# CSP

*progetto CORDA*
*informatica*

*alberto ferrari*

# CSP (Constraint Satisfaction Problem)

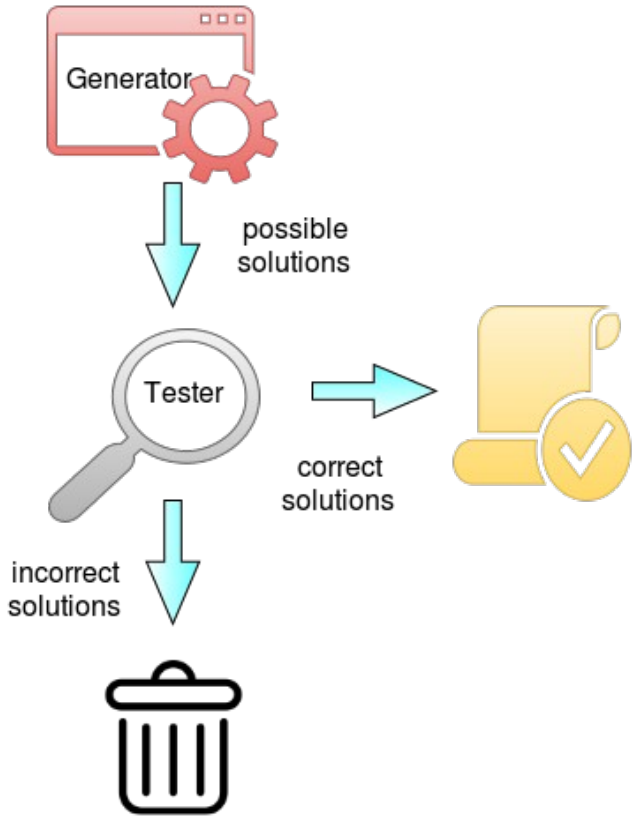
- CSP = problemi di soddisfacimento di vincoli
- caratteristiche
  - i problemi di soddisfacimento di vincoli sono caratterizzati da:
    - un insieme di variabili che possono assumere valori in un certo dominio
    - un insieme di vincoli che devono essere rispettati dai valori delle variabili

# esempio di problema CSP

- criptoaritmetica:
- assegnare un valore numerico a ogni lettera in modo che sia soddisfatta la condizione:
- COCA + COLA = OASIS
- C≠O≠A≠L≠S≠I
- C,A,L,S,I in [0..9] e O in [1..9]

# Generate & Test (G&T)

- G&T è una semplice tecnica per risolvere problemi CSP
- si assegna un valore ad ogni variabile
- si verifica se tutti i vincoli sono soddisfatti
- se i vincoli sono soddisfatti -> è stata trovata una soluzione
- altrimenti si prova con valori diversi
- il procedimento continua
  finché non ci sono più assegnamenti
  nuovi da testare
- tutte le soluzioni sono testate



# generate ...

```
/* genera tutte le combinazioni di valori delle variabili
se i vincoli sono soddifatti stampa la soluzione */

void generaValori(int indice){
    int i;
    if (indice == DIM) {          //DIM dimensione array valVar
        if(vincoliSoddisfatti())
            stampaSoluzione();
    }
    else
        for(i=0;i<=MAXVAL;i++) { //ogni variabile ha valore 0..MAXVAL
            valVar[indice]=i;
            generaValori(indice+1);
        }
}


int main(void) {
    generaValori(0);
    return 0;
}
```

# ... & test

```
/* return 1 se tutti i vincoli sono soddisfatti */
int vincoliSoddisfatti(){
  int c,a,l,s,i,o;     // valori assegnati alle variabili
  int r1,r2,r3;        // riporti
  c = valVar[0]; a = valVar[1];
  l = valVar[2]; s = valVar[3];
  i = valVar[4]; o = valVar[5];
  if (o==0)   return 0;
  if (!tuttiDiversi())   return 0;
  if (s != (a + a) % 10)    return 0;
  r1 = (a + a) / 10;
  if (i != (r1 + c + l) % 10)   return 0;
  r2 = (r1 + c + l) / 10;
  if (s != (r2 + o + o) % 10)   return 0;
  r3 = (r2 + o + o) / 10;
  if (a != (r3 + c + c) % 10)   return 0;
  if (o != (r3 + c + c) / 10)   return 0;
  return 1;
}
```

