Generera och testa

- Bordsplacering
- Sudoku
- Färga en karta

Generera och testa

- Programstruktur:

```
searchProblem(X) :-
    generate(X),
    testSolution(X).
```

Exempel 1: Bordsplacering

- Antag k gäster runt ett långbord.
- Vill undvika att placera vissa bredvid varandra.
- Vissa par ska placeras bredvid varandra.

Huvudprogram

```
bordsplacering(Personer, Villkor, Bord) :-
    permutation(Personer, Bord),
    uppfyller(Villkor, Bord).
```

Huvudprogram

```
bordsplacering(Personer, Villkor, Bord) :-
    permutation(Personer, Bord),
    uppfyller(Villkor, Bord).
```

- Personer: `[ann, bo, cia, ...]`
- Otillåtet par: `o(ann, bo)`
- Intillsittande par: `i(cia, dan)`

Huvudprogram

```
bordsplacering(Personer, Villkor, Bord) :-
    permutation(Personer, Bord),
    uppfyller(Villkor, Bord).
```

- Personer: `[ann, bo, cia, ...]`
- Otillåtet par: `o(ann, bo)`
- Intillsittande par: `i(cia, dan)`

```
uppfyller([], _).
```

Huvudprogram

```
bordsplacering(Personer, Villkor, Bord) :-
    permutation(Personer, Bord),
    uppfyller(Villkor, Bord).
```

- Personer: `[ann, bo, cia, ...]`
- Otillåtet par: `o(ann, bo)`
- Intillsittande par: `i(cia, dan)`

```
uppfyller([], _).
uppfyller([i(A, B)|Cs], Pl) :-
    intill(A, B, Pl),
    uppfyller(Cs, Pl).
```

Huvudprogram

```
bordsplacering(Personer, Villkor, Bord) :-
    permutation(Personer, Bord),
    uppfyller(Villkor, Bord).
```

- Personer: `[ann, bo, cia, ...]`
- Otillåtet par: `o(ann, bo)`
- Intillsittande par: `i(cia, dan)`

```
uppfyller([], _).
uppfyller([i(A, B)|Cs], Pl) :-
    intill(A, B, Pl),
    uppfyller(Cs, Pl).
uppfyller([o(A, B)|Cs], Pl) :-
    ejintill(A, B, Pl),
    uppfyller(Cs, Pl).
```

Probleminstans

```
personer([ann, bo, cia, dan, eje,
          fia, gu, han,  ia]).
bivillkor([i(cia, dan),
           o(ann, bo),
           i(eje, bo),
           o(gu, han),
           o(ia, gu)]).
```

Probleminstans

```
personer([ann, bo, cia, dan, eje,
          fia, gu, han,  ia]).
bivillkor([i(cia, dan),
           o(ann, bo),
           i(eje, bo),
           o(gu, han),
           o(ia, gu)]).
```

```
| ?- personer(P), bivillkor(C),
     bordsplacering(P, C, Pl).
```

Probleminstans

```
personer([ann, bo, cia, dan, eje,
          fia, gu, han,  ia]).
bivillkor([i(cia, dan),
           o(ann, bo),
           i(eje, bo),
           o(gu, han),
           o(ia, gu)]).
```

```
| ?- personer(P), bivillkor(C),
     bordsplacering(P, C, Pl).
C = [i(cia,dan),o(ann,bo),i(eje,bo),
o(gu,han),o(ia,gu)],
P = [ann,bo,cia,dan,eje,fia,gu,han,ia],
Pl = [ann,cia,dan,bo,eje,gu,fia,han,ia] ?
yes
```

Praktisk lösning?

- Lösningsgenerering ej effektiv!
- Jämförbar med ”permutation sort”.
- Måste generera smartare!

Exempel 2: Sudoku

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

Placera ut siffrorna 1 – 9 så att varje siffra återfinns endast en gång i varje rad, kolumn, och delkvadrat.

Ansats med generera/testa

```
sudoku(S) :-
    transform(S, ByColumn, BySubsquare),
    verifyLists(S),
    verifyLists(ByColumn),
    verifyLists(BySubsquare).
```

