Eliminiamo la ricorsione

- Nei metodi visti per le visite Preorder, Inorder e Postorder, vi è una una doppia chiamata ricorsiva
- E' possibile? E' sempre possibile: il sistema esegue il codice ricorsivo in maniera iterativa
- A che scopo? Per migliorare l'efficienza dell'implementazione

Implementazione iterativa per Preorder

```
protected void iterPreorder()
{       IntBTNodo p = root;
        Pila aiuto = new Pila();
        if (p != null)
        {       aiuto.push(p);
                while (!aiuto.isEmpty())
                {       p = (IntBTNodo) aiuto.pop();
                        p.visit();    // da gestire !!
                        if (p.right != null) aiuto.push(p.right);
                        if (p.left != null) aiuto.push(p.left);
                }
        }
}
```

Discussione

- Considerando i metodi per gestire la pila, il codice totale è sensibilmente aumentato
- Codice ricorsivo più intuitivo
- L'ordine delle invocazioni di `visit` e `push` è ininfluente
- E' cruciale che l'invocazione di pop preceda le altre invocazioni

Si simuli `iterPreorder` su un BT a piacere

Implementazione iterativa per Inorder

```
protected void iterInorder()
{       IntBTNodo p = root;
        Pila aiuto = new Pila();
        while (p != null)
        {       while (p != null)
                {       // impila figlio dx se esiste, e il nodo
                        // stesso procedendo verso sx
                        if (p.right != null) aiuto.push(p.right);
                        aiuto.push(p);
                        p = p.left;
                }
                // estrai un nodo senza figlio sinistro
                p = (IntBTNodo) aiuto.pop();
                ...
```

Implementazione iterativa per Inorder

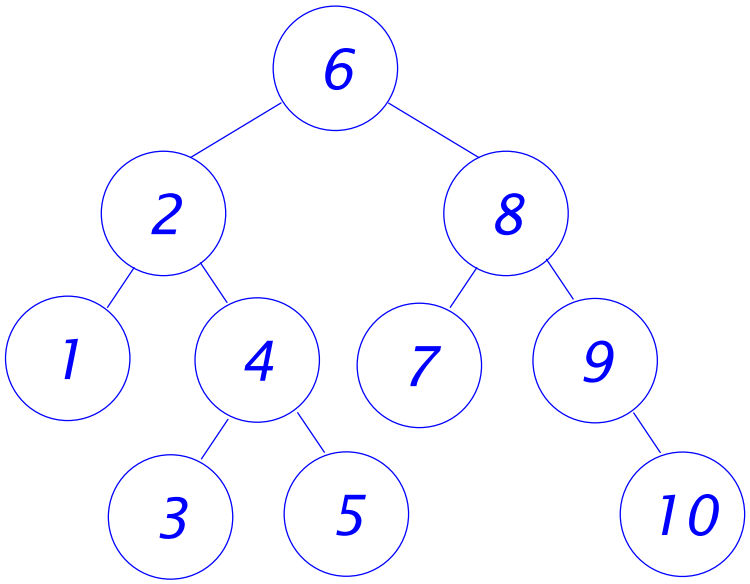
```
            while (!aiuto.isEmpty() && p.right == null)
            {   //visita nodo e tutti quelli senza figlio dx
                   p.visit();          // da gestire !!
                   p = (IntBTNodo) aiuto.pop();
            }

            // visita anche il primo nodo con un figlio dx
            p.visit();
            if (!aiuto.isEmpty())
                   p = (IntBTNodo) aiuto.pop();
            else p = null;

       } // fine while più esterno
} // fine metodo
```

Discussione

- Comprendere il funzionamento del metodo è abbastanza complicato
- Mentre l'equivalente metodo ricorsivo era molto leggibile

Si simuli `iterInorder` su un BT a piacere

Esempio



Implementazione iterativa per Postorder

```
protected void iterPostorder()
{  IntBSTNodo p = root, q = root;
   Pila aiuto = new Pila();
   while (p != null)
   {  for ( ; p.left != null; p = p.left) aiuto.push(p);
      while (p != null && (p.right == null || p.right == q))
      {  p.visit();        // da gestire !!
         q = p;
         if (aiuto.isEmpty()) return;
         p = (IntBSTNodo) aiuto.pop();
      }
      aiuto.push(p);
      p = p.right;
   }
}
```

Discussione

- Ciascun nodo con due figli viene inserito due volte sulla pila, una volta prima di attraversarne il sotto-albero sinistro, l'altra prima di attraversarne il destro
- Ciascun nodo con un solo figlio viene inserito una volta sulla pila
- Le foglie non vengono inserite