# ... & test (valutazione alterntiva)

```
/* return 1 se tutti i vincoli sono soddisfatti */
int vincoliSoddisfatti(){
  int c,a,l,s,i,o;
  int coca,cola,oasis;
  c = valVar[0]; a = valVar[1]; l = valVar[2];
  s = valVar[3]; i = valVar[4]; o = valVar[5];
  if (o==0)   return 0;
  if (!tuttiDiversi()) return 0;
  coca = c*1000 + o*100 + c*10 + a;
  cola = c*1000 + o*100 + l*10 + a;
  oasis = o*10000 + a*1000 + s*100 + i*10 + s;
  if (!(coca+cola==oasis)) // COCA + COLA = OASIS
      return 0;
  return 1;
}
```

# Generate & Test: caratteristiche

- se abbiamo **N** variabili ed un dominio con **D** valori
- il numero di test è $\mathbf{N^D}$
- **G&T non è applicabile per problemi complessi**

# proviamo a migliorare la soluzione

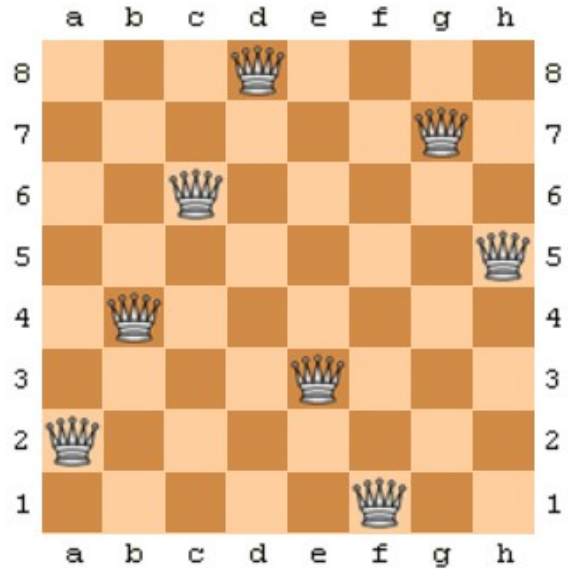
- contare il numero di valori generati complessivamente nelle soluzioni precedenti
- verificare se è possibile evitare di generare tutte le combinazioni di valori
- contare il numero di valori generati nella nuova soluzione

# Standard Backtracking (SBT)

- Standard Backtracking: a seguito di ogni assegnamento si verifica se tutti i vincoli sono soddisfatti
- se sono soddisfatti si continua verso la soluzione
- altrimenti si verifica se la variabile appena assegnata ha ancora valori da provare
  - se ***sì*** si prova con un nuovo valore
  - se ***no*** si ***torna indietro*** e si sceglie una nuova variabile
- il procedimento continua finché non ci sono più assegnamenti nuovi da provare
- tutte le soluzioni sono testate

# problema delle 8 regine

- posizionare ***8 regine*** su una ***scacchiera 8x8*** in modo che nessuna di esse possa catturarne un'altra.
- nessuna regina deve avere una colonna, riga o diagonale in comune con un'altra regina.
- il problema è un esempio del più generale problema delle ***n*** regine su una ***scacchiera n × n***



# Standard Backtracking: caratteristiche

- il limite teorico di $\mathbf{N^D}$ valori generati è sempre valido, però tipicamente non si provano tutte
- SBT sfrutta un principio generale: nel cercare una soluzione conviene accorgersi delle inconsistenze il prima possibile