Ansats med generera/testa

```
sudoku(S) :-
    transform(S, ByColumn, BySubsquare),
    verifyLists(S),
    verifyLists(ByColumn),
    verifyLists(BySubsquare).

verifyLists([]).
verifyLists([L|Ls]) :-
    verifyList(L),
    verifyLists(Ls).

verifyList(L) :-
    permutation([1,2,3,4,5,6,7,8,9], L).
```

```
transform([[X11,X12,X13,X14,X15,X16,X17,X18,X19],
           [X21,X22,X23,X24,X25,X26,X27,X28,X29],
           [X31,X32,X33,X34,X35,X36,X37,X38,X39],
           [X41,X42,X43,X44,X45,X46,X47,X48,X49],
           [X51,X52,X53,X54,X55,X56,X57,X58,X59],
           [X61,X62,X63,X64,X65,X66,X67,X68,X69],
           [X71,X72,X73,X74,X75,X76,X77,X78,X79],
           [X81,X82,X83,X84,X85,X86,X87,X88,X89],
           [X91,X92,X93,X94,X95,X96,X97,X98,X99]],

          [[X11,X21,X31,X41,X51,X61,X71,X81,X91],
           [X12,X22,X32,X42,X52,X62,X72,X82,X92],
           [X13,X23,X33,X43,X53,X63,X73,X83,X93],
           [X14,X24,X34,X44,X54,X64,X74,X84,X94],
           [X15,X25,X35,X45,X55,X65,X75,X85,X95],
           [X16,X26,X36,X46,X56,X66,X76,X86,X96],
           [X17,X27,X37,X47,X57,X67,X77,X87,X97],
           [X18,X28,X38,X48,X58,X68,X78,X88,X98],
           [X19,X29,X39,X49,X59,X69,X79,X89,X99]],

          [[X11,X12,X13, X21,X22,X23, X31,X32,X33],
           [X14,X15,X16, X24,X25,X26, X34,X35,X36],
           [X17,X18,X19, X27,X28,X29, X37,X38,X39],
           [X41,X42,X43, X51,X52,X53, X61,X62,X63],
           [X44,X45,X46, X54,X55,X56, X64,X65,X66],
           [X47,X48,X49, X57,X58,X59, X67,X68,X69],
           [X71,X72,X73, X81,X82,X83, X91,X92,X93],
           [X74,X75,X76, X84,X85,X86, X94,X95,X96],
           [X77,X78,X79, X87,X88,X89, X97,X98,X99]]).
```

Hur funkar det?

- Testning snabbt (”är detta en sudoku?”)

Hur funkar det?

- Testning snabbt (”är detta en sudoku?”)
- Långsamt att lösa en sudoku!

Hur funkar det?

- Testning snabbt (”är detta en sudoku?”)
- Långsamt att lösa en sudoku!
- Klarar bara ett litet antal okända (klarar inte ICAs ”enkla” sudoku med 45 okända)

Hur funkar det?

- Testning snabbt (”är detta en sudoku?”)
- Långsamt att lösa en sudoku!
- Klarar bara ett litet antal okända (klarar inte ICAs ”enkla” sudoku med 45 okända)
- ”17 okända går på några sekunder”

Hur funkar det?

- Testning snabbt (”är detta en sudoku?”)
- Långsamt att lösa en sudoku!
- Klarar bara ett litet antal okända (klarar inte ICAs ”enkla” sudoku med 45 okända)
- ”17 okända går på några sekunder”
- Vad göra?

Hur funkar det?

- Testning snabbt (”är detta en sudoku?”)
- Långsamt att lösa en sudoku!
- Klarar bara ett litet antal okända (klarar inte ICAs ”enkla” sudoku med 45 okända)
- ”17 okända går på några sekunder”
- Vad göra?
- Generera smartare!

Exempel 3: Färga en karta

- **Sats:** Varje planär graf kan färgas med fyra färger.
- **Alltså:** Varje karta kan färgas med fyra färger.
- Skriv ett program som färgar en karta!

(Från Sterling-Shapiro: The Art of Prolog)

Datastruktur

```
region(Area, AreaColor, NeighborColors)
```

Exempelinstans:

```
[region(a, A, [B,C,D]),
 region(b, B, [A, C, D]),
 region(c, C, [A,B,D,E,F]),
 region(d, D, [A,C, F]),
 region(e, E, [B, C, F]),
 region(f, F, [C, D, E])]
```

En bit av Europa

```
%                          färg, grannars färger
map(europe,
     [region(portugal,  P, [E]),
      region(spain,     E, [F, P]),
      region(france,    F, [E, I, S, B, G, L]),
      region(belgium,   B, [F, H, L, G]),
      region(holland,   H, [B, G]),
      region(germany,   G, [F, A, S, H, B, L]),
      region(luxembourg,L,[F, B, G]),
      region(italy,     I, [F, A, S]),
      region(switzerl,  S, [F, I, A, G]),
      region(austria,   A, [I, S, G])])
).
```

Programmet

```
% colorMap(Map, ColorList)
% För varje region: Välj en färg
colorMap([],_).
colorMap([Region|Regions], Colors) :-
     colorRegion(Region, Colors),
     colorMap(Regions, Colors).
```

