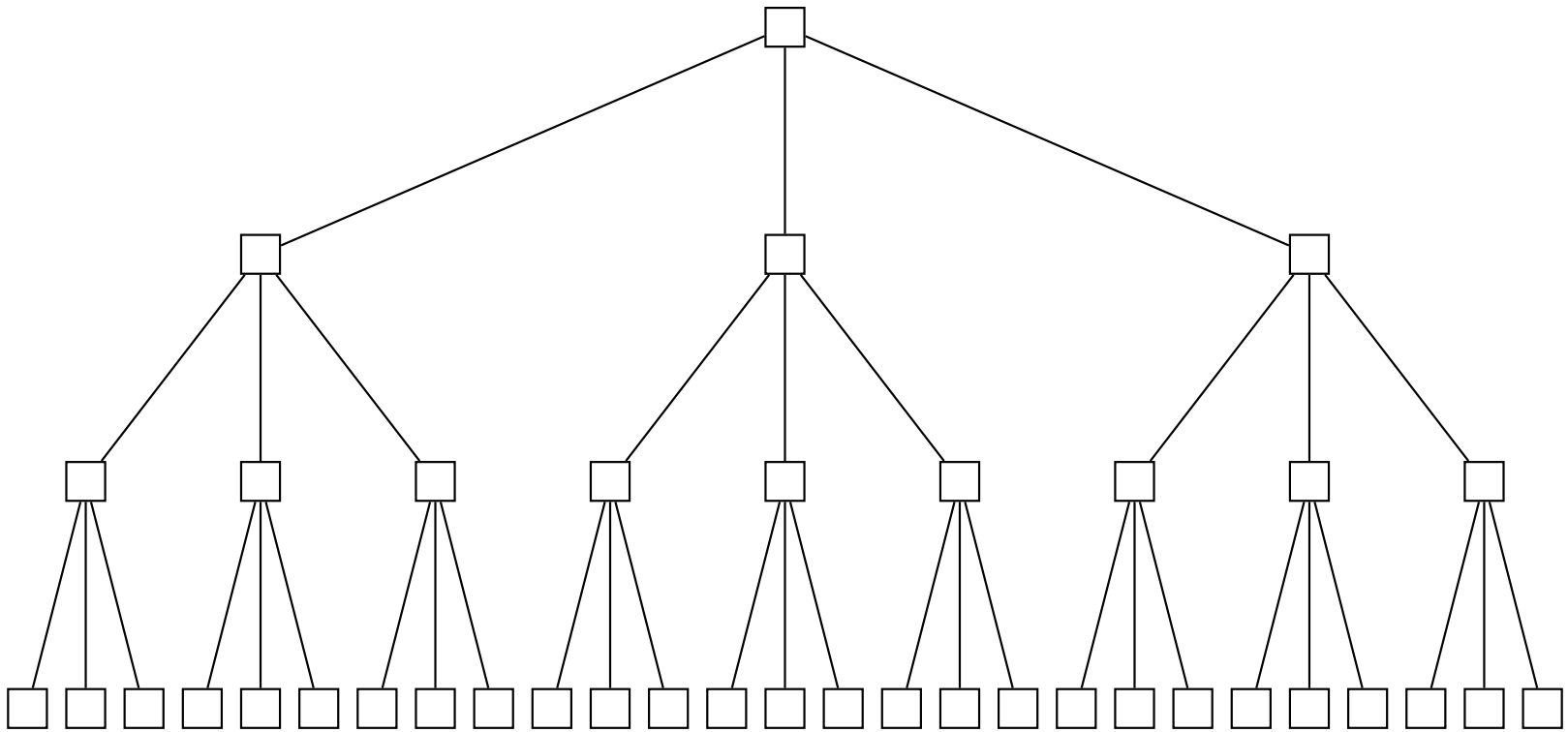
- Cerca exhaustiva
  - Backtracking
  - Branch-and bound
- Reducció i NP
- Problemes

Basat en:

- A. Atserias. *Petits Apunts sobre P i NP*. 2004–2006.
- G. Valiente. *Algorithms on Trees and Graphs*. Springer-Verlag, 2002.

- La resolució de problemes es pot entendre com la cerca d'una solució a un problema donat dins un espai de solucions: la cerca de solucions possibles al problema.
- La cerca exhaustiva consisteix en una exploració sistemàtica de l'espai de solucions possibles a un problema donat.
- La resolució de problemes per cerca exhaustiva sol comportar l'exploració d'espais molt grans de solucions, per la qual cosa resulta pràctica només per a instàncies petites del problema. Amb tot, la cerca exhaustiva és la base de tècniques més elaborades, incloent-hi backtracking, branch-and-bound, i algorismes heurístics.
- La cerca exhaustiva és el procés de generació de totes les solucions possibles al problema i de testeig si realment resolen el problema.
- Una cerca exhaustiva comporta la generació de cada una de les solucions possibles exactament una vegada, evitant així tant les repeticions com les omissions.

- L'estructura de l'espai de solucions se sol descriure mitjançant un arbre de decisió, els nodes del qual representen solucions parcials al problema i algunes de les fulles del qual representen solucions completes al problema en qüestió.



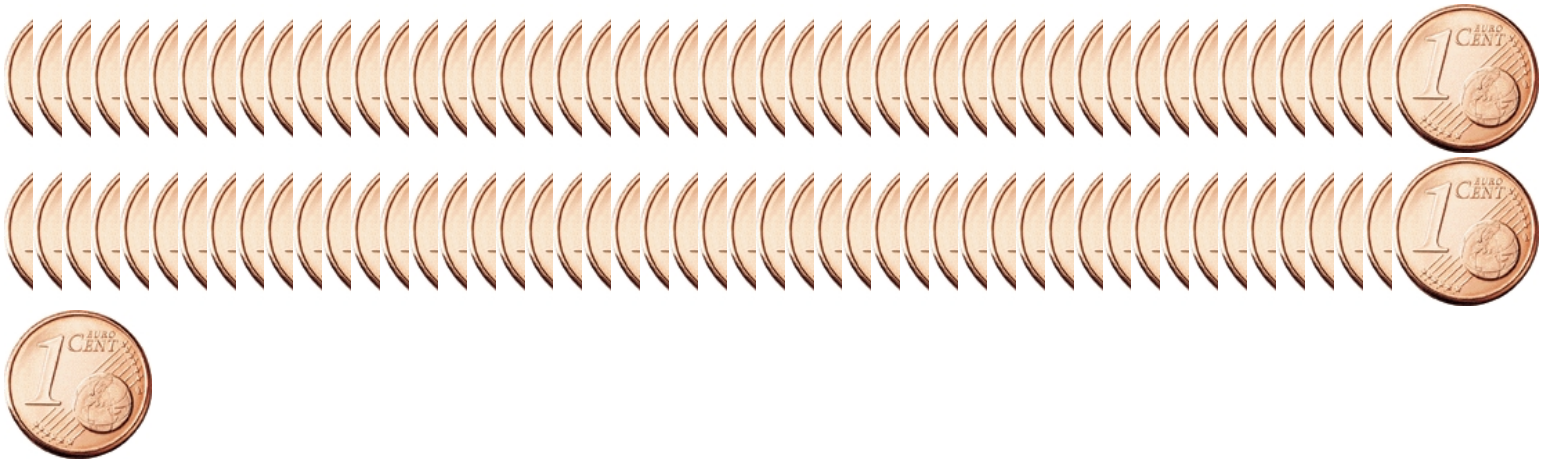
Exemple (Màquina expenedora)

Cal programar una màquina expenedora per tal que torni canvi usant el mínim nombre de monedes possibles. Dissenyeu un algorisme que resolgui aquest problema quan el canvi s'ha de donar en monedes de cèntims d'euros de curs legal (monedes de 50, 20, 10, 5, 2 i 1 cèntims).

