Vuelta Atrás (Backtracking)

Vuelta Atrás (Backtracking)
Objetivos

- Comprender la finalidad de backtracking.
- Comprender el concepto de backtracking.
- Comprender el esquema de backtracking.

Vuelta Atrás (Backtracking)
Comprender la finalidad de backtracking

Backtracking es una técnica aplicada a problemas que cumplan estas condiciones:

- Que la solución se pueda expresar mediante una tupla (x1,x2,x3,...,xn).
- Que el problema se formule o reformule como la búsqueda de aquella tupla que maximiza un determinado criterio.

Backtracking realiza una búsqueda sistemática de la solución óptima la problema, sea cual sea el problema.

Vuelta Atrás (Backtracking)
Comprender el concepto de backtracking

Para ello, backtracking aborda el problema de forma **recursiva**:

- Se prueban todas las combinaciones del primer valor de la tupla.
  - Para cada una de ellas, se prueban todas las combinaciones del segundo valor de la tupla.
    - Para cada una de ellas, se prueban todas las combinaciones del tercer valor de la tupla.
      - ...

Luego se mejora el tiempo de exploración aplicando **podas**:

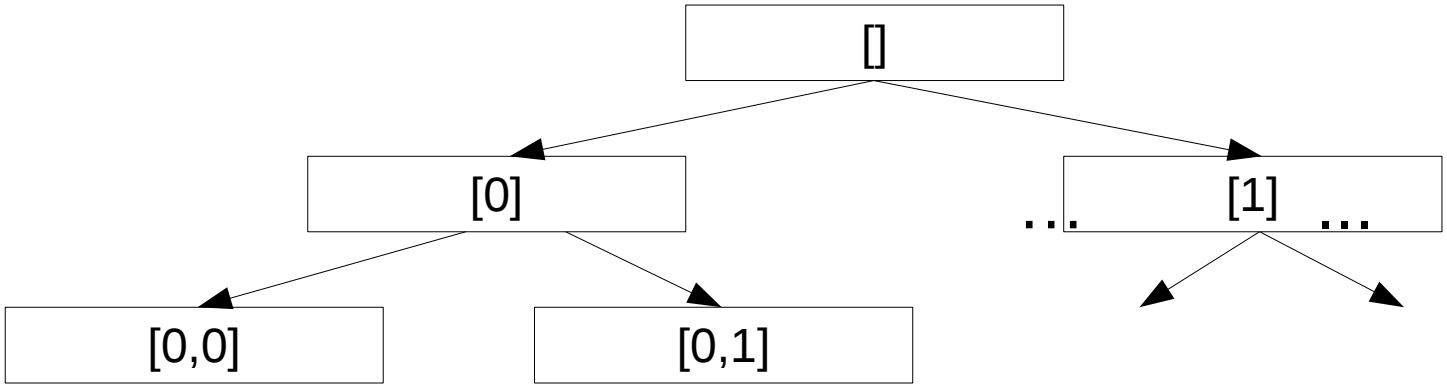
- Si solución encontrada no es factible.
- Si sabemos que la solución encontrada no deja alcanzar la óptima.
- Si localizamos ciclos (no está acotado el tamaño de la tupla).

Vuelta Atrás (Backtracking)
Comprender el concepto de backtracking

Se impone una estructura de árbol sobre el conjunto de posibles soluciones (espacio de soluciones).

Se generan las soluciones en forma de recorrido preorden del árbol.

Se procesan las hojas (soluciones completas).



Vuelta Atrás (Backtracking)
Comprender el concepto de backtracking

Se definen las restricciones implícitas como las restricciones que se nos dan en el enunciado del problema y que debemos comprobar para saber si una solución es factible.

Se definen las restricciones explícitas como aquellas que están limitadas por el funcionamiento del algoritmo (por ejemplo un bucle de 0 hasta n no va a valer más allá de n) y que no debemos comprobar para saber si una solución es factible.

Vuelta Atrás (Backtracking)
Comprender el esquema de backtracking

```
int mejor[]; //Contiene la mejor tupla hasta el momento.
int costemejor; //Contiene el mejor coste hasta el momento

backtracking(int v[],int pos) {
  // v = vector que contiene la tupla.
  // pos = posición sobre la que debemos trabajar ahora.
    //Las de antes están rellenas, las de después faltan.
  if (solucion(v,pos)) {
    coste = calculamos_coste(v,pos);
    if (coste < costemejor) {
      copiamos a “mejor” el contenido de “v”.
      costemejor = coste;
    }
  }
  else {
    for (i = 0;i < x;i++) {
      v[pos] = i;
      backtracking(v,pos+1);
    }
  }
}
```

Vuelta Atrás (Backtracking)
Comprender el esquema de backtracking

```
TDA solucion { coste, tupla[], pos }
  // coste = coste de la solución actual (se calcula al final)
  // tupla = vector que contiene la tupla.
  // pos = posición sobre la que debemos trabajar ahora.
    //Las de antes están rellenas, las de después faltan.
solucion mejor;

solucion backtracking(solucion a) {
  if (a.esCompleta()) //El coste con a.calcularCoste();
    return a;
  else {
    solucion mejor;
    int pos = a.getPosicionSiguiente();
    a.pos++; //Incremento la posición.
    for (i = 0;i < x;i++) { //Valores posibles.
      a.Poner(pos,i);
      actual = backtracking(a);
      if (actual.calcularCoste() < mejor.calcularCoste())
        mejor = actual;
    }
    return mejor;
  }
}
```

Vuelta Atrás (Backtracking)
¿Qué hemos aprendido?

- **Finalidad de backtracking.**
- **Concepto de backtracking.**
- **Esquema de backtracking.**

Vuelta Atrás (Backtracking)
Ejemplos

Ejemplo: Viajante de comercio

Objetivo: Escoger un ciclo que recorra una sola vez cada ciudad y cuya sumatoria del coste sea la menor posible.