (Exception e) {
99             return null;
100         }
101     }
102 }

```

Main.java

```

1     package praktikumkb4;
2     public class Main {
3
4         public static void main(String[] args) {
5             System.out.println("-----
6             -----");
7

```

8	System.out.println("CSP
9	PROBLEM");
10	System.out.println("-----
11	-----");
12	String[] nama = {"Lampung", "Sumatera Selatan",
13	"Bengkulu", "Jambi", "Sumatera Barat", "Riau", "Sumatera
14	Utara", "Aceh"};
15	Area n[] = new Area[nama.length];
16	for (int i = 0; i < n.length; i++) {
17	n[i] = new Area(nama[i], "");
18	}
19	
20	n[0].setTetangga(new Edge[]{
21	new Edge(n[1])});
22	n[1].setTetangga(new Edge[]{
23	new Edge(n[2]),
24	new Edge(n[3]),
25	new Edge(n[0])});
26	n[2].setTetangga(new Edge[]{
27	new Edge(n[1]),
28	new Edge(n[3]),
29	new Edge(n[4])});
30	n[3].setTetangga(new Edge[]{
31	new Edge(n[1]),
32	new Edge(n[2]),
33	new Edge(n[4]),
34	new Edge(n[5])});
35	n[4].setTetangga(new Edge[]{
36	new Edge(n[2]),
37	new Edge(n[3]),
38	new Edge(n[5]),
39	new Edge(n[6])});
40	n[5].setTetangga(new Edge[]{
41	new Edge(n[3]),
42	new Edge(n[4]),
43	new Edge(n[6])});
44	n[6].setTetangga(new Edge[]{
45	new Edge(n[4]),
46	new Edge(n[5]),
47	new Edge(n[7])});
48	n[7].setTetangga(new Edge[]{
49	new Edge(n[6])});
50	System.out.println("Problem : Mewarnai Peta");
51	System.out.println("Constraint : Area yang
52	bertetangga tidak bewarna sama");
53	String warna[] = {"Merah", "Hijau", "Kuning"};
54	CSP test = new CSP();
55	test.start(n, warna, warna[0]);
56	test.getPath();
57	}
58	}

Penjelasan

Area.java

- Baris 3-6: mendeklarasi instance variables atau array yang ada pada class Area h, nama, warna, level, pare dan tetangga. Variable pre digunakan untuk menyimpan area sebelumnya dan array tetangga menunjukkan hubungan dengan area lain
- Baris 8-17: constructors untuk menginisialisasi instance variables
- Baris 19-21: method setTetangga berfungsi untuk mengeset nilai tetangga sesuai dengan parameter
- Baris 25-28: method toString digunakan untuk mencetak nama dan warna area
- Baris 30-43: method equals digunakan untuk membandingkan state sekarang dengan state yang ada di parameter apakah sama atau tidak.

Edge.java

- Class Edge tidak memiliki method dan berfungsi sebagai pemhubung antar Area

CPS.java

- Baris 4: menginisialisasi instance linked list rute untuk menyimpan langkah solusi.
- Baris 5-54: instance method void start digunakan untuk memulai proses pewarnaan peta dengan parameter data yang sudah dipetakan, warna yang digunakan, dan warna awal. Dalam method ini terdapat variable Boolean selesai untuk membantu pewarnaan. Method ini akan melakukan push nilai awal ke linkedlist Stack dan Dup. Nilai awal didapatkan dengan method nextMVR berparameter data dan i. Selama Stack tidak kosong dan variable selesai false, akan dilakukan pop dari stack dan Area yang dipop disimpan di variable curr. Area berikutnya yang dipush ditentukan dengan method nextMVR berparameter data dan nilai i + nilai level dari curr. Jika nilai next null, maka pewarnaan selesai, nilai variable selesai diganti true dan buat solusi dengan makePath. Jika tidak maka method akan mengecek nilai warna dari semua tetangga next, jika memenuhi constraint (warna tidak boleh sama dengan tetangga), maka next akan dimasukkan stack dengan bantuan method checkDup.

Jika nilai variable selesai false dan perulangan berhenti, method ini akan mencetak keterangan.

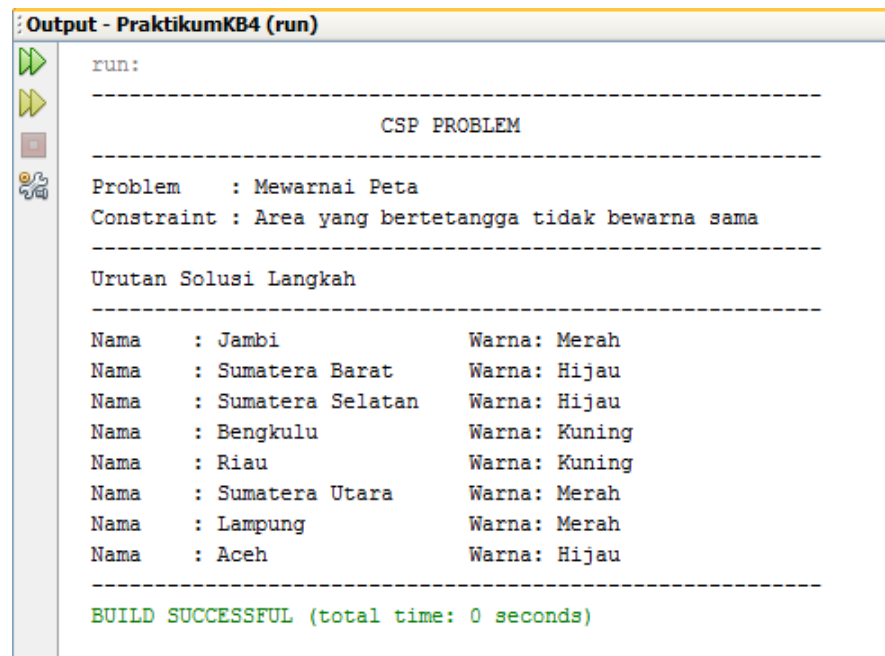
- Baris 56-62: method checkDup memiliki parameter linkedlist<Area> asli dan Dup, area yang ingin dipush. Method ini akan melakukan push ke linkedlist<State> asli dan Dup jika state yang akan dipush tidak ada di Dup.
- Baris 64-71: method makePath berparameter goal (Area terakhir). method akan melakukan push Area dan predecessor nya secara berurutan sampai ke Area yang tidak memiliki predecessor ke linkedlist rute.

- Baris 73-86: method getPath digunakan untuk mencetak solusi dengan mencetak nilai pop dari linkedlist rute.
- Baris 88-101: method nextMVR digunakan untuk mencari nilai dengan mengecek nilai heuristic (h) dari semua data array dari indeks i yang didefinisikan diparameter dan dicari yang terbesar. Nilai h sendiri adalah jumlah tetangga.

Main method

- Main method diawali dengan menginialisasi nilai Area, mengisi nilai tetangganya, mencetak keterangan problem dan constraint, dan menjalankan method start dari class CPS. Diakhiri dengan getPath untuk mendapatkan solusinya.

Screenshot:



