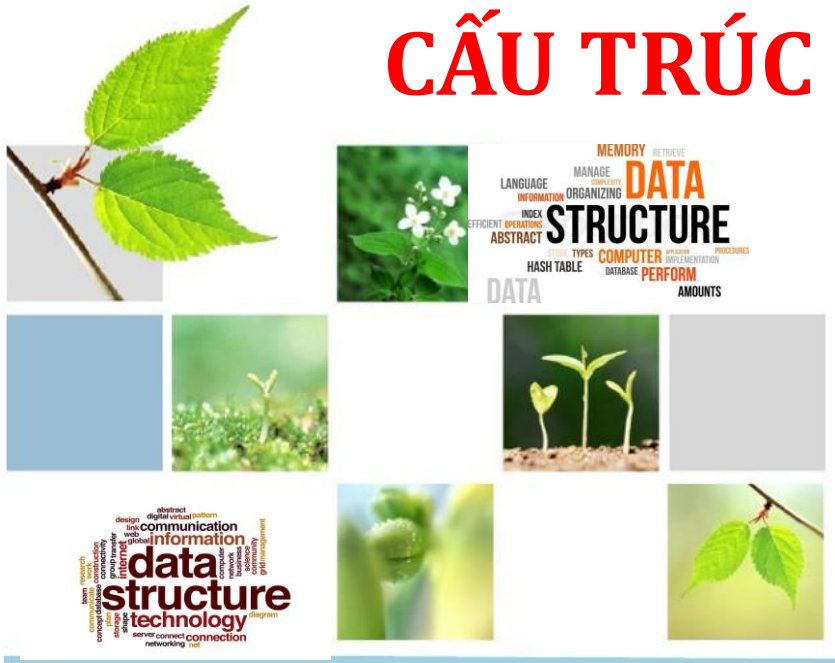
BUỔI 9: B-TREE



LÝ THUYẾT

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

IT003.N210

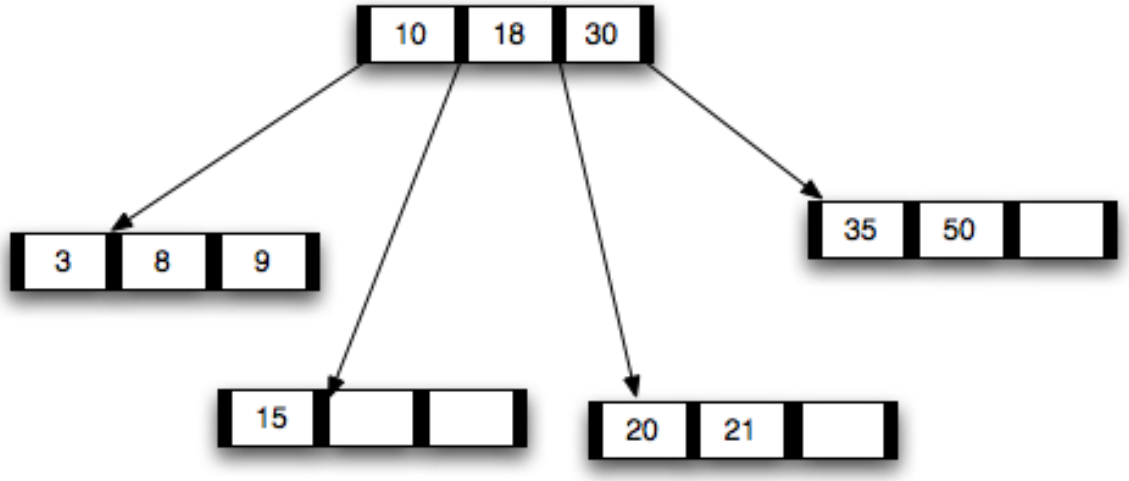
Giảng viên: Ths Đặng Văn Em

Email: vanem@uit.edu.vn

Điện thoại: 0966661006

CÂY B-TREE LÀ GÌ :