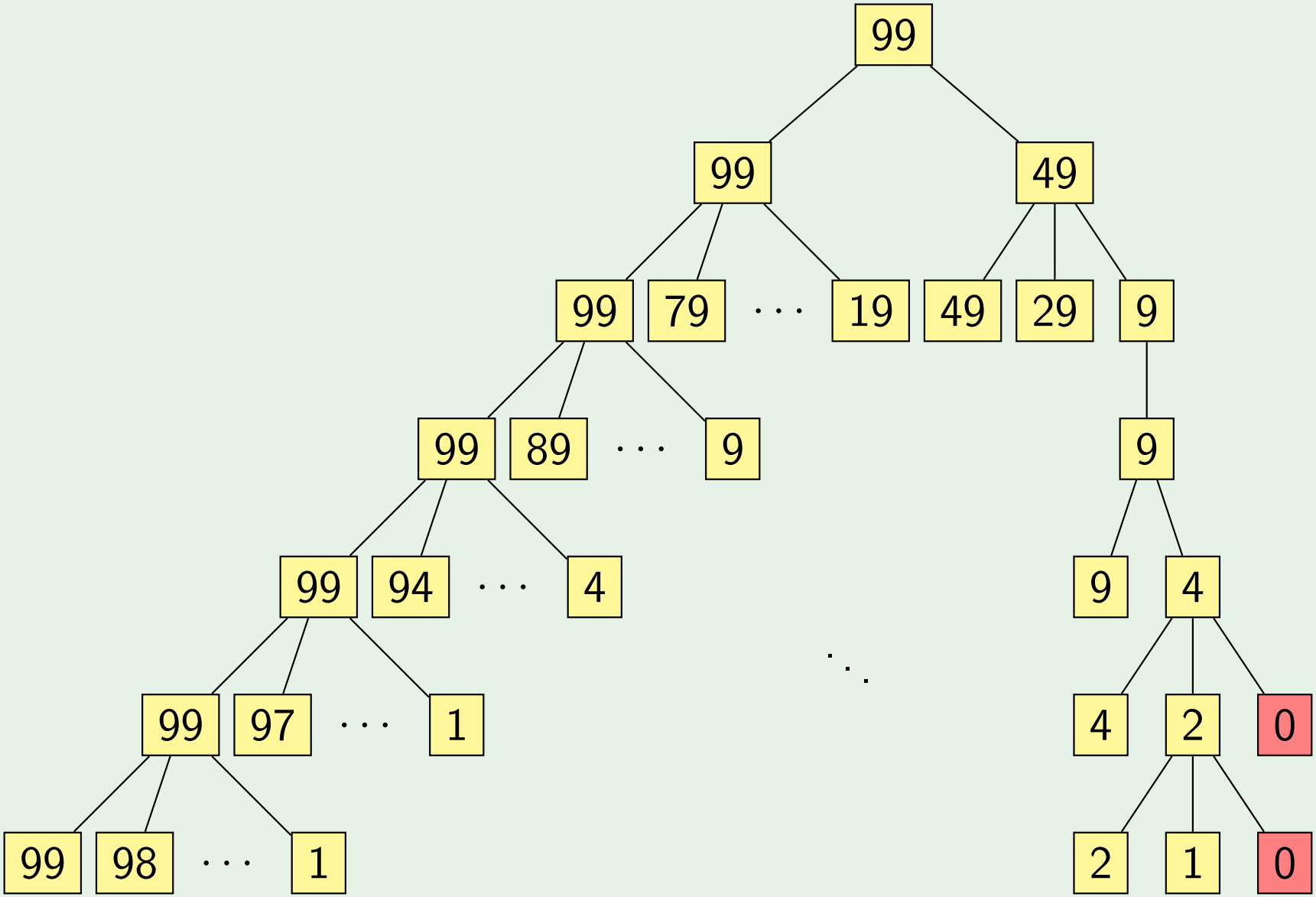
6 monedes

⋮

99 monedes



Exemple (Màquina expenedora, cont.)

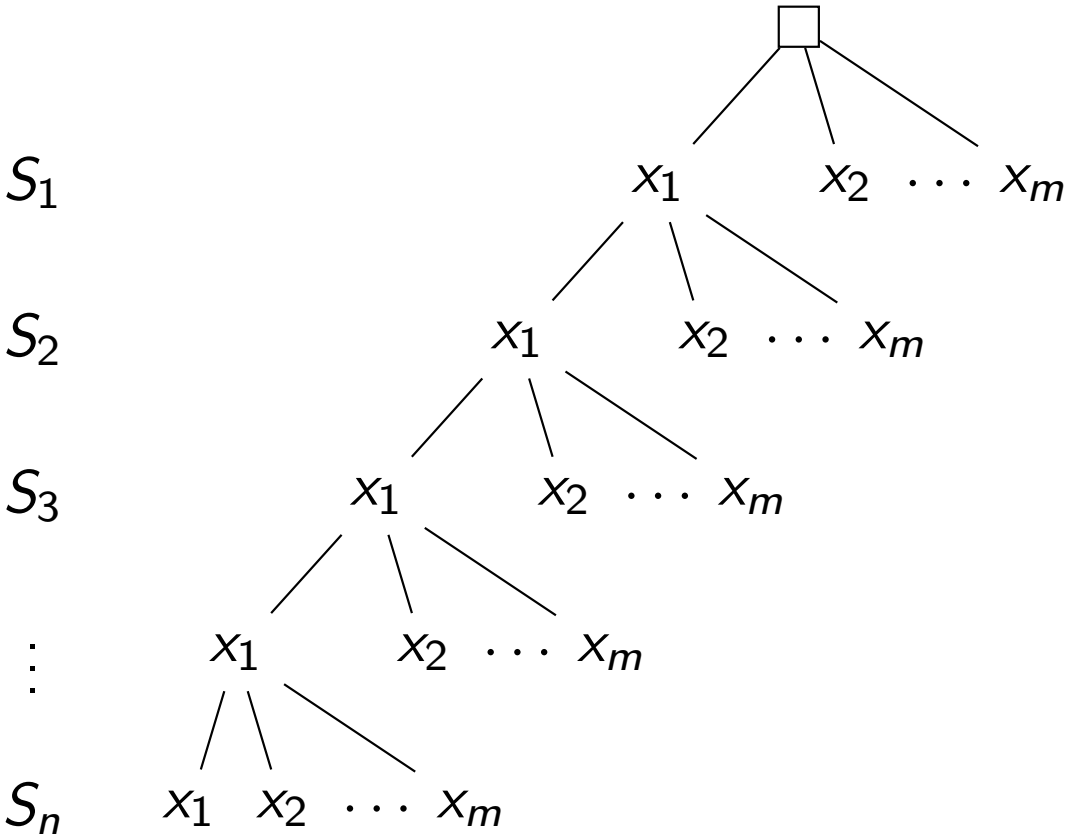


Exemple (Màquina expenedora, cont.)