```

Output - PraktikumKB4 (run)
run:
-----
                        CSP PROBLEM
-----
Problem      : Mewarnai Peta
Constraint   : Area yang bertetangga tidak bewarna sama
-----
Urutan Solusi Langkah
-----
Nama      : Jambi           Warna: Merah
Nama      : Sumatera Barat  Warna: Hijau
Nama      : Sumatera Selatan Warna: Hijau
Nama      : Bengkulu        Warna: Kuning
Nama      : Riau            Warna: Kuning
Nama      : Sumatera Utara   Warna: Merah
Nama      : Lampung         Warna: Merah
Nama      : Aceh            Warna: Hijau
-----
BUILD SUCCESSFUL (total time: 0 seconds)

```

C. KESIMPULAN

1. Jelaskan apa yang dimaksud dengan CSP dan berikan contoh permasalahannya!
Secara singkat, CSP (constraint satisfaction problem) adalah sebuah pemecahan masalah dengan memperhatikan constraints yang ada dari permasalahan itu. Secara detil, CSP adalah sekumpulan variabel X_1, X_2, \dots, X_n , dan sekumpulan constraints (batasan), C_1, C_2, \dots, C_m . Tiap variabel X_i , memiliki domain D_i yang tidak kosong dan memiliki kemungkinan-kemungkinan values (nilai). Tiap constraint C_i memiliki beberapa subset dari variabel dan menjelaskan kombinasi-kombinasi yang diperbolehkan dari nilai-nilai untuk subset itu. Sebuah keadaan permasalahan didefinisikan oleh assignment (pengisian) nilai-nilai sebagian atau semua variabel-variabel ($X_i = v_i, X_j = v_j, \dots$). Sebuah assignment yang tidak melanggar constraints apapun disebut sebagai assignment yang legal atau konsisten. Sebuah assignment yang komplit atau utuh adalah ketika semua variabel ter-assign oleh nilai-nilai, dan sebuah solusi dari sebuah CSP adalah assignment yang komplit yang memenuhi semua constraints. Beberapa CSP juga memiliki sebuah solusi untuk memaksimalkan sebuah fungsi objektif.
Contoh permasalahan yang dapat diselesaikan dengan CSP yaitu n-queen problem, crosswords (teka-teki silang), penjadwalan, boolean satisfiability problem (SAT), dan pewarnaan peta(map coloring)
2. Sebutkan dan jelaskan komponen CSP!
Komponen CSP ada empat, yakni
 - 1) Variable (variabel), yakni bagian permasalahan yang diisi oleh domain dengan memperhatikan constraints yang ada untuk menyelesaikan permasalahan
 - 2) Constraint, yakni batasan-batasan dari suatu permasalahan yang diperhatikan dan dipertimbangkan agar solusi yang dicari valid
 - 3) Domain, yakni bagian permasalahan yang mengisi variables
 - 4) Solution atau goal test, yakni assignments domain pada semua variables yang memenuhi constraints yang ada
3. Jelaskan bagaimana algoritma backtracking menyelesaikan permasalahan CSP?
Algoritma backtracking search (penelusuran kembali) adalah suatu bentuk algoritma depth-first-search. Backtracking dapat dilihat sebagaimana searching dalam tree karena setiap node mewakili state dan turunan dari setiap node mewakili ekstensi dari state tersebut. Pada metode backtracking, variabel diisi secara sequential selagi semua variabel relevan dengan constraint yang sudah diinisialisasi. Jika solusi partial melanggar constraint, backtracking melakukan langkah kembali ke solusi partial sebelumnya dan memilih nilai lain yang belum dicoba untuk variabel yang ingin diisi. Langkah tersebut berguna untuk menghindari eksplorasi lebih lanjut dari solusi partial yang salah.

Keuntungan backtracking: pemeriksaan consistency hanya perlu dilakukan terhadap assignment yang terakhir dilakukan karena pemeriksaan terhadap assignment yang sebelumnya sudah dilakukan sebelumnya. Pada algoritma backtracking, teknik look ahead digunakan untuk meramalkan efek pemilihan variabel branching untuk mengevaluasi nilai-nilai variabel tersebut.