

METOD 1. GIRIGA ALGORITMER (GREEDY ALGORITHMS)

ALGORITMER SOM LÖSER, ITERATIVT,
EN BIT AV PROBLEMET I TAGET.

I VARJE STEG GÖRS DET SOM GER
BÄST UTDELNING/KÖSTAR MINST.

GIRIGA ALGORITMER ÄR SPECIELLT ANVÄNDBARA
FÖR APPROXIMATIONSALGORITMER OCH HEURISTIKER.

IBLAND KAN DOCK GIRIGA ALGORITMER GE OPTIMALA
LÖSNINGAR. DÅ MÅSTE MAN BEVISA ATT ALGORITMEN
GÖR DET.

EXEMPEL PÅ PROBLEM SOM KAN LÖSAS OPTIMALT
MED EN GIRIG ALGORITM:

- HITTA KORTASTE VÄGEN MELLAN TVÅ PUNKTER I EN GRAF MED KANTVIKTER
- HITTA MINIMALA TRÄDET SOM SPÄNNER UPP EN GRAF
- GIVET EN MÅNGD AKTIVITETER MED START- OCH SLUTTIDER, HITTA STÖRSTA MÅNGDEN AKTIVITETER SOM INTE ÖVERLAPPAR

METOD 2 TOTALSÖKNING (EXHAUSTIVE SEARCH)

- GÅ IGENOM ALLA TÄNKBARA LÖSNINGAR OCH KOLLA OM DET ÄR DEN SÖKTA LÖSNINGEN
- DETTA GÖRS MED FÖRDEL REKURSIVT.

OFTA ÄR ANTALET TÄNKBARA LÖSNINGAR EXPONENTIELLT MÅNGA
(2^n) OCH DÅ GÅR DET BARA ATT ANVÄNDA FÖR SMÅ n .
TOTALSÖKNING ÄR EN METOD MAN TAR TILL I SISTA HAND.

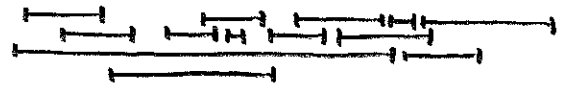
DET SVÄRASTE MED TOTALSÖKNING ÄR NORMALT ATT SE TILL
ATT MAN GÅR IGENOM VARJE TÄNKBAR LÖSNING EN (OCH HÖRST
INTE MER ÄN EN) GÅNG. ATT SEDAN KOLLA OM DET ÄR DEN SÖKTA
(ELLER OPTIMALA) LÖSNINGEN BRUKAR VARA LÄTT.

IBLAND KAN MAN REDAN INNAN EN TÄNKBAR LÖSNING ÄR
FÄRDIGKONSTRUERAD SE ATT DEN INTE ÄR MÖJLIG SOM LÖSNING.
DÅ KAN MAN STRUNTA I ATT GÅ VIDARE MED DEN OCH I STÄLLET
GÅ TILLBAKA OCH KONSTRUERA NÄSTA MÖJLIGA LÖSNING.

DETTA KALLAS BACKTRACKING.

EXEMPEL: LÖSNING AV DET GENERELLA HANDELSRESANDEPROBLEMET.

GIRIG ALGORITM FÖR AKTIVITETSPROBLEMET



IF $n=0$ THEN RETURN \emptyset
SORTERA AKTIVITETERNA EFTER SLUTTID f_i
SÅ ATT $f_1 \leq f_2 \leq f_3 \leq \dots \leq f_n$
 $S \leftarrow \{1\}$; current $\leftarrow f_1$
FÖR $i \leftarrow 2$ TO n DO
IF $s_i \geq$ current THEN
 $S \leftarrow S \cup \{i\}$; current $\leftarrow f_i$
RETURN S

BEVIS:

LÄT D VARA EN OPTIMAL DELMÄNGD AKTIVITETER
SORTERADE EFTER STIGANDE SLUTTID

OM D ÄR TOM ELLER BESTÅR AV ETT ELEMENT GÖR ALGORITMEN RÄTT.
LÄT k VARA FÖRSTA AKTIVITETEN I D .

