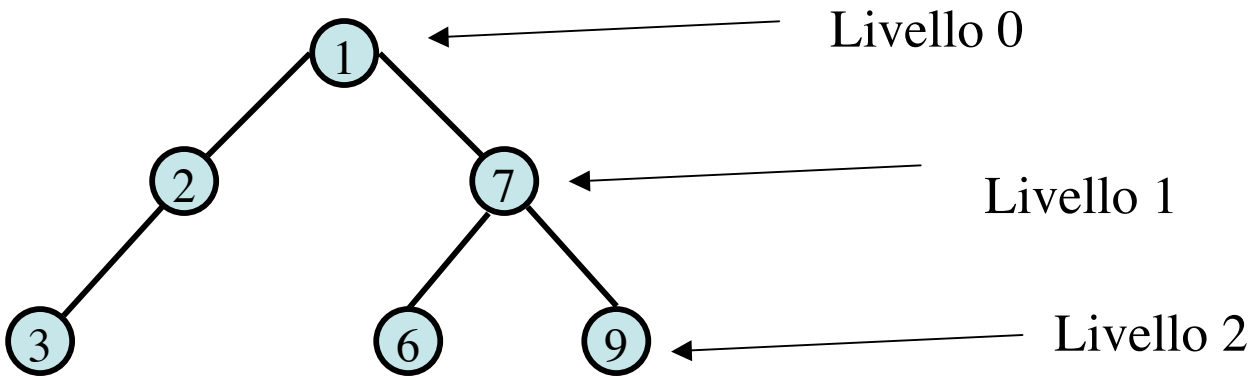
visita per livelli: definizione



Si vuole visitare i nodi in ordine da sinistra a destra e in ordine discendente di livello.
Nel caso li si voglia semplicemente visualizzare a monitor potremmo avere, nel caso dell'esempio: 1 27 369

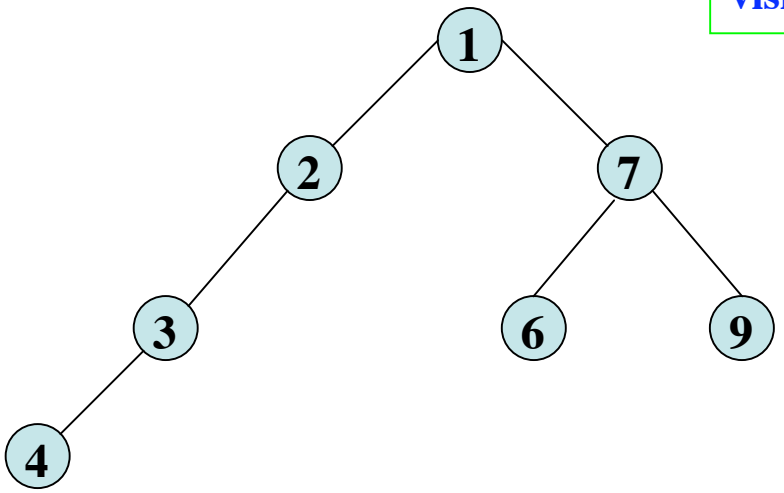
Non ricorsivo!! Un algoritmo iterativo userà una coda per memorizzare i nodi mano mano che si scende di livello.

visita per livelli: l'algoritmo

1. **accoda il nodo radice, se l'albero è non vuoto**
2. **Finché la coda non è vuota:**
   estrai un nodo dalla coda
   visita il nodo
   se il puntatore al figlio sinistro non è NULL
   inserisci il figlio sinistro nella coda
   se il puntatore al figlio destro non è NULL
   inserisci il figlio destro nella coda

La coda per tutta la durata del ciclo conterrà, ordinati da sinistra verso destra, i nodi di un livello oppure i nodi più a destra del livello liv seguiti dai nodi del livello liv +1, figli (se esistenti) dei nodi del livello liv che precedono quelli nella coda e che non sono più nella coda.

visita per livelli: esempio di esecuzione dell'algoritmo



Contenuto coda		
1		Nodo estratto e visitato: 1
2	7	Nodo estratto e visitato: 2
7	3	Nodo estratto e visitato: 7
3	6 9	Nodo estratto e visitato: 3
6	9 4	Nodo estratto e visitato: 6
9	4	Nodo estratto e visitato: 9

visita per livelli: la funzione

```
void levelOrder(TreePtr tPtr)
/* postc: visita i nodi di un albero binario qualunque  per livelli, usando una coda di
puntatori di tipo TreePtr */
{CodaP coda ;
 TreePtr tempPtr;
 costrCoda(numNodi(tPtr));
 if (treePtr !=  NULL)
   {accoda(tPtr,coda);
      while (!(vuota(coda)))
         {tempPtr = estrae(coda);
          visit(tempPtr);
           if  (tempPtr -> lPtr != NULL  && !piena(coda))   accoda(tempPtr->lPtr,coda);
           if  (tempPtr -> rPtr != NULL  && !piena(coda))  accoda(tempPtr->rPtr,coda);
         }
   }
distrCoda(coda);
}
```

visita per livelli: schema utilizzo file