

# Industry.cpp

In questo laboratorio era richiesto progettare e implementare una struttura dati a supporto della gestione dei processi produttivi di un'industria semplificata.

Tale industria è in grado di fabbricare oggetti, detti *item*, i quali possono essere di due tipi, *Basic*, oggetti elementari disponibili senza dipendenze, oppure *Composti*, oggetti che possono essere prodotti solo a partire da uno o più altri *item*.

Ho deciso di progettare la struttura nel seguente modo:

- **Array Dinamico Ordinato** con *size* e *maxSize* per la gestione degli *item Basic*:

Questa struttura dati risulta comoda quando dobbiamo effettuare molteplici ricerche, in particolare utilizzando sempre un algoritmo di *ricerca binaria* riusciamo a ridurre la complessità totale dell'operazione a  $\theta(\log(n))$ , dove  $n$  è il numero di *item* attualmente presenti.

Le operazioni di Inserimento e Cancellazione risulteranno più complesse e costose in termini di efficienza, nel caso peggiore  $\theta(n)$ , poiché richiedono uno shift degli elementi.

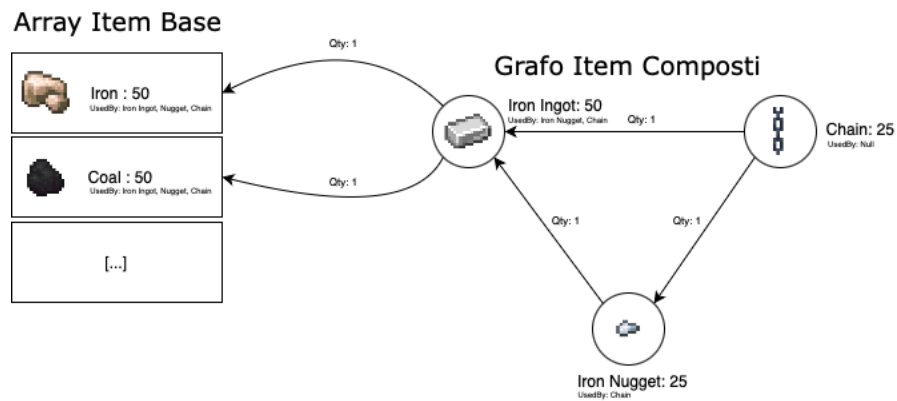
Tuttavia, ho valutato che nella pratica queste operazioni sono rare, poiché una azienda modifica di raro la propria lista di materie prime. Ciascun elemento dell'*Array* ha un proprio campo *Label* per contenere il proprio nome, una *Quantità* finita non negativa, un campo *booleano Visited*, comodo per visite *dfs* e una *lista doppiamente collegata* che ha il compito di memorizzare tutti gli *item composti* di cui l'elemento corrente è *componente*.

- **Grafo Orientato Aciclico** implementato con **triple liste di Adiacenza** per immagazzinare gli *item composti*. Ciascun *Vertice* del *Grafo* rappresenta un *item composto*, ciascun *Vertice* infatti possiede due *liste semplici* di *archi pesati*, che contengono le dipendenze suddivise in *item di base* e *item composti* e una *lista doppiamente collegata* che, come nel caso degli *item di base*, contiene i puntatori agli *item composti* di cui è componente.

Tale rappresentazione risulta particolarmente efficace per supportare in modo efficiente le funzioni *listNeededBy* e *listNeededByChain*.

Per gestire l'ordinamento delle liste di *output* ho riutilizzato il codice dell'algoritmo *MergeSort* prodotto in un primo laboratorio.

Rappresentazione Grafica della Struttura Dati tramite un esempio:



Possiamo notare come ciascun *arco* punti all'*item* richiesto senza generare *cicli*. Un esempio simile è stato utilizzato nel *main* di testing provvisorio.