- L'arbre de decisió per a la instància $y = 99$ conté $2 \cdot 5 \cdot 10 \cdot 20 \cdot 50 \cdot 100 = 10^7$ fulles, de les quals només 4366 (el 0.04366%) representen solucions possibles.
- **procedure** monedes (y: nat)
  for $x_1 := 0$ **to** 1 **do**
  for $x_2 := 0$ **to** 4 **do**
  for $x_3 := 0$ **to** 9 **do**
  for $x_4 := 0$ **to** 19 **do**
  for $x_5 := 0$ **to** 49 **do**
  for $x_6 := 0$ **to** 99 **do**
  if $50x_1 + 20x_2 + 10x_3 + 5x_4 + 2x_5 + x_6 = y$ **then**
  print $(x_1, x_2, x_3, x_4, x_5, x_6)$

- El backtracking (tornada enrera) és una tècnica general per organitzar la cerca exhaustiva de solucions d'un problema.
- Aquesta tècnica consisteix a estendre repetidament una solució parcial del problema per arribar a una solució completa del problema, estenent la representació de la solució parcial una variable cada vegada, i encongint la representació de la solució parcial (backtracking) quan no sigui possible continuar estenent-la.
- La tècnica de backtracking es pot aplicar a aquells problemes que mostren el principi del dòmino, que vol dir que si una solució parcial no compleix amb una restricció, aleshores cap extensió de la solució parcial no complirà amb aquesta restricció.
- El principi del dòmino permet aturar l'extensió d'una solució parcial així que es pugui determinar que l'extensió no portarà a una solució del problema, perquè la solució parcial no compleix amb alguna de les restriccions.

- Se sol usar un vector $X = (x_1, x_2, \ldots, x_n)$ per representar una solució d'un problema, on cada valor x_i es pren d'un conjunt ordenat $S_i = \{x_1, x_2, \ldots, x_m\}$ de valors possibles o candidats.



- En una implementació iterativa de la tècnica de backtracking, la representació de la solució s'estén i s'encongeix una variable cada vegada mitjançant l'increment i decrement del valor d'un índex k.
- **procedure** backtrack (X)
  $k := 1$
  càlcul de S_1
  while $k > 0$ **do**
    while $S_k \neq \emptyset$ **do**
      $x_k := \text{front}(S_k)$
      $\text{pop\_front}(S_k)$
      if $X = (x_1, x_2, \ldots, x_k)$ és una solució **then**
        return X
      $k := k + 1$
      càlcul de S_k
    $k := k - 1$

- La tècnica de backtracking es pot entendre també com un recorregut en preordre o en profunditat de l'arbre de decisió del problema, la qual cosa dóna una implementació recursiva natural.
- **procedure** backtrack (X, k)
  if $X = (x_1, x_2, \ldots, x_k)$ és una solució parcial **then**
    if $k = n$ **then**
      return X
    else
      $k := k + 1$
      càlcul de S_k
      for $x_k \in S_k$ **do**
        push_back(X, x_k)
        backtrack(X, k)
        pop_back(X)
- **procedure** backtrack (X)
  make_empty(X)
  backtrack$(X, 0)$

- L'anàlisi asimptòtic dels algorismes de tornada enrera sol donar un cost exponencial en el cas pitjor, donat per una fita superior de la talla de l'arbre de decisió corresponent.

Exemple (Màquina expenedora, cont.)

- $X = (x_1, x_2, \ldots, x_6)$ representa el nombre de monedes de 50, 20, 10, 5, 2 i 1 cèntims que permeten canviar $0 \leqslant y \leqslant 99$ cèntims.
  - $S_1 \subseteq \{0, 1\}$ són els valors possibles de 50 cèntims.
  - $S_2 \subseteq \{0, 1, 2, 3, 4\}$ són els valors possibles de 20 cèntims.
  - $S_3 \subseteq \{0, 1, 2, \ldots, 9\}$ són els valors possibles de 10 cèntims.
  - $S_4 \subseteq \{0, 1, 2, \ldots, 19\}$ són els valors possibles de 5 cèntims.
  - $S_5 \subseteq \{0, 1, 2, \ldots, 49\}$ són els valors possibles de 2 cèntims.
  - $S_6 \subseteq \{0, 1, 2, \ldots, 99\}$ són els valors possibles de 1 cèntim.
- Òbviament, tota solució compleix les restriccions $x_1 \in S_1$, $x_2 \in S_2$, $x_3 \in S_3$, $x_4 \in S_4$, $x_5 \in S_5$, $x_6 \in S_6$, $50x_1 + 20x_2 + 10x_3 + 5x_4 + 2x_5 + x_6 = y$.
- A més a més, recordem que interessen les solucions amb el mínim nombre $x_1 + x_2 + x_3 + x_4 + x_5 + x_6$ de monedes possibles.

Exemple (Màquina expenedora, cont.)

- L'espai de solucions per a la instància $y = 99$ conté 4366 solucions possibles, incloent-hi les següents.

x_1	x_2	x_3	x_4	x_5	x_6	c
0	0	0	0	0	99	99
0	0	0	0	1	97	98
0	0	0	0	2	95	97
0	0	0	0	3	93	96
0	0	0	0	4	91	95
0	0	0	0	5	89	94
0	0	0	0	6	87	93
0	0	0	0	7	85	92
0	0	0	0	8	83	91
0	0	0	0	9	81	90

$\cdots$

x_1	x_2	x_3	x_4	x_5	x_6	c
1	1	2	1	1	2	8
1	1	2	1	2	0	7
1	2	0	0	0	9	12
1	2	0	0	1	7	11
1	2	0	0	2	5	10
1	2	0	0	3	3	9
1	2	0	0	4	1	8
1	2	0	1	0	4	8
1	2	0	1	1	2	7
1	2	0	1	2	0	6

Exemple (Generació de les n^n permutacions amb repetició dels n nombres $1, 2, \ldots, n$)

- **procedure** backtrack (X, k)
  - **if** $X = (x_1, x_2, \ldots, x_k)$ és una solució parcial **then**
    - **if** $k = n$ **then**
      - **print** X
    - **else**
      - $k := k + 1$
      - $S_k := \{1, 2, \ldots, n\}$
      - **for** $x_k \in S_k$ **do**
        - push_back(X, x_k)
        - backtrack(X, k)
        - pop_back(X)
- **procedure** backtrack (X)
  - make_empty(X)
  - backtrack$(X, 0)$

Exemple (Generació de les n^n permutacions amb repetició dels n nombres $1, 2, \ldots, n$, cont.)

- backtrack(X) amb $n = 3$

– 1 1 1	– 2 1 1	– 3 1 1
– 1 1 2	– 2 1 2	– 3 1 2
– 1 1 3	– 2 1 3	– 3 1 3
– 1 2 1	– 2 2 1	– 3 2 1
– 1 2 2	– 2 2 2	– 3 2 2
– 1 2 3	– 2 2 3	– 3 2 3
– 1 3 1	– 2 3 1	– 3 3 1
– 1 3 2	– 2 3 2	– 3 3 2
– 1 3 3	– 2 3 3	– 3 3 3

Exemple (Generació de les $n!$ permutacions ordinàries dels n nombres $1, 2, \ldots, n$)

- **procedure** backtrack (X, k)
  if $X = (x_1, x_2, \ldots, x_k)$ és una solució parcial **then**
  if $k = n$ **then**
  print X
  else
  $k := k + 1$
  $S_k := \{1, 2, \ldots, n\} \setminus X$
  for $x_k \in S_k$ **do**
  push_back(X, x_k)
  backtrack(X, k)
  pop_back(X)
- **procedure** backtrack (X)
  make_empty(X)
  backtrack$(X, 0)$

Exemple (Generació de les $n!$ permutacions ordinàries dels n nombres $1, 2, \ldots, n$, cont.)