OM $k=1$ VÄLJER ALGORITMEN SAMMA AKTIVITET (1)
 $k \neq 1$ VÄLJER ALGORITMEN EN AKTIVITET MED SLUTTID $f_i \leq f_k$
VI KAN ALLTSÅ BYTA k MOT 1 I D OCH FORTFARANDE
HA EN LÖSNING!

INGEN AKTIVITET MED STARTTID $\leq f_1$ (ÖVER 1) KAN VARA MED I D ,
SÅ VI PLOCKAR BORT DESSA.

NU HAR VI SAMMA PROBLEM FAST MED EN MINDRE MÅNGD
AKTIVITETER (SOM INTE ÖVERLAPPAR MED 1).

INDUKTION!

TSP - HANDELSRESANDEPROBLEMET (TRAVELING SALESPERSON PROBLEM)



HITTA KORTASTE TUREN SOM PASSERAR ALLA
STÄDER EN GÅNG.

OLIKA VARIANTER AV PROBLEMET:

- GENERELL TSP

PROBLEMINSTANS: GRAF MED KANTVIKTER

- EUKLIDISK TSP I DIMENSION d

PROBLEMINSTANS: STÄDERNA GIVNA SOM
KOORDINATER I \mathbb{R}^d

LÖSNING AV GENERELL TSP MED TOTALSÖKNING

DATASTRUKTURER: PERM[1..n] HÄR KONSTRUERAR VI VARJE TÄNKBAR LÖSNING, DVS PERMUTATION AV STÄDERNA
VISITED[1..n] ANGER OM EN STAD BESÖKTS HITTILLS I DEN PERMUTATION SOM KONSTRUERAS

```
TSP(n, d[1..n, 1..n]) =  
  minlength ← ∞;  
  FOR i ← 1 TO n DO VISITED[i] ← FALSE;  
  FOR i ← 1 TO n DO  
    PERM[1] ← i; VISITED[i] ← TRUE;  
    CHECKPERM(2, 0);  
    VISITED[i] ← FALSE;  
  RETURN minlength
```

```
CHECKPERM(k, length) =  
  IF k > n THEN  
    totalLength ← length + d[PERM[n], PERM[1]];  
    IF totalLength < minlength THEN minlength ← totalLength;  
  ELSE ← BACKTRACKING KAN INFÖRAS HÄR MED SÄRSÉN  
    IF length < minlength THEN  
      FOR i ← 1 TO n DO  
        IF NOT VISITED[i] THEN  
          PERM[k] ← i; VISITED[i] ← TRUE;  
          CHECKPERM(k+1, length + d[PERM[k-1], i]);  
          VISITED[i] ← FALSE;
```

TIDSKOMPLEXITET: $O(n^2 \cdot n!)$ VILKET ENKELT KAN MINSKAS TILL $O(n!)$ OM MAN HÅLLER REDA PÅ VILKA STÄDER SOM ÄNNU INTE BESÖKTS EFFEKTIVARE (TEX MED EN KÖ)

8-DAMSPROBLEMET

PLACERA UT 8 DAMER PÅ ETT SCHACKBRÄDE
UTAN ATT NÅGON DJÄS HÖTAR NÅGON ANNAN!

TOTALSÖKNINGSALGORITM:

PLACERING AV EN DAM PÅ RAD ROW:

- PRÖVA ATT PLACERA DAMEN I VARJE POSITION PÅ RADEN I TUR OCH ÖRDNING
- KAN DAMEN STÅ OHOTAD I DEN POSITIONEN SÅ PLACERAS NÄSTA DAM PÅ RAD ROW+1 UT PÅ SAMMA SÄTT, OM INTE ROW=8 FÖR DÅ HAR MAN HITTAT EN LÖSNING.

```
int queenpos[9];  
void TestRow(int row)  
{ int i;  
  for (i = 1; i <= 8; i++) {  
    queenpos[row] = i;  
    if (PosOK(row)) {  
      if (row == 8) WriteSolution();  
      else TestRow(row+1);  
    }  
  }  
}
```

STARTA MED TestRow(1);

