# Tecniche Algoritmiche II: Paradigmi e Tecnica di Progetto, Prima Parte

## I. Tecniche Algoritmiche e Paradigmi

Le **tecniche algoritmiche** sono basate su diversi paradigmi di utilizzo dello spazio di ricerca.

Consideriamo:

- la **tecnica di enumerazione**;
- la **tecnica di backtracking**;
- la **tecnica golosa (greedy)**;
- la **tecnica divide et impera**.

I **paradigmi** sui quali si basano queste tecniche sono:

- **Paradigma Selettivo**
  Tecniche che, per ogni istanza del problema, visitano lo spazio di ricerca tentando di trovare un elemento ammissibile.
  Ogni algoritmo considera l'intero spazio di ricerca che viene esplorato sistematicamente e con una modalità ben definita.

  A questo paradigma appartengono la **tecnica enumerativa** e di **backtracking**.

- **Paradigma Generativo**
  Tecniche che generano una soluzione senza selezionarla tra gli elementi dello spazio di ricerca.
  In queste tecniche lo spazio di ricerca viene considerato solo in fase di progettazione dell'algoritmo per caratterizzare le soluzioni.

  Appartengono a questo paradigma la **tecnica golosa** e la **divide-et-impera**.

## II. Paradigma Selettivo

Le tecniche algoritmiche si distinguono in base a come utilizzano lo **spazio di ricerca**.
Le tecniche che appartengono al paradigma selettivo cercano un elemento ammissibile tra quelli nello spazio di ricerca.

Le varie tecniche si distinguono per la **modalità** in cui esplorano lo spazio di ricerca.

> 📎 **Esempio:** Ordinamento di un vettore di interi in ordine decrescente
> Lo spazio di ricerca sarà costituito da tutte le permutazioni (combinazioni in ordine diverso) del vettore.
> Un algoritmo selettivo visita lo spazio di ricerca e controlla se c'è una permutazione che risolve l'istanza del problema.
>
> Il **processo di esplorazione** dello spazio di ricerca può essere rappresentato mediante **albero**, dove ognuna delle foglie contiene un possibile ordinamento (considerando l'esempio del vettore).
>
> La **profondità** dell'albero rappresenta il numero di confronti necessario nel **caso peggiore**.

### 2.1 - Tecnica Enumerativa

La **tecnica enumerativa** ispeziona ogni elemento dello spazio di ricerca e garantisce la **terminazione** se lo spazio di ricerca è finito.

Proprio per quanto riguarda la **terminazione**, in base al tipo di problema abbiamo:

- **Problema di Ricerca**: si termina quando si è trovato un elemento **ammissibile** oppure quando lo **spazio di ricerca è esaurito**.
- **Problema di Ottimizzazione**: si confrontano tutti gli elementi per selezionare quelli ammissibili e si termina quando lo **spazio di ricerca è esaurito**.

Per garantire una visita sistematica si associa allo spazio di ricerca una **relazione di ordinamento totale** in modo da poter definire:

- un metodo per stabilire il primo elemento da considerare;
- un metodo per stabilire l'elemento successivo;

- un metodo per verificare se sono stati esaminati tutti gli elementi.

### 2.1.1 - Algoritmo Enumerativo per Problemi di Ricerca

```
considera il primo elemento x dello spazio di ricerca

se a(x) == true
  allora fornisci O(x) come risultato
se tutti gli elementi dello spazio di ricerca sono stati esaminati
  allora fornisci ⊥ come risultato
altrimenti
  considera come nuovo x l'elemento successivo dello spazio di ricerca e ripeti
```

### 2.1.1 - Algoritmo Enumerativo per Problemi di Enumerazione

```
considera il primo elemento x dello spazio di ricerca

if (a(x) == true && soluzioniAmmissibili == 0) || if (migliore(x, soluzioneOttimaCorrente) == true)
  soluzioneOttimaCorrente = x

if (spazioDiRicercaEsaminato == true)
  if (soluzioni ammissibili == 0)
    return ⊥
  else if (soluzioni ammissibili > 0)
    return soluzioneOttimaCorrente
else considera x come elemento successivo dello spazio di ricerca e ripeti
```

## 2.2. Backtracking

Nel **backtracking** la generazione di elementi dello spazio di ricerca, che devono essere ispezionati, avviene tramite un processo suddiviso in **stadi**.

In questo processo, ogni elemento dello spazio di ricerca è costituito da componenti.
Ad ogni stadio viene scelta una componente diversa.

Ogni elemento, quindi, è strutturato in $n$ componenti.
Dopo $i$ stadi $(i < n)$, si è costruita una **soluzione parziale**: se questa soluzione parziale è fallimentare l'algoritmo si interrompe e tenta altre vie.

Quest'ultimo passaggio è ciò che caratterizza la tecnica del backtracking.

### 2.2.1: Backtracking: Albero di Ricerca

Sia $P$ un **problema** e sia dato un metodo per associare ogni **istanza** $i$ del problema ad uno **spazio di ricerca** $Z_i$.
Sia inoltre definito un metodo per strutturare ogni elemento $Z_i$ in un numero finito di componenti.

L'**albero di ricerca** associato ad un istanza $i$ tramite $Z_i$ è un albero nel quale:

- La **radice** (livello 0) rappresenta una soluzione parziale fittizia (vuota)
- Ogni **nodo** rappresenta una **soluzione parziale** alla quale sono stati già aggiunti $j$ componenti.
  Si potrà aggiungere la prossima componente (cioè la *j+1-esima*) in tanti modi quanti sono i figli del nodo in questione.
- Ogni **foglia** è una **soluzione ammissibile**, quindi un elemento $Z_i$ dello spazio di ricerca $Z$.

Inoltre è necessario definire:

- un **metodo** per **rappresentare** ogni s**oluzione parziale** (nodo dell'albero);
- un **metodo** per **stabilire** se una soluzione **viola i vincoli**;
- una **funzione di ammissibilità** che restituisce $\mathrm{true}$ se una **soluzione parziale viola i vincoli** o se l'**elemento dello spazio di ricerca** corrispondente al nodo dell'albero **non è ammissibile**.