- backtrack(X) amb $n = 3$

  - 1 2 3
  - 1 3 2
  - 2 1 3
  - 2 3 1
  - 3 1 2
  - 3 2 1

Exemple (Generació de les m^n variacions amb repetició dels m nombres $1, 2, \ldots, m$ agafats de n en n)

- **procedure** backtrack (X, k)
  if $X = (x_1, x_2, \ldots, x_k)$ és una solució parcial **then**
  if $k = n$ **then**
  print X
  else
  $k := k + 1$
  $S_k := \{1, 2, \ldots, m\}$
  for $x_k \in S_k$ **do**
  push_back(X, x_k)
  backtrack(X, k)
  pop_back(X)
- **procedure** backtrack (X)
  make_empty(X)
  backtrack$(X, 0)$

Exemple (Generació de les m^n variacions amb repetició dels m nombres $1, 2, \ldots, m$ agafats de n en n, cont.)

- backtrack(X) amb $m = 5$ i $n = 3$

– 1 1 1	– 1 3 1	– 1 5 1	– 2 2 1	– 2 4 1	– 3 1 1
– 1 1 2	– 1 3 2	– 1 5 2	– 2 2 2	– 2 4 2	– 3 1 2
– 1 1 3	– 1 3 3	– 1 5 3	– 2 2 3	– 2 4 3	– 3 1 3
– 1 1 4	– 1 3 4	– 1 5 4	– 2 2 4	– 2 4 4	– 3 1 4
– 1 1 5	– 1 3 5	– 1 5 5	– 2 2 5	– 2 4 5	– 3 1 5
– 1 2 1	– 1 4 1	– 2 1 1	– 2 3 1	– 2 5 1	– 3 2 1
– 1 2 2	– 1 4 2	– 2 1 2	– 2 3 2	– 2 5 2	– 3 2 2
– 1 2 3	– 1 4 3	– 2 1 3	– 2 3 3	– 2 5 3	– 3 2 3
– 1 2 4	– 1 4 4	– 2 1 4	– 2 3 4	– 2 5 4	– 3 2 4
– 1 2 5	– 1 4 5	– 2 1 5	– 2 3 5	– 2 5 5	– 3 2 5

Exemple (Generació de les m^n variacions amb repetició dels m nombres $1, 2, \ldots, m$ agafats de n en n, cont.)

- backtrack(X) amb $m = 5$ i $n = 3$

– 3 3 1	– 3 5 1	– 4 2 1	– 4 4 1	– 5 1 1	– 5 3 1
– 3 3 2	– 3 5 2	– 4 2 2	– 4 4 2	– 5 1 2	– 5 3 2
– 3 3 3	– 3 5 3	– 4 2 3	– 4 4 3	– 5 1 3	– 5 3 3
– 3 3 4	– 3 5 4	– 4 2 4	– 4 4 4	– 5 1 4	– 5 3 4
– 3 3 5	– 3 5 5	– 4 2 5	– 4 4 5	– 5 1 5	– 5 3 5
– 3 4 1	– 4 1 1	– 4 3 1	– 4 5 1	– 5 2 1	– 5 4 1
– 3 4 2	– 4 1 2	– 4 3 2	– 4 5 2	– 5 2 2	– 5 4 2
– 3 4 3	– 4 1 3	– 4 3 3	– 4 5 3	– 5 2 3	– 5 4 3
– 3 4 4	– 4 1 4	– 4 3 4	– 4 5 4	– 5 2 4	– 5 4 4
– 3 4 5	– 4 1 5	– 4 3 5	– 4 5 5	– 5 2 5	– 5 4 5

Exemple (Generació de les m^n variacions amb repetició dels m nombres $1, 2, \ldots, m$ agafats de n en n, cont.)

- backtrack(X) amb $m = 5$ i $n = 3$

– 5 5 1 – 5 5 2 – 5 5 3 – 5 5 4 – 5 5 5

Exemple (Generació de les $m!/(m-n)!$ variacions ordinàries dels m nombres $1, 2, \ldots, m$ agafats de n en n)

- **procedure** backtrack (X, k)
  - **if** $X = (x_1, x_2, \ldots, x_k)$ és una solució parcial **then**
    - **if** $k = n$ **then**
      - **print** X
    - **else**
      - $k := k + 1$
      - $S_k := \{1, 2, \ldots, m\} \setminus X$
      - **for** $x_k \in S_k$ **do**
        - push_back(X, x_k)
        - backtrack(X, k)
        - pop_back(X)
- **procedure** backtrack (X)
  - make_empty(X)
  - backtrack$(X, 0)$

Exemple (Generació de les $m!/(m-n)!$ variacions ordinàries dels m nombres $1, 2, \ldots, m$ agafats de n en n, cont.)

- backtrack(X) amb $m = 5$ i $n = 3$

– 1 2 3	– 1 5 3	– 2 4 5	– 3 4 1	– 4 2 3	– 5 1 4
– 1 2 4	– 1 5 4	– 2 5 1	– 3 4 2	– 4 2 5	– 5 2 1
– 1 2 5	– 2 1 3	– 2 5 3	– 3 4 5	– 4 3 1	– 5 2 3
– 1 3 2	– 2 1 4	– 2 5 4	– 3 5 1	– 4 3 2	– 5 2 4
– 1 3 4	– 2 1 5	– 3 1 2	– 3 5 2	– 4 3 5	– 5 3 1
– 1 3 5	– 2 3 1	– 3 1 4	– 3 5 4	– 4 5 1	– 5 3 2
– 1 4 2	– 2 3 4	– 3 1 5	– 4 1 2	– 4 5 2	– 5 3 4
– 1 4 3	– 2 3 5	– 3 2 1	– 4 1 3	– 4 5 3	– 5 4 1
– 1 4 5	– 2 4 1	– 3 2 4	– 4 1 5	– 5 1 2	– 5 4 2
– 1 5 2	– 2 4 3	– 3 2 5	– 4 2 1	– 5 1 3	– 5 4 3

- La tècnica de branch-and-bound (ramificació i poda) consisteix a recordar la solució òptima (de menor cost) trobada en cada pas durant la cerca exhaustiva de solucions, i usar el cost de la solució òptima trobada fins al moment com una fita inferior del cost d'una solució òptima del problema, per poder descartar solucions parcials així que es pugui determinar que aquestes no representaran cap millora de la solució òptima trobada fins al moment.
- La tècnica de branch-and-bound es pot aplicar a aquells problemes que mostren una extensió del principi del dòmino, que vol dir que si una solució parcial no compleix amb una restricció, aleshores cap extensió de la solució parcial no complirà amb aquesta restricció i a més a més, el cost de tota extensió de la solució parcial serà més gran o igual que el cost de la solució parcial mateixa.

- El principi del dòmino estès permet aturar l'extensió d'una solució parcial així que es pugui determinar que l'extensió no portarà a una solució del problema, perquè la solució parcial no compleix amb alguna de les restriccions, i també així que es pugui determinar que l'extensió no portarà a una solució òptima del problema, perquè el cost de la solució parcial arriba a o supera el cost de la solució òptima trobada fins al moment.

