

Backtracking

- *Backtracking* (o búsqueda atrás) es una técnica de programación para hacer búsqueda sistemática a través de todas las configuraciones posibles dentro de un espacio de búsqueda.
- Para lograr esto, los algoritmos de tipo *backtracking* construyen posibles soluciones candidatas de manera sistemática. En general, dado una solución candidata s :
 1. Verifican si s es solución. Si lo es, hacen algo con ella (depende del problema).
 2. Construyen todas las posibles extensiones de s , e invocan recursivamente al algoritmo con todas ellas.
- A veces los algoritmos de tipo *backtracking* se usan para encontrar una solución, pero otras veces interesa que las revisen todas (por ejemplo, para encontrar la más corta).

Suposiciones sobre el espacio de soluciones

- Supondremos que una solución se puede modelar como un vector $a = (a_1, a_2, \dots, a_n)$, donde cada elemento a_i está tomado de un conjunto ordenado finito S_i .
- Representamos a una solución candidata como un vector $a = (a_1, \dots, a_k)$.
- Las soluciones candidatas se extenderán agregando un elemento al final.

Algoritmo Genérico de *Backtracking*

- El siguiente es un algoritmo genérico de *backtracking*:

BT(A, k)

```
1  if SOLUCION?( $A, k$ )
2      then PROCESAR_SOLUCION( $A, k$ )
3      else for each  $c \in \text{SUCESORES}(A, k)$ 
4          do  $A[k] = c$ 
5              BT( $A, k + 1$ )
6              if terminar?
7                  then return
```

donde

- SOLUCION?(\cdot) es una función que retorna verdadero ssi su argumento es una solución.
- PROCESAR_SOLUCION(\cdot), depende del problema y que maneja una solución.
- SUCESORES(\cdot) es una función que dado un candidato, genera todos los candidatos que son extensiones de éste.
- *terminar?* es una variable global booleana inicialmente es falsa, pero que puede ser hecha verdadera por PROCESAR_SOLUCION, en caso que sólo interesa encontrar

una solución.

Un ejemplo práctico

- Supongamos que queremos un algoritmo para encontrar todos los subconjuntos de n elementos de un conjunto de m elementos.
- Supongamos, además, que los conjuntos están implementados en un arreglo y que la variable $s[]$ contiene al conjunto.

- ¿Qué es un candidato?

Como un candidato es una solución parcial, y representa a un conjunto que tiene a lo más n elementos.

El candidato se representa por un vector binario (a_1, \dots, a_k) donde $a_i = 1$ ssi el i -ésimo elemento de s está en el subconjunto representado.

- ¿Cuáles son las formas de extender un candidato (a_1, \dots, a_k) ?

Agregando un 0 o un 1 al final de éste.

- ¿Cuándo un candidato es una solución?

Si $\sum_{i=0}^k a_k = n$ y $k = m$

- Implementación: [demo]

Modificaciones menores

- ¿Y si ahora queremos mostrar todos los conjuntos?

Sólo tenemos que cambiar la función `solucion` para que retorne verdadero cuando el vector es una solución completa.

Implementación: demo.

El problema del vendedor viajero

- El problema del vendedor viajero consiste en que tenemos un grafo no dirigido en el cual los arcos tienen costos asociados, y queremos encontrar un camino cerrado que recorra a todos los nodos del grafo, con costo mínimo.
- Para resolver este problema suponemos que el grafo está representado por una matriz de costos.
- Implementacion: [demo].