INF221 – Algoritmos y Complejidad

Clase #14
Backtracking

Aldo Berrios Valenzuela

Miércoles 21 de septiembre de 2016

1. Backtracking

Otra estrategia recursiva... La idea es ir construyendo la solución incrementalmente, explorando distintas ramas y volviendo atrás (backtrack) si resulta que un camino es sin salida.

Ejemplo 1.1 (Un clásico)**.** 8 reinas.

/* Dibujo */

Pasos:

- Reducir espacio de búsqueda.

  $\rightsquigarrow$ Si son 8 reinas, hay una por columna.

  Por lo tanto, llenar por columnas, solución (parcial) indica las filas de las reinas ya ubicadas.

- Ordenar avance.
- Subproblemas similares.

En este caso:

- Ubicar reina en columna 1, 2, ...
- Registrar filas libres (por omitir ocupadas al ubicar la siguiente reina)
- Registrar diagonales libres.
  Reina en i, j:

  $$y - j = 1 \cdot (x - i)$$
  $$i - j = x - y \rightsquigarrow x - ycte$$
  $$y - j = -1 \cdot (x - i) \rightsquigarrow x + ycte$$

  Elegimos Python (!), en Python los arreglos tienen índices desde 0, rango de i, h es $0 \ldots 7$. ¿Rangos para?:

- $i - j$: $-7 \ldots 7 \rightsquigarrow$ sumar 7 para llevar al rango $0 \ldots 14$.
- $i + j$: $0 \ldots 14$

Por lo tanto, la estructura registra una solución parcial es:

```
/* Pero que clase de python es este? */
int queen[8];       /* Fila de reinas */
int n;              /* N de reinas ubicadas */
bool rfree[8];      /* Filas libres */
bool dufree[14], ddfree[14];
```

```
/* Colocar un dibujo con algunas reinas en el tablero para mostrar más o menos cómo funciona
el algoritmo. */
```

Ahora si en Python:

```
def solve(n):
  if n == 8:
    # Escribir solucion, ...
    return
  else:
    for k in range(8):
      if rfree[k] and ddfree[n+k] and dufree[n+7-k]:
        rfree[k] = ddfree[n+k] = dufree[n+7-k]  = False
        queen[n] = k
        solve(n+1)
        rfree[k] = ddfree[n+k] = dufree[n+7-k] = True
```

■

```
/* Sudoku es otro ejemplo de un problema de Backtracking */
```