- **procedure** branch-and-bound (X, k, X_ℓ)
  if cost$(X) <$ cost(X_ℓ) **then**
  if $X = (x_1, x_2, \ldots, x_k)$ és una solució parcial **then**
  if $k = n$ **then**
  $X_\ell := X$
  else
  $k := k + 1$
  càlcul de S_k
  for $x_k \in S_k$ **do**
  push_back(X, x_k)
  branch-and-bound(X, k, X_ℓ)
  pop_back(X)
- **procedure** branch-and-bound (X)
  make_empty(X)
  $X_\ell := (\max_1, \max_2, \ldots, \max_k)$
  branch-and-bound$(X, 0, X_\ell)$

Exemple (Màquina expenedora, cont.)

- L'espai de solucions per a la instància $y = 99$ conté 94 solucions potencialment òptimes, incloent-hi les següents.

x_1	x_2	x_3	x_4	x_5	x_6	c
0	0	0	0	0	99	99
0	0	0	0	1	97	98
0	0	0	0	2	95	97
0	0	0	0	3	93	96
0	0	0	0	4	91	95
0	0	0	0	5	89	94
0	0	0	0	6	87	93
0	0	0	0	7	85	92
0	0	0	0	8	83	91
0	0	0	0	9	81	90

$\cdots$

x_1	x_2	x_3	x_4	x_5	x_6	c
0	0	6	7	2	0	15
0	0	7	5	2	0	14
0	0	8	3	2	0	13
0	0	9	1	2	0	12
0	1	7	1	2	0	11
0	2	5	1	2	0	10
0	3	3	1	2	0	9
0	4	1	1	2	0	8
1	1	2	1	2	0	7
1	2	0	1	2	0	6

- Un problema computacional es pot formular com segueix. Donada una entrada $x \in E$, trobar, si existeix, una sortida $y \in S$ tal que $(x, y) \in R$.
- Així doncs, un problema computacional ve donat per
  - Un conjunt E d'entrades possibles
  - Un conjunt S de sortides possibles
  - Una relació $R \subseteq E \times S$ d'entrada/sortida
  - Una funció $|\cdot| : E \longrightarrow \mathbb{N}$ de talla
- La relació d'entrada/sortida R indica els parells (x, y) formats per una entrada $x \in E$ i una sortida $y \in S$ que són solucions vàlides
- La talla d'una entrada $x \in E$ es denota per $|x|$.

Exemple (Graf eulerià)

Donat un graf connex no dirigit G, obtenir un cicle eulerià de G si aquest existeix.

- Un problema decisional es pot formular com segueix. Donada una entrada $x \in E$, determinar si $x \in L$.
- Així doncs, un problema decisional ve donat per
  - Un conjunt E d'entrades possibles
  - Un subconjunt $L \subseteq E$ d'instàncies positives
  - Una funció $|\cdot| : E \to \mathbb{N}$ de talla
- Un problema decisional és el cas particular de problema computacional en què $S = \{0, 1\}$ i la relació d'entrada sortida és una funció $R : E \to S$, és a dir, per a cada $x \in E$ existeix un únic $y \in S$ tal que $(x, y) \in R$.

Exemple (Graf eulerià)

Donat un graf connex no dirigit G, determinar si G és eulerià.

- E és el conjunt de tots els grafs no dirigits i connexos
- $L \subseteq E$ és el subconjunt de tots els grafs no dirigits i connexos que són eulerians

- Sigui $A : E \longrightarrow S$ un algorisme que pren entrades de E i retorna sortides de S.
- Per a cada entrada $x \in E$, sigui $T_A(x)$ el nombre de passos que fa l'algorisme A amb entrada x.
- Per a cada $n \in \mathbb{N}$, sigui $t_A(n) = \max\{ T_A(x) : x \in E, |x| = n\}$.
- Direm que A és un algorisme polinòmic si existeix una constant $c \geqslant 0$ tal que $t_A(n) = O(n^c)$.

- Sigui $L \subseteq E$ un problema decisional.
- Direm que L és decidible en temps polinòmic si existeix un algorisme polinòmic $A : E \to \{0, 1\}$ tal que $x \in L \leftrightarrow A(x) = 1$ per a tot $x \in E$. En aquest cas, direm també que $L \in \mathbf{P}$, la classe de problemes que són decidibles en temps polinòmic.
- Direm que L és decidible en temps polinòmic indeterminista si existeix un polinomi $p(n)$ i un algorisme polinòmic $B : E \times E' \to \{0, 1\}$ tal que $x \in L \leftrightarrow \exists y \in E' : |y| \leqslant p(|x|), B(x, y) = 1$ per a tot $x \in E$. En aquest cas, direm també que $L \in \mathbf{NP}$, la classe de problemes que són decidibles en temps polinòmic indeterminista.
- Així doncs, un problema decisional és decidible en temps polinòmic indeterminista si les instàncies positives del problema tenen certificats de talla polinòmica que es poden verificar en temps polinòmic.

- Siguin $L \subseteq E$ i $L' \subseteq E'$ problemes decisionals.
- Direm que L es redueix a L' en temps polinòmic si existeix un algorisme polinòmic $A : E \to E'$ tal que $x \in L \leftrightarrow A(x) \in L'$ per a tot $x \in E$. En aquest cas, direm també que A és una reducció polinòmica de L a L'.
- Així doncs, si $A' : E' \to \{0, 1\}$ és un algorisme per a L' i $A : E \to E'$ és una reducció polinòmica de L a L', aleshores la composició $A' \circ A : E \to \{0, 1\}$ és un algorisme per a L. S'ha reduït el problema L al problema L'.
- Direm que un problema decisional $L \subseteq E$ és **NP**-hard si tot problema $L' \in \mathbf{NP}$ es redueix a L.
- Direm que un problema decisional $L \subseteq E$ és **NP**-complet si L és **NP**-hard i a més a més, $L \in \mathbf{NP}$.
- Així doncs, un problema **NP**-hard o **NP**-complet és almenys tan difícil com qualsevol problema de **NP** perquè si el poguéssim resoldre amb un algorisme polinòmic, aleshores tots els problemes de **NP** també es podrien resoldre amb un algorisme polinòmic.

Exemple (CIRCUIT-SAT)

Donat un circuit booleà C amb connectives AND, OR, NOT, entrades $x_1, \ldots, x_n$ i una única sortida, determinar si C és satisfactible.

- Un circuit booleà és un graf dirigit acíclic en què tots els vèrtexs tenen grau d'entrada 0, 1 o 2, i en què hi ha exactament un vèrtex amb grau de sortida 0.
- Els vèrtexs amb grau d'entrada 0 són entrades i estan etiquetats amb una variable x_i.
- Els vèrtexs amb grau d'entrada 1 porten l'etiqueta NOT.
- Els vèrtexs amb grau d'entrada 2 porten l'etiqueta AND o OR segons el tipus de porta lògica que representen.
- La sortida del circuit és l'únic vèrtex amb grau de sortida 0.

• C és satisfactible si i només si existeixen $a_1, \ldots, a_n \in \{0, 1\}$ tals que $C[x_1 := a_1, \ldots, x_n := a_n] = 1$.

Teorema (Cook, 1971, ACM Turing Award 1982)

CIRCUIT-SAT és **NP***-complet.*

Demostració.