Programmet

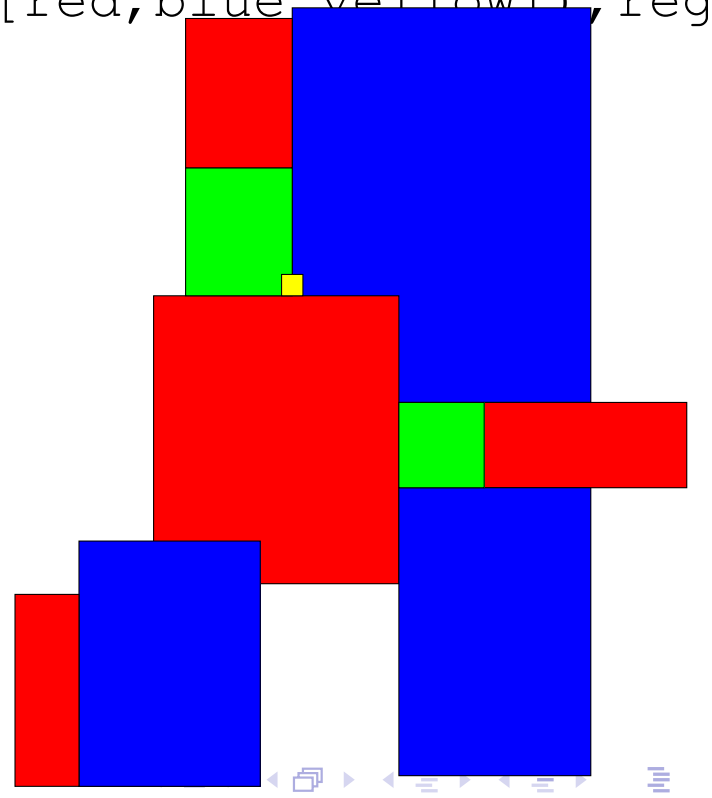
```
% colorMap(Map, ColorList)
% För varje region: Välj en färg
colorMap([],_).
colorMap([Region|Regions], Colors) :-
     colorRegion(Region, Colors),
     colorMap(Regions, Colors).

% colorRegion(Region, ColorList)
% En region tar en färg och kräver att grannregionerna
% inte använder den färgen
colorRegion(region(Name,
                   Color,
                   Neighbors), Colors) :-
     select(Color, Colors, Colors1), %Generera
     members(Neighbors, Colors1).    %Testa
```

Exempelkörning

```
| ?- map(europe, M),
     colorMap(M, [red,blue,green,yellow]).
```

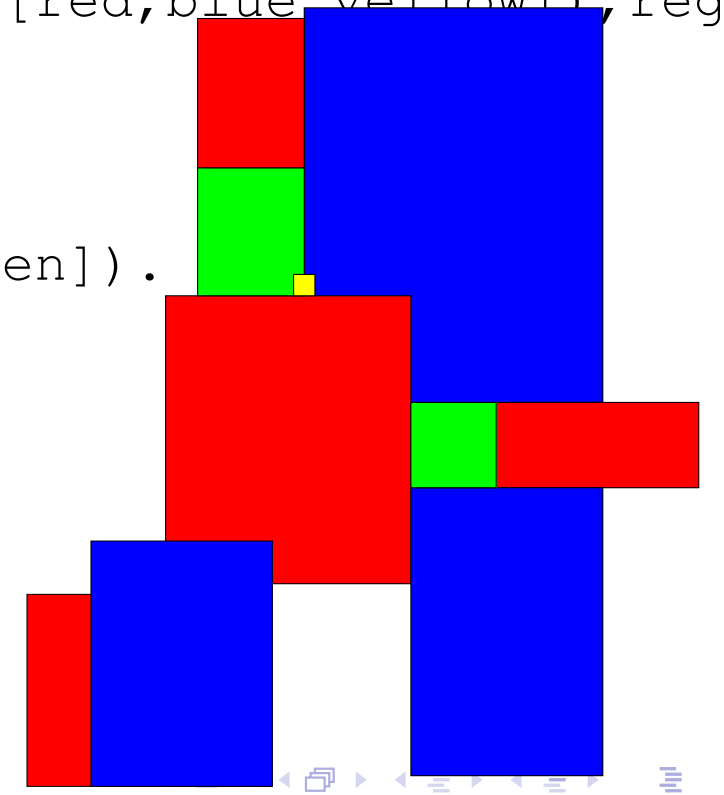
Exempelkörning

```
| ?- map(europe, M),
     colorMap(M, [red,blue,green,yellow]).
M = [region(portugal,red,[blue]),region(spain,blue,[red,
egion(belgium,blue,[red,red,green,yellow]),region(hollan
...]),region(luxembourg,green,[red,blue,yellow]),region(
ion(...)] ?
yes
| ?-
```



Exempelkörning

```
| ?- map(europe, M),
     colorMap(M, [red,blue,green,yellow]).
M = [region(portugal,red,[blue]),region(spain,blue,[red,
egion(belgium,blue,[red,red,green,yellow]),region(hollan
...]),region(luxembourg,green,[red,blue,yellow]),region(
ion(...)] ?
yes
| ?- map(europe, M),
     colorMap(M, [red,blue,green]).
no
```



Om programmet

- Implementerar en *snål* algoritm.

Om programmet

- Implementerar en *snål* algoritm.
- Riskerar vara ineffektiv.

Om programmet

- Implementerar en *snål* algoritm.
- Riskerar vara ineffektiv.
- Fler färger $\Rightarrow$ snabbare