# Esercizi su BST

IASD a.a. 2015-16
A. De Bonis

# Esercizio 1

- Scrivere un algoritmo che restituisce una lista con gli m elementi con chiave piu` piccola di un BST

# Soluzione 1 esercizio 1

Algo(Albero,m):

1. IF(Albero.dimensione==0) return null;
2. ELSE {Lista=creaListaVuota();
3. Algo_Aux(Albero.radice, Lista,m);
4. Return Lista;
5. }

# Soluzione 1 esercizio 1

Assumiamo che L abbia il campo dimensione

```
1. Algo_Aux(nodo,L,m):
2. IF(nodo ==null) Return;
3. ELSE {
4.    Algo_Aux(nodo.sx, L,m);
5.     if(L.dimensione<m)
6.          inserisciFondo(L, nodo.dato); //inserisce l'elemento alla fine di L
7.    if(L.dimensione<m)
8.         Algo_Aux(nodo.dx, L,m);
9.  Return;
10.}
```

# Soluzione 2 esercizio 1

- Ora assumiamo che la lista NON abbia il campo dimensione per rendere l'esercizio piu` interessante
- Algo_Aux restituisce un intero e prende in input un altro parametro di tipo intero

# Soluzione 2 esercizio 1

1. Algo(Albero,m):
2. IF(Albero.dimensione==0) return null;
3. Lista=creaListaVuota;
4. ELSE {
5. Algo_Aux(Albero.radice, Lista,m,0);
6. Return Lista;
7. }

# Soluzione 2 esercizio 1

```
1.  Algo_Aux(nodo,L,m,d):
2.  IF(nodo ==null) return d;
3.  ELSE {
4.      if(d<m)
5.     d=Algo_Aux(nodo.sx, L,m,d);
6.     if(d<m) {
7.          inserisciFondo(L, nodo.dato); //inserisce l'elemento alla fine di L
8.           d=d+1;}
9.     if(d<m)
10.       d=dAlgo_Aux(nodo.dx, L,m,d);
11.  return d;
12. }
```

# Esercizio 2

- Scrivere un algoritmo che restituisce una lista contenente gli elementi con chiave piu` grande di k
- L'algoritmo non deve effetuare la visita di tutto l'albero

# Soluzione esercizio 2

1. Larger_Than(Albero,k)
2. IF(Albero.dimensione==0) return null;
3. ELSE { Lista=creaListaVuota;
4. LargerThanAux(Albero.radice, Lista,k);
5. Return Lista;
6. }

# Soluzione Esercizio 2

```
LargerThanAux(Nodo,L,k):
1.  if (Nodo==null)  return;
2.          else {
3.           chiaveNodo = Nodo.dato.chiave;
4.           if (k<=chiaveNodo) {
5.                LargerThanAux(Nodo.sx,L,k);
6.                 InsFondo(L, nodo.dato);
7.           }
8.           LargerThanAux(Nodo.dx,L,k);
9.
10.        return;
11.         }
```

# Esercizio 3

- Scrivere un algoritmo che restituisce la lista degli elementi di un BST con chiavi comprese tra k1 e k2 (k1<k2)
- L'algoritmo non deve effettuare la visita di tutto l'albero

# Soluzione esercizio 3

1. Between (Albero,k1,k2)
2. IF(Albero.dimensione==0) return null;
3. ELSE { Lista=creaListaVuota;
4. Between_Aux(Albero.radice, Lista,k1,k2);
5. Return Lista;
6. }

# Soluzione esercizio 3

```
Between_Aux(Nodo,L,k1,k2):
1.  if (Nodo==null)  return;
2.        else {
3.          chiaveNodo = Nodo.dato.chiave;
4.          if (chiaveNodo >k1)
5.              Between_Aux(Nodo.sx,L,k1,k2);
6.          if (chiaveNodo >=k1 && chiaveNodo < =k2 )
7.              InsFondo(L, nodo.dato);
8.          if (chiaveNodo < k2)
9.              Between_Aux(Nodo.dx,L,k1,k2);
10.       return;
11.        }
```