Tecniche Algoritmiche III

I. Paradigma Generativo

Le tecniche che rientrano nel paradigma generativo generano soluzioni senza selezionarla dallo spazio di ricerca.

II. Tecnica Golosa (Greedy)

La **tecnica greedy** si applica principalmente a problemi di ottimizzazione.

L'algoritmo costruisce un elemento dello spazio di ricerca in **stadi**.

Ad ogni stadio i, per la componente i-esima viene scelto il valore migliore tra quelli ammissibili secondo un determinato criterio.
Una volta effettuata la scelta, l'algoritmo non torna più sulla decisione presa.

In sostanza, l'algoritmo prende la decisione che al momento sembra migliore, ma non garantisce sempre una soluzione ottimale.

2.1 - Algoritmo Greedy

Si presuppone che l'algoritmo:

- acquisisca la rappresentazione di una istanza del problema;
- disponga di un metodo per organizzare in stadi la costruzione di un elemento dello spazio di ricerca.

L'algoritmo è:

1. Poni $i = 1$ ed inizializza z
2. Determina l'insieme A dei valori ammissibili per la componente i-esima di z.
   Se $A \neq \emptyset$ scegli il migliore in A in base al criterio di preferenza fissato.
3. Se l'i-esimo stadio è l'ultimo, allora termina e restituisce $o(z)$ come risultato.
4. Altrimenti incrementa i di 1 e torna al punto 2.

2.2 - Tecnica Greedy per Problemi di Ottimizzazione

Per utilizzare la **tecnica greedy in problemi di ottimizzazione** devono essere verificate due proprietà:

- **Scelta Greedy**
  Questa proprietà assicura che si può ottenere una **soluzione ottima globale** prendendo decisioni che sono **ottimi locali**.
- **Sottostruttura Ottima**
  Questa proprietà è verificata se una soluzione ottima di un problema contiene al suo interno una soluzione ottima dei sotto-problemi.

La tecnica greedy risulta utile per problemi di scheduling, in cui i task devono essere eseguiti in ordine ottimale in base ad un criterio.

III. Tecnica Divide Et Impera

La tecnica **divide et impera** consiste nel dividere il problema in sottoproblemi più piccoli dello stesso tipo, risolverli, e successivamente ricombinare le soluzioni parziali per ottenere la soluzione del problema principale.

Il principio per il quale si può dividere il problema è il **principio di decomposizione induttiva**.

Solitamente gli algoritmi che utilizzano questa tecnica sono più **efficienti** di quelli del paradigma generativo.
Nello specifico, l'**efficienza** dipende dal numero di sottoproblemi generati e dal costo necessario a scomporre e ricomporre il problema.

Per applicare questa tecnica è **necessario**:

- una **relazione d'ordinamento totale** sulle istanze del problema, che le ordina in base alla dimensione dell'input;
- un **metodo di risoluzione diretto** per tutte le istanze del problema che non superano una certa dimensione;
- un **meccanismo per suddividere i dati in ingresso** per ogni istanza del problema in diverse parti più piccole, per poi rappresentale l'input di una nuova istanza dello stesso problema;
- un **meccanismo per ricomporre le soluzioni** parziali nella soluzione originale.

3.1 - Algoritmo Divide Et Impera

📌
1. Se l'input ha dimensione inferiore ad un certo valore k, risolvi l'istanza del problema direttamente
2. Altrimenti, dividi l'input in parti inferiori all'input originario (**divide**)
3. Esegui ricorsivamente l'algoritmo su ciascuno degli input individuati al passo precedente
4. Ricomponi le soluzioni parziali per ottenere la soluzione al problema originario (**impera**).

Solitamente questa tecnica si applica negli algoritmi di ordinamento (natural merge-sort e quicksort).