CIRCUIT-SAT $\in$ **NP**. Qualsevol algorisme pot acabar essent implementat amb un circuit electrònic amb portes AND, OR i NOT si l'entrada es codifica amb valors booleans. Sigui $L \in$ **NP** amb verificador B i certificats afitats pel polinomi $p(n)$. Sigui també C_n el circuit que implementa B amb entrades de talla n. Aleshores, per a tot $x \in E$ de talla $|x| = n$,

$$\begin{aligned} x \in L &\leftrightarrow \exists y : |y| \leqslant p(|x|), B(x,y) = 1 \\ &\leftrightarrow \exists y : |y| \leqslant p(|x|), C_n(x,y) = 1 \\ &\leftrightarrow C_n(x,\cdot) \text{ és satisfactible.} \end{aligned}$$

La funció $x \longrightarrow C_n(x,\cdot)$ és doncs una reducció polinòmica de L a CIRCUIT-SAT. □

Exemple (3-SAT)

Donat un conjunt de clàusules $\{C_1, \ldots, C_m\}$ de com a molt tres literals cadascuna, determinar si la conjunció $C_1 \wedge \cdots \wedge C_m$ és satisfactible.

Teorema

3-SAT és **NP***-complet.*

Demostració.

Per reducció de CIRCUIT-SAT (a 3-SAT), doncs 3-SAT $\in$ **NP**. Sigui C un circuit booleà amb connectives AND, OR, NOT, entrades $x_1, \ldots, x_n$ i una única sortida. Construim un conjunt de clàusules $F = \{C_1, \ldots, C_m\}$ tal que C és satisfactible si i només si $C_1 \wedge \cdots \wedge C_m$ també ho és, com segueix. Per cada porta lògica u de C, incloses les entrades, definim una nova variable y_u i afegim les clàusules següents (de com a molt tres literals) a F.

Demostració.

- Si u és una entrada de C etiquetada x_i, afegim a F les clàusules

$$\neg y_u \vee x_i \qquad y_u \vee \neg x_i$$

- Si u és una porta lògica NOT i el cable ve de la porta lògica v, afegim a F les clàusules

$$\neg y_v \vee \neg y_u \qquad y_v \vee y_u$$

- Si u és una porta lògica AND i els cables venen de v_1 i v_2, afegim a F les clàusules

$$\neg y_{v_1} \vee \neg y_{v_2} \vee y_u$$
$$\neg y_u \vee y_{v_1} \qquad \neg y_u \vee y_{v_2}$$

- Si u és una porta lògica OR i els cables venen de v_1 i v_2, afegim a F les clàusules

$$\neg y_{v_1} \vee y_u \qquad \neg y_{v_2} \vee y_u$$
$$\neg y_u \vee y_{v_1} \qquad \neg y_u \vee y_{v_2}$$

- Si u és la sortida de C, afegim la clàusula y_u a F.

Demostració.

El conjunt de clàusules F es pot construir a partir de C en temps polinòmic. A més, C és satisfactible si i només si la conjunció de les clàusules de F és satisfactible. Per exemple, una porta lògica AND amb entrades v_1 i v_2 i sortida u equival a l'expressió $v_1 \wedge v_2 = u$, on les variables corresponents poden prendre els valors a la taula de l'esquerra.

y_{v_1}	y_{v_2}	y_u
0	0	0
0	1	0
1	0	0
1	1	1

y_{v_1}	y_{v_2}	y_u
0	0	1
0	1	1
1	0	1
1	1	0

$$
\begin{array}{c}
y_{v_1} \vee y_{v_2} \vee \neg y_u \\
y_{v_1} \vee \neg y_{v_2} \vee \neg y_u \\
\neg y_{v_1} \vee y_{v_2} \vee \neg y_u \\
\neg y_{v_1} \vee \neg y_{v_2} \vee y_u
\end{array}
$$

Les variables no poden prendre els valors a la taula del mig, la qual cosa es pot expressar amb les clàusules (de tres literals) de la dreta i també amb la clàusula 3-CNF $(\neg y_{v_1} \vee \neg y_{v_2} \vee y_u) \wedge (y_{v_1} \vee \neg y_u) \wedge (y_{v_2} \vee \neg y_u)$.

Demostració.

Per tant, l'algorisme que donat un circuit C, retorna el conjunt de clàusules F corresponents, és una reducció polinòmica de CIRCUIT-SAT a 3-SAT i aleshores, 3-SAT és **NP**-complet. □

Exemple (2-SAT)

Donat un conjunt de clàusules $\{C_1, \ldots, C_m\}$ de com a molt dos literals cadascuna, determinar si la conjunció $C_1 \wedge \cdots \wedge C_m$ és satisfactible.

Teorema

2-SAT pertany a **P**.

Demostració.

Sigui $F = \{C_1, \ldots, C_m\}$ un conjunt de clàusules de com a molt dos literals cadascuna. Definim un graf dirigit $G = (V, E)$ com segueix.

- $V = \{y : y \text{ apareix a } F\} \cup \{\neg y : y \text{ apareix a } F\}$
- $E = \{(\neg x, y) : (x \vee y) \in F\} \cup \{(\neg y, x) : (x \vee y) \in F\}$

Els camins dins G representen implicacions vàlides. Aleshores, F és insatisfactible si i només si existeix una variable y tal que hi ha camins de y a $\neg y$ i de $\neg y$ a y dins G. □

Exemple (CONJUNT INDEPENDENT)

Donat un graf no dirigit $G = (V, E)$ i un nombre natural k, determinar si existeix un subconjunt $A \subseteq V$ de k vèrtexs tal que $\{u, v\} \notin E$ per a tot $u, v \in A$.

Teorema

CONJUNT INDEPENDENT és **NP***-complet.*

Demostració.

Per reducció de 3-SAT. Sigui $F = \{C_1, \ldots, C_m\}$ un conjunt de clàusules de com a molt tres literals. Definim k i construim un graf no dirigit $G = (V, E)$ com segueix.

- $V = \{(\ell, C_i) : \ell \text{ apareix a } C_i\}$
- $E = \{\{(\ell_1, C_i), (\ell_2, C_j)\} : i = j \vee \ell_1 \equiv \neg\ell_2\}$
- $k = m$

Demostració.

$C_1 \wedge \cdots \wedge C_m$ és satisfactible si i només si G conté un conjunt independent de talla k. De fet, si $C_1 \wedge \cdots \wedge C_m$ és satisfactible, tot conjunt $A \subseteq V$ obtingut sel·leccionant un vèrtex de cada clàusula és de talla k i és independent, perquè les arestes uneixen els literals d'una mateixa clàusula (i només se sel·lecciona un vèrtex de cada clàusula) o bé un literal amb la seva negació (i no es pot haver satisfet tant una literal com la seva negació). Per tant, A és un conjunt independent de talla k. D'altra banda, si $A \subseteq V$ és un conjunt independent de talla k, només pot contenir un vèrtex de cada clàusula i com que hi ha k clàusules, A conté exactament un vèrtex de cada clàusula. A més, $\neg \ell_i \notin A$ per a tot $\ell_i \in A$, perquè A és independent, i l'assignació $T(x_i) = 1$ si $\ell_i \in A$ i $T(\ell_i) = 0$ altrament, satisfà totes les clàusules $C_1, \ldots, C_m$. Per tant, l'algorisme que donat un conjunt de clàusules $\{C_1, \ldots, C_m\}$ de com a molt tres literals, retorna el graf no dirigit G corresponent i el número k és una reducció polinòmica de 3-SAT a CONJUNT INDEPENDENT. Així, CONJUNT INDEPENDENT és **NP**-hard i donat que pertany a **NP**, és **NP**-complet. □

Els problemes marcats amb una estrelleta ($\star$) són més difícils. Si no es diu el contrari i us cal, considereu que tots els logaritmes són en base 2.

- 7.3
- 7.6 (un o dos)
- 7.7 (un o dos)
- 7.9 (en teoria)
- 7.13
- 7.17
- 7.19
- 7.21
- 7.22
- 7.23
- 7.24

Problema (7.3)

Dissenyeu un algorisme que trobi una possible manera de col·locar n reines en un tauler amb $n \times n$ escacs de forma que cap amenaci cap altra o indiqui no hi ha tal col·locació.

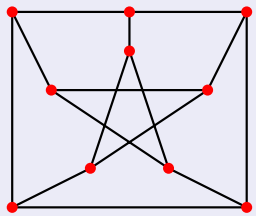
Problema (7.6)

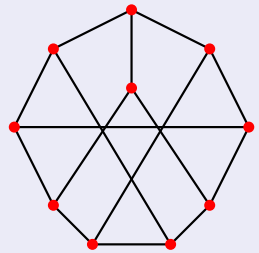
Dissenyeu i analitzeu un algorisme per determinar si un graf connex no dirigit és hamiltonià. Modifiqueu-lo per obtenir-ne un cicle hamiltonià si aquest existeix.

Ja que hi sou, dissenyeu i analitzeu un algorisme per determinar si un graf connex no dirigit és eulerià. Quin és el seu cost? Dissenyeu un algorisme per obtenir un cicle eulerià si aquest existeix.

Problema (7.7)

Dos grafs G_1 i G_2 es diuen isomorfs si existeix una bijecció f entre els vèrtexs d'ambós grafs de forma que $\{u, v\} \in E(G_1)$ si i només si $\{f(u), f(v)\} \in E(G_2)$. Per exemple, els dos grafs següents són isomorfs.





Dissenyeu i analitzeu un algorisme per determinar si dos grafs (donats a través de matrius d'adjacència) són isomorfs.

Problema (7.9)

Es disposa d'una motxilla que pot aguantar fins a C unitats de pes i d'una col·lecció de n objectes, cadascun dels quals té un pes $p[i]$ i un valor $v[i]$. Es vol triar quins objectes cal col·locar a la motxilla, de forma que la suma dels seus valors sigui màxima però que la suma dels seus pesos no sobrepassi C. Els objectes no es poden dividir.

Dissenyeu i analitzeu un algorisme per al problema de la motxilla. Feu-ho amb l'esquema de tornada enrera (backtracking) i amb l'esquema de ramificació i poda (branch and bound).

Problema (7.13)

Dissenyeu i analitzeu un algorisme que, donat un graf $G = (V, E)$ i un nombre natural k, determini si G té un conjunt independent de k vèrtexs (és a dir, si existeix un subconjunt $A \subseteq V$ de k vèrtexs tal que $\{u, v\} \notin E$ per a tot $u, v \in A$).

Problema (7.17)

Es disposa de $c \geqslant 3$ colors diferents i d'un mapa de n països, juntament amb una taula de booleans $v[i][j]$ que indica si dos països i i j són veïns. Escriviu un algorisme de cerca exhaustiva que trobi totes les coloracions possibles del mapa amb la restricció que dos països veïns han de ser pintats amb colors diferents.

Com solucionareu el problema per a $c = 2$ (bicoloració)?

Problema (7.19)

Tenim una col.lecció de n objectes que cal empaquetar en envasos de capacitat C. L'objecte i té volum v[i]. Dissenyeu un algorisme que calculi quin és l'empaquetament òptim, és a dir, aquell que minimitza el nombre d'envasos. Els objectes no es poden fraccionar però, per simplificar el problema, podeu considerar que un objecte cap dins d'un envàs si el volum de l'objecte és igual o inferior a l'espai lliure que queda a l'envàs (independentment de la forma que l'objecte pugui tenir).

Solucioneu aquest problema amb l'esquema de tornada enrera (backtracking) i amb l'esquema de ramificació i poda (branch and bound).

Problema (7.21)

Probablament ja heu perdut moltes hores jugant al Sudoku... Les regles són senzilles: Hom reb una graella 9×9 *amb alguns números i cal que ompli les caselles buides de forma que cada fila, cada columna i cada caixa* 3×3 *contingui tots els números del 1 al 9. Per exemple, el sudoku de l'esquerra té com a única solució la graella de la dreta:*

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

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

Escriviu un programa que resolgui sudokus. L'entrada és una matriu de 9×9 *números entre 0 i 9. Els zeros representen caselles buides.*

Problema (7.22)

Considereu els problemes NP-complets següents:

- *BIN-PACKING: Donades n peces de mides $d_1, \ldots, d_n$ i k contenidors de capacitat m, decidir si es poden col·locar totes les peces en els contenidors quedant tots ells plens.*
- *TRIPARTITE-MATCHING: Donats tres conjunts C_1, C_2, C_3 disjunts amb n elements cadascun d'ells, i un subconjunt $S \subseteq C_1 \times C_2 \times C_3$, decidir si es poden escollir n elements de S de manera que tot element de C_1, C_2 i C_3 aparegui una única vegada entre aquestes n ternes.*
- *3-COLORABILITAT: Donat un graf, decidir si es poden assignar colors als vèrtexs, a escollir d'entre tres colors possibles, de manera que cap aresta tingui els seus extrems del mateix color.*

Cadascuna de les escenes següents presenta un problema. Mostreu que aquest problema és difícil tot veient que la seva versió decisional es pot identificar amb algun del problemes de la llista anterior.

Problema (7.22, cont.)

Considereu els problemes NP-complets següents:

- *GRAF-HAMILTONIÀ: Donat un graf, decidir si hi ha un cicle que passa per tots els seus vèrtexs un sol cop.*
- *CONJUNT-DOMINADOR: Donat un graf i un natural k, decidir si hi ha un subconjunt de k vèrtexs tals que tot altre vèrtex és adjacent a un d'ells.*

Cadascuna de les escenes següents presenta un problema. Mostreu que aquest problema és difícil tot veient que la seva versió decisional es pot identificar amb algun del problemes de la llista anterior.

Problema (7.22, cont; escena 1)

En Roy, a petició d'en Jonny, és l'encarregat d'organitzar un sopar per al grupet de col·legues de sempre. Ha demanat taula en un restaurant; és una taula rodona molt gran i hi caben tots. Quan arriben al lloc, però, apareixen les primeres dificultats.

Jonny: *Escolta Roy, es veu que algunes persones del grup estan enfadades entre elles. Seria convenient que dues persones enfadades no seguessin l'una al costat de l'altra.*

Roy: *Ja comencem... Mira, anem al gra. Vull que recopilis tota la informació al respecte; és a dir, per a cada parell de persones, si estan enfadades o no. Jo miraré com les podem fer seure sense que ningú tingui un veí de taula amb el qual està enfadat.*

Problema (7.22, cont; escena 2)

Tenint totes les dades a la ma, i després d'una bona estona, en Roy no ha trobat encara cap solució. A més, en Jonny torna a aparèixer amb cara de circumstàncies.

Jonny: *Ep, es veu que els que estan enfadats mútuament no volen ni tan sols seure a la mateixa taula. Però tranquil, he preguntat al restaurant i m'han dit que ens deixen tres taules grans.*

Roy: *Vejam, intentaré distribuir a la gent en tres grups de manera que en un mateix grup no hi hagi dues persones mútuament enfadades.*

Problema (7.22, cont; escena 3)

Mentre en Roy pensa com distribuir la gent al voltant de les tres taules, en Jonny apareix de nou informant que el restaurant és a punt de tancar, i que ja poden anar buscant un altre lloc per sopar.

Roy: *Mira, abans de buscar un altre lloc, m'agradaria aclarir quatre coses a la gent: Això de "ai, ai, no vull seure amb en tal o amb en qual" és una criaturada! Vull parlar amb ells seriosament. El problema és que estic una mica afònic i si parlo amb tothom em quedaré sense veu. Vull que escullis k representants de manera que, per a qualsevol dels del grup, hi hagi un dels k representants amb qui no estigui enfadat. D'aquesta manera els k representants faran arribar a tot el grup el missatge.*

Problema (7.22, cont; escena 4)

Com que en Jonny no s'en surt en triar els k representants, finalment agafa un megàfon i comunica a tothom que si continuen amb tanta conya, hauran d'anar cap a caseta.

Jonny: *Escolta, tinc un mapa de la ciutat on he marcat les places on hi ha bons restaurants, i els carrers que les uneixen. Podríem anar passejant per totes i veure si tenen taules lliures.*

Roy: *Ok, deixa'm una estona el mapa per escollir el trajecte, que fa fred i no tinc ganes de passar dues vegades pel mateix lloc.*

Problema (7.22, cont; escena 5)

Malgrat tenir totes les dades a mà, i després d'una bona estona, en Roy no se'n surt. En Jonny el vol ajudar:

Jonny: *Escolta Roy, jo tinc un munt de col·legues a la ciutat i a cadascuna d'aquestes places hi tinc un amic que hi viu. Dona'm pasta que els truco i els demano que mirin si hi ha lloc als restaurants de la seva plaça.*

Roy: *Mira tio, estic escurat i no tinc diners per tantes trucades. Te'n dono prou per fer m trucades. Si de cas, demana als teus amics que mirin no només a la plaça on viuen, sinó també a les places que distin un carrer d'on ells són. Escull als amics adequats perquè puguin mirar totes les places.*

Problema (7.22, cont; escena 6)

En no sortir-se'n, Jonny truca al primer de la llista, i resulta que a prop seu hi ha un restaurant amb taules lliures. Quan hi arriben, resulta que només hi ha una taula amb k cadires i que tanquen el restaurant d'aquí poc.

Jonny: *Ja està, tinc la solució: Anirem sopant primer uns i després els altres, quan algú hagi acabat, sortirà i n'entrarà un altre. Mira, fa un temps vaig dedicar-me a recopilar informació sobre quan trigava cada persona en sopar. Ja saps que tinc hobbies una mica estranys...*

Roy: *Mare meva...*

Jonny: *Amb aquestes dades, podrem fer una planificació distribuint les persones a les cadires, de manera que tothom pugui sopar abans que tanquin.*

Roy: *D'acord, dona'm aquesta llista del temps que triga cadascú a sopar i veuré què hi puc fer. Algun dia m'hauràs d'explicar quines altres dades reculls sobre els demés...*

Problema (7.22, cont; escena 7)

Entretant, en Jonny rep la trucada d'un amic dient que a prop d'allà hi ha un restaurant molt guai. Se n'hi van tots de dret. Quan hi arriben, resulta que només hi ha un munt de tauletes petites, totes de colors diferents.

Jonny: *Escolta Roy. Es veu que la penya estan una mica avorrits i volen aprofitar la situació per asseure's adequadament per parelles. Resulta que som tants nois com noies, i cadascú té les seves preferències. Però les taules de colors també juguen el seu paper. Per exemple, tal persona accepta seure amb tal altra només si la taula és de color rosa, i amb no sé quina altra només només si la taula és de color blau. Haurem de trobar una distribució que satisfaci les restriccions de tothom. Es veu que amb tanta estona sense menjar, la penya està una mica anada.*

Roy: *A la penya els fotré jo una bona pinya. Mare de déu, no estic ara per gaires tonteries, eh! Però vaja, ho intentaré.*

Problema (7.22, cont; escena 8)

No se'n surten i s'ha fet molt tard. En Roy i en Jonny només troben una tasca on hi ha una taula amb una cadira. Això sí, obren un munt d'hores.

Jonny: *Resulta que la tribe estan una mica ratllats. Alguns se'n van a fer un volt i tornaran més tard. Aquí tinc apuntat, per cada persona, a quina hora tornarà i a quina hora se n'haurà d'anar a casa. I encara tinc la llista del temps que necessita cada persona per sopar. Crec que, amb aquestes dades, hauria de ser fàcil veure si podem distribuir a tothom segons els seus horaris perquè puguin anar seient a la cadira i sopant.*

Roy: *Escolta Jonny, n'estic fins als collons. Vull que sàpigues que aquesta és la darrera cosa que faig per aquesta gent. Va, vinga, dona'm aquestes dades i veuré què hi puc fer.*

(En aquesta escena no n'hi ha prou amb identificar un problema de la llista; cal fer una reducció.)

Problema (7.22, cont; escena 9)

Finalment, en Roy i en Jonny decideixen sopar tots dos junts en aquell lloc, només els ha calgut demanar una altra cadira a la cambrera. Als demés, els han donat una adreça d'un lloc on només hi ha un descampat, a la zona amb major índex de delinqüència i criminalitat de la ciutat.

Jonny: *Vinga home, no t'ho prenguis així! Això d'avui simplement ha estat mala sort.*

Roy: *Mira, de mala sort, res. No és el primer cop que em passa això. És que sóc un passarell i sembla que no n'aprendré mai. Ara, vull que escoltis amb atenció. Sé que t'ho he dit altres cops, però aquesta vegada va de debò: No tornaré a organitzar res per aquesta penya mai més. I quan dic mai és mai. De fet, ni tan sols tornaré a quedar amb ells.*

Problema (7.22, cont; escena 9, cont.)

Jonny: *D'acord, d'acord! Llàstima, però. La setmana vinent havíem pensat anar tots a una casa de colònies. Ja saps, natura, aire pur, tranquil·litat, diversió, carn a la brasa, banys al riu tots despullats...*

Roy: *Vaja! Sona bé això! Però no sé, després les coses sempre acaben sortint malament.*

Jonny: *A més, aquest cop vindrà la Steffy. Te'n recordes?*

Roy: *La Steffy?*

Problema (7.22, cont; escena 9, cont.)

Jonny: *Sí, i diu que té moltes ganes de veure't. Crec que s'endurà una desil·lusió molt gran si no vens.*

Roy: *Hmmm... bé... potser n'he fet un gra massa. Si en el fons són bons paios... A vegades estan inaguantables, però són bons paios en definitiva, i això és el que compta. Al cap i a la fi, l'amistat està plena de mals moments, però sobretot de moments meravellosos plens de joia. Va vinga! Què punyetes? M'hi apunto! I si vols t'ajudo a organitzar les coses.*

Jonny: *Perfecte! Ets el millor! Un crack, sí senyor! Va, vinga, posem-nos-hi ara mateix. A veure, l'únic problema que podria haver-hi és que només hi ha k cotxes i...*

Problema (7.23)

Reduïu el problema SUBSET-SUM al problema de la PARTICIÓ: Donat un conjunt d'enters S, trobar si aquest es pot particionar en dos subconjunts S_1 i S_2 tal que $S = S_1 \cup S_2$ i $\sum_{x \in S_1} x = \sum_{x \in S_2} x$.

Demostreu que PARTICIÓ pertany a NP.

Problema (7.24)

Reduïu el problema de CLIQUE al de SUBGRAF-INDUÏT: Donats dos grafs G_1 i G_2, determinar si G_1 és un subgraf induït de G_2.

Demostreu que SUBGRAF-INDUÏT pertany a NP.