Si simuli `iterPostorder` su un BT a piacere

Alberi di decisione

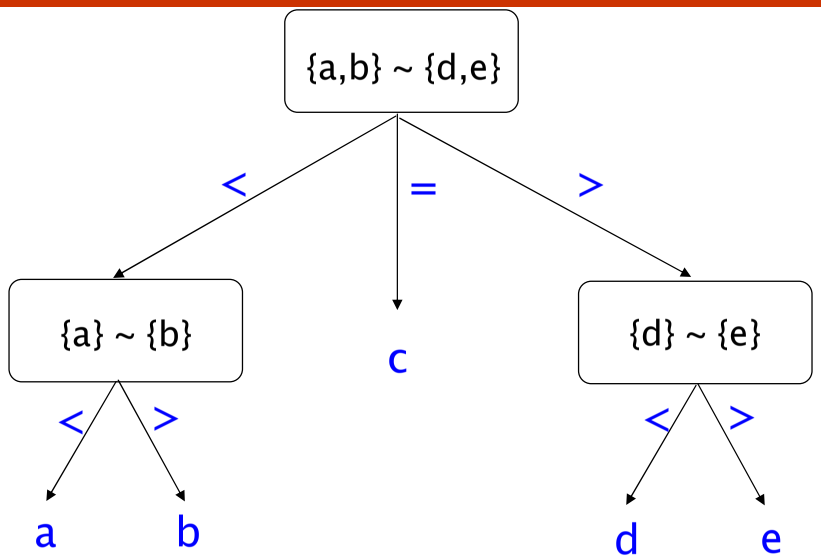
Un albero di ***decisione*** (o ***decisionale***) è un diagramma ad albero che riassume tutte le possibilità di un processo atto a risolvere un problema tramite una sequenza di decisioni. Ogni nodo interno rappresenta una domanda, ogni arco una risposta, ogni foglia dell'albero una soluzione.

Gli alberi di decisione trovano applicazione nel campo dell'Intelligenza Artificiale e del Machine Learning.

Monete contraffatte

Esempio 9.3 – (Schaumm pag. 170). Si devono esaminare 5 monete all'apparenza identiche per stabilire quale di loro è contraffatta. L'unica cosa che distingue la moneta fasulla dalle altre è il peso, minore di quello delle monete autentiche. Come devono essere pesate le monete per trovare quella fasulla?

Monete Contraffatte

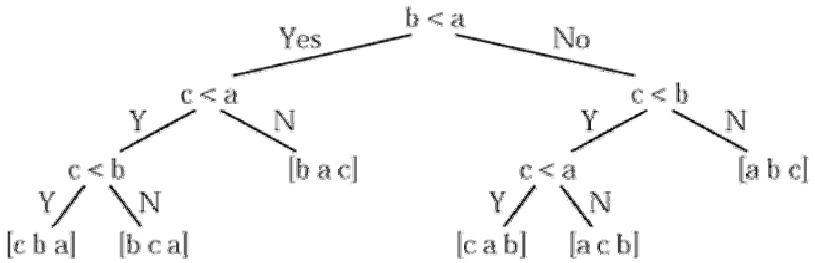


Esercizi.

a) Risolvere il problema con 7 o più monete.

b) Supporre che la moneta contraffatta, abbia un peso non necessariamente più basso da quelle "buone"

Ordinamento per confronti



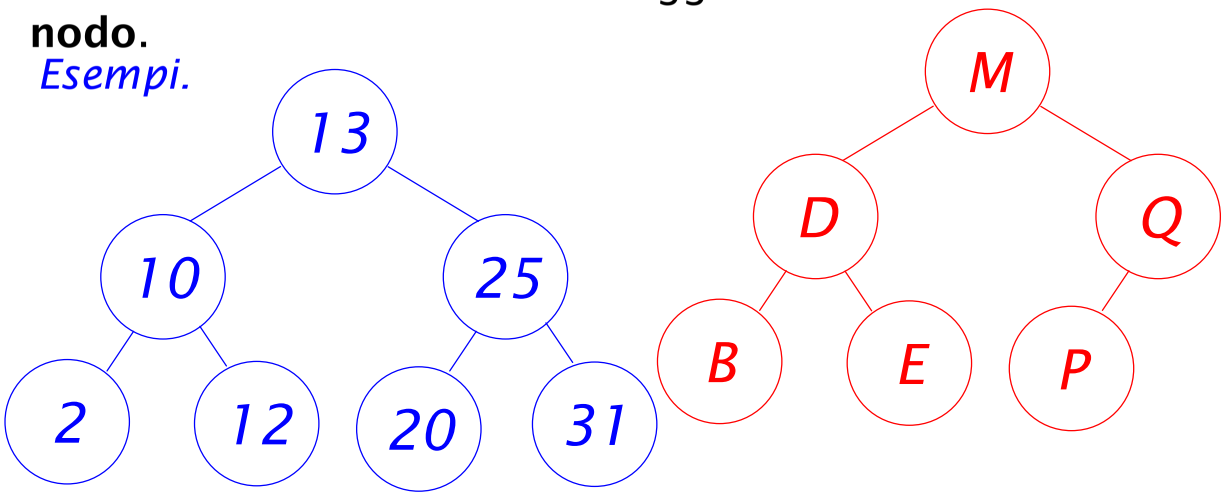
Si supponga di voler ordinare 3 numeri (a, b, c) procedendo per confronti. Possiamo generare l'albero di decisione corrispondente che dovrà avere **almeno** 6 foglie, ovvero 3! foglie.

Verrà utilizzato per calcolare un *lower bound* per gli algoritmi di ordinamento basati su confronti.

Albero binario di ricerca (BST)

Un albero binario si dice **di ricerca** se, per ogni **nodo**, tutte le etichette del sottoalbero sinistro sono minori dell'etichetta del **nodo**, e tutte le etichette del sottoalbero destro sono maggiori dell'etichetta del **nodo**.

Esempi.



Minore e maggiore hanno senso per qualunque relazione di ordinamento totale.