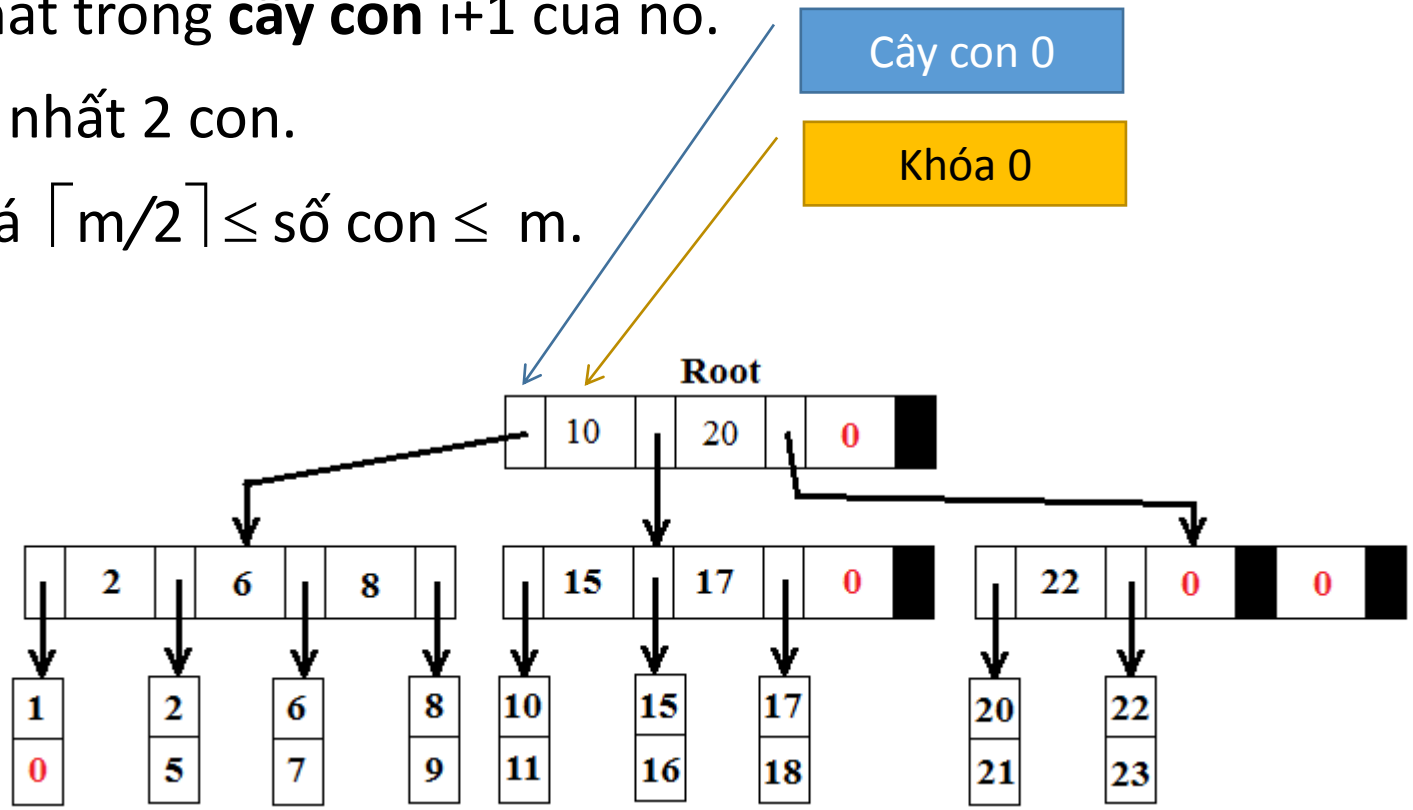
Chương 12 : B-Tree

12. 1 Định nghĩa : Một B-Tree cấp m là một cây thỏa :

1. Các thông tin được lưu trữ ở nút lá.
2. Một nút không phải nút lá có nhiều nhất m-1 khóa (key) (được sắp thứ tự); khóa i (trong nút đang xét) là khóa bé nhất trong **cây con** i+1 của nó.
3. Nút gốc, nếu không là nút lá, có ít nhất 2 con.
4. Số con của một nút không là nút lá $\lceil m/2 \rceil \leq$ số con $\leq$ m.
5. Các nút lá có cùng mức.

Ví dụ : B-Tree cấp 4



12.2 Khai báo :

```
struct BT_m_NODE {
        int key[m-1];
        BT_m_NODE *subtree[m];
        int leaf ; // leaf = 1 : nút lá; leaf = 0 : không là nút lá

        int ID[n]; // Danh sách liên kết , BST, . . .
};
```

12.3 Liệt kê các phần tử ở nút lá :

```
void BT_print(BT_m_NODE *p)
{  int i;
   if (p->leaf !=1 ){
       for(i=0 ;i<m;i++) if (p->subtree[i]!=NULL) BT_print(p->subtree[i]);
                                 else break;
   }
   else  for(i=0 ;i<n;i++) printf("%d , ", p->ID[i]);
}
int main(int argc, char* argv[])
{ BT_m_NODE *Root, *p;
  . . .
  BT_print(Root);
}
```

12.4 Tìm kiếm :

```
int BT_search(BT_m_NODE *p, int K)  // 1 : tim thay,   0 : khong tim thay
{    int i;
     if (p==NULL) return 0;
     if (p->leaf!=1)
     {      if (K < p->key[0])  return BT_search(p->subtree[0],K);
            else  for(i=m-2 ;i>=0;i--)
                        if ((K>= p->key[i])&&(p->key[i]!=0))
                              return BT_search(p->subtree[i+1], K);
      }
     else   for(i=0 ;i<n;i++)  if (p->ID[i]==K) return 1;
     return 0;
}
```