ESTRATEGIAS DE PROGRAMACIÓN
(PRIMERA PARTE)

Programación 3

Javier Miranda

Escuela de Ingeniería Informática
Universidad de Las Palmas de Gran Canaria

Estrategias básicas de programación

- Fuerza bruta
- Vuelta atrás (*backtracking*)
- Voráz (*greedy*)

- Divide y vencerás
  - Reduce y vencerás
  - Programación Dinámica

Fuerza Bruta (*Brute Force*)

- Basada en la definición del problema
  - Evalúa todas las posibles combinaciones que resuelven el problema

- Fortalezas:
  - Amplia aplicabilidad
  - Simple
  - Genera soluciones razonables para algunos problemas

- *Debilidades*:
  - Algoritmos poco eficientes
  - En general, con poco esfuerzo podemos hacerlo mucho mejor !

https://en.wikipedia.org/wiki/Brute-force_search

Soporte Python para Fuerza Bruta: iteradores (evaluación perezosa)

- Itertools.permutations()

```
a = [1,2,3]
list(itertools.permutations(a))
# [(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]

list(itertools.permutations(a, 2))
[(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)]
```

https://docs.python.org/3/library/itertools.html

Soporte Python para Fuerza Bruta: iteradores (evaluación perezosa)

- Itertools.combinations()

```
a = [1,2,3,4,5]
b = list(itertools.combinations(a, 3))
```

```
[(1, 2, 3), (1, 2, 4), (1, 2, 5), (1, 3, 4),
 (1, 3, 5), (1, 4, 5), (2, 3, 4), (2, 3, 5),
 (2, 4, 5), (3, 4, 5)]
```

https://docs.python.org/3/library/itertools.html

Python: Generadores (*lazy evaluation*)

```
x = (x for x in range(5))
print(x)                 # <generator object <genexpr> at ….>
print(next(x))           # 0
print(next(x))           # 1
...
x = (x for x in range(5))
print(list(x))           # [0, 1, 2, 3, 4]
```

Para depurar y ver la lista completa

```
def f(x): return 2*x
def g(x): return 3*x

f_x = list(f(x) for x in range(5))          # [0, 2, 4, 6, 8]
g_f_x = list(g(f(x)) for x in range(5))     # [0, 6, 12, 18, 24]
```

Python: Generadores (*lazy evaluation*)

```
x = (x for x in range(5))
print(x)                    # <generator object <genexpr> at ….>
print(next(x))              # 0
print(next(x))              # 1
...
print ([x for x in range(5)])                # [0, 1, 2, 3, 4]
```

Sintaxis equivalente: [ ]

```
def f(x): return 2*x
def g(x): return 3*x

f_x = [f(x) for x in range(5)]          # [0, 2, 4, 6, 8]
g_f_x = [g(f(x)) for x in range(5)]     # [0, 6, 12, 18, 24]
```

Sintaxis equivalente: [ ]

Python: Escribiendo funciones generadoras

```
def numbers(n):
    for j in range(n):
        yield j
```

Función generadora

```
g = numbers(2)
print(g)            # <generator object numbers at 0x….>

print(next(g))      # 0
print(next(g))      # 1

print(list(numbers(5)))          # [0, 1, 2, 3, 4]
a, b, c, d = numbers(4)          # a=0 b=1 c=2 d=3
```

Vuelta atrás (*Backtracking*)

- Estrategia de búsqueda que descarta (*poda del árbol*) las combinaciones que no llevan a la solución.

Algoritmo Genérico

1. Generamos una combinación componente a componente
2. Evaluamos si nos puede llevar hacia la solución
3. Si no satisface alguna restricción del problema descartamos esta solución (y todas las que dependan de ella)
4. Volvemos al paso 1

Implementación Iterativa

Ejemplo

- Encontrar todas las permutaciones de 3 variables binarias (x_1, x_2, x_3) en las que no haya valores iguales consecutivos.
  - Ejemplo de solución: 1-0-1

Ejemplo: Solución por fuerza bruta

- Utilizando un iterador de combinaciones

$x_1\ x_2\ x_3$

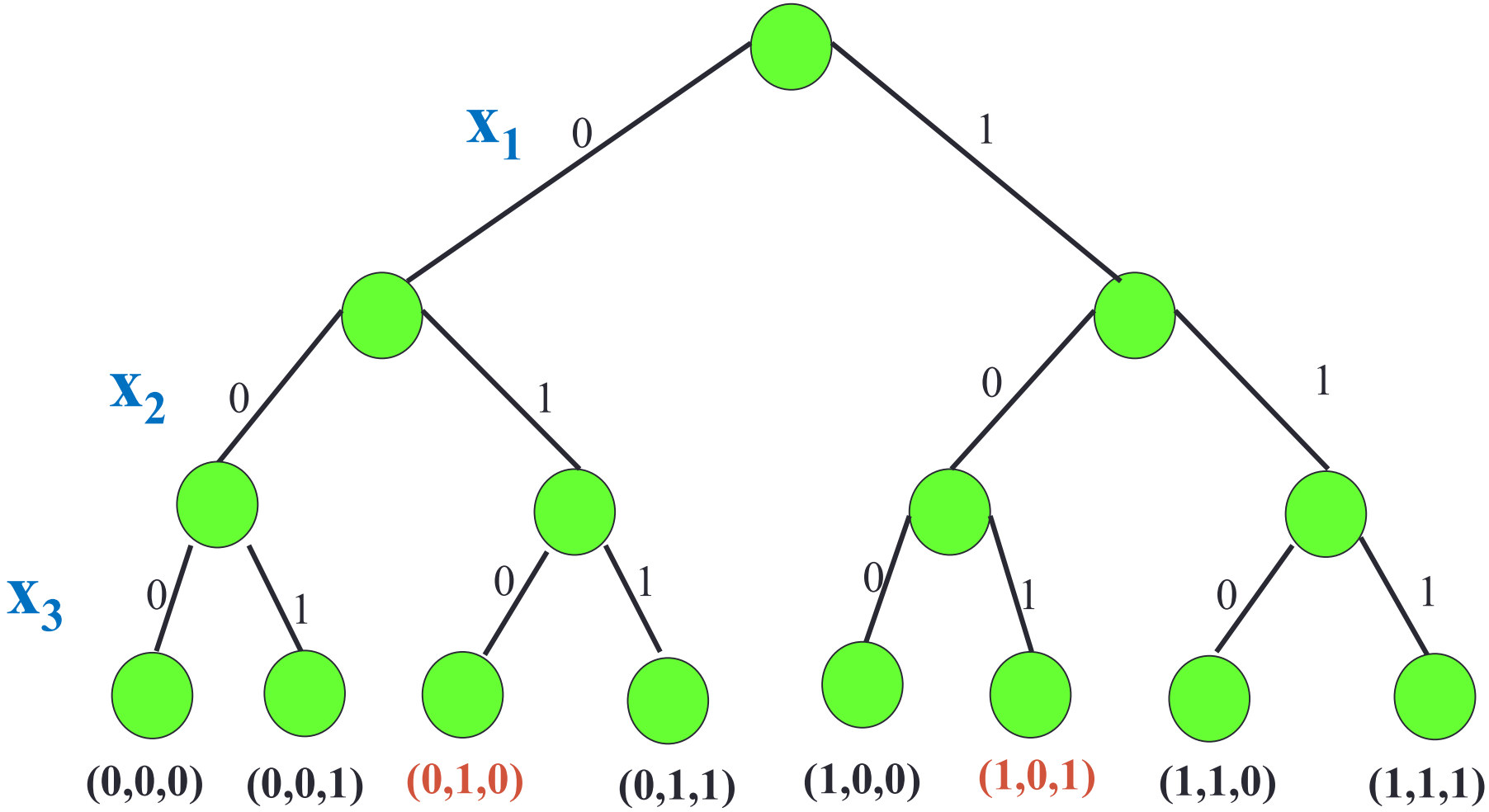
0-0-0
0-0-1
0-1-0
0-1-1
1-0-0
1-0-1
1-1-0
1-1-1

Coste de ejecución = $2^3 = 8$

Veamos gráficamente estas combinaciones

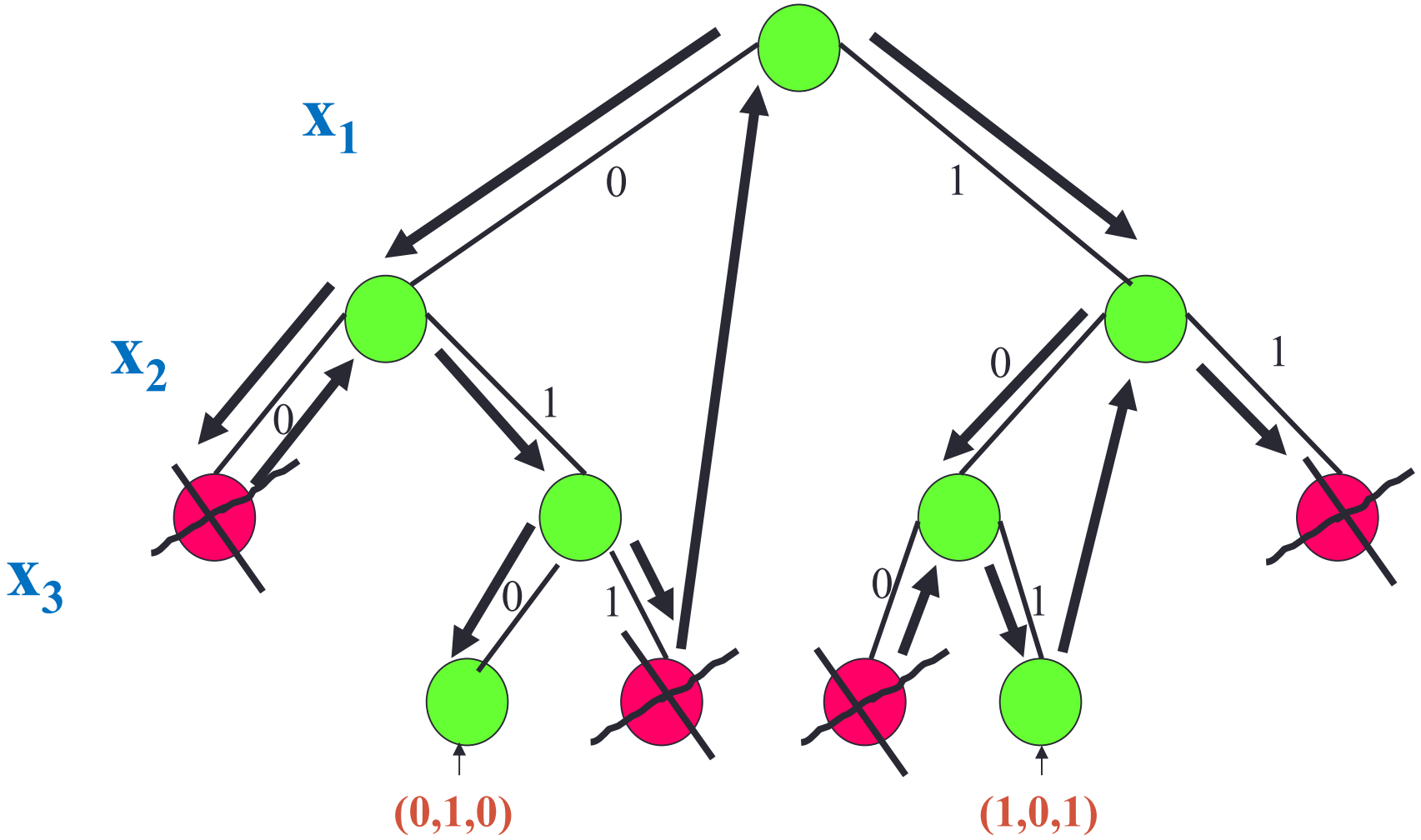
Ejemplo: Solución por fuerza bruta

Asociando a cada variable a un nivel de profundidad (x_1, x_2, x_3) obtenemos el árbol de combinaciones.



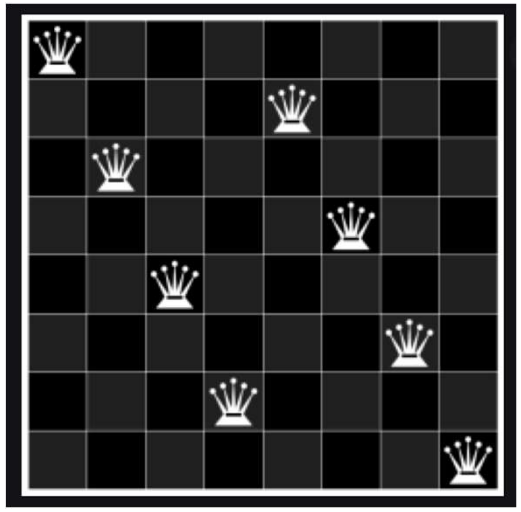
Ejemplo: Solución por Backtracking

- Con backtracking sólo creamos combinaciones válidas

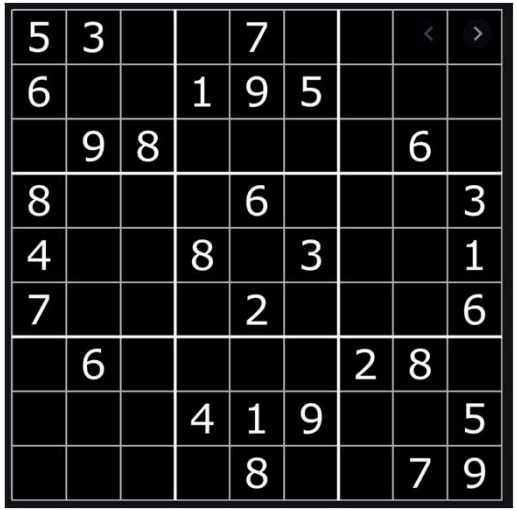


Vuelta atrás (*Backtracking*)

Ejemplos de uso:



N-Queens



Sudoku

- Es básicamente una estrategia de fuerza bruta con poda
- Puede implementarse recursivamente o iterativamente

https://en.wikipedia.org/wiki/Backtracking

Ejemplo: Sudoku con Backtracking

```
def solveSudoku(grid, i=0, j=0):
        i,j = findNextCellToFill(grid, i, j)
        if i == -1:
                return True
        for e in range(1,10):
                if isValid(grid,i,j,e):
                        grid[i][j] = e
                        if solveSudoku(grid, i, j):
                                return True
                        # Undo the current cell for backtracking
                        grid[i][j] = 0
        return False
```