Algorithmus Backtracking / Branch and Bound:

1. **Für alle möglichen i-ten Kandidaten, führe die folgenden Schritte durch:**
2. **Nimm den Kandidaten in die Lösungsmenge auf.**
3. **Falls dies zu einer Lösung führen kann, führe die folgenden Schritte durch:**
4. **Zeichne die Situation auf.**
5. **Falls i=n gilt, führe Schritt 6 aus:**
6. **Falls die aktuelle Lösungsmenge optimal ist, halte diese fest.**
7. **Ansonsten führe Schritt 8 durch:**
8. **Aufruf des Algorithmus mit Parametern i+1 und aktueller Lösungsmenge.**
9. **Lösche die Aufzeichnung.**

Beispiel Backtracking (8 Damen):

positioniere (i):

```
for j=1 to 8 do

if  zeile_frei[j] and spalte_frei[i] and diagonale1_frei[i+j] and  diagonale2_frei[i-j]
then

  // Position der Dame in Spalte I ist Zeile j
  pos[i]=j;
  zeile_frei[j]=false;
  spalte_frei[i]=false;
  diagonale1_frei[i+j]=false;
  diagonale2_frei[i-j]=false;

  if i=n then

    Ausgabe pos;

  else

    positioniere (i+1)

  pos[I]=0;
  zeile_frei[j]=true;
  spalte_frei[i]=true;
  diagonale1_frei[i+j]=true;
  diagonale2_frei[i-j]=true;
```

Beispiel Branch and Bound (Rucksack):

```
behandle (i)

// Kandidat 1: i-ter Gegenstand in Rucksack
if  Gesamtgewicht(rucksack) +gewicht[I] <= maximales Gewicht
then

  rucksack[i]=true;

  if i=n then

    if Gesamtwert(rucksack) > optimaler Wert then
      optimaler Rucksack= rucksack;
      optimaler Wert = Gesamtwert(rucksack);

  else

    // Nur für Gesamtgewicht < maximales Gewicht notwendig!
    behandle(i+1);

// Kandidat 2: i-ter Gegenstand nicht in Rucksack
If erreichbarer Wert(rucksack) – wert[i] >= optimaler Wert
then

  rucksack[i]=false;

  if i=n then

    // >= ist immer erfüllt!
    if Gesamtwert(rucksack) > optimaler Wert then
      optimaler Rucksack= rucksack;
      optimaler Wert = Gesamtwert(rucksack);

  else

    behandle(i+1);
```

Beispiel Dynamic Programming (Rucksack):

Voraussetzung: Für alle i=1,...,n gilt gewicht[i] <= Maximalgewicht.

Opt(i,w) sei der Optimale Wert des Rucksacks mit Gesamtgewicht w, der die Gegenstände 1,...,i enthält. Dann gilt:

Opt(i+1,w) = Max(Opt(i,w), Opt(i,w-gewicht[i+1])+wert[i+1])

Ist Opt(i+1,w) = Opt(i,w), so ist der Gegenstand i+1 nicht im zugehörigen optimalen Rucksack, ansonsten ist i+1 im zugehörigen optimalen Rucksack.

```
For w=0 to Maximalgewicht do

  Opt(0,w)=0;

For i=1 to n do

  Opt(i,0)=0;

  For w=1 to Maximalgewicht do

    Enthalten(i,w) = false;
    Opt(i,w) = Opt(i-1,w);

    If Opt(i-1, w-gewicht[i]) + wert[I] >= Opt(i-1,w)
      then

      Opt(i,w) = Opt(i-1,w-gewicht[i]) + wert[i];
      Enthalten(i,w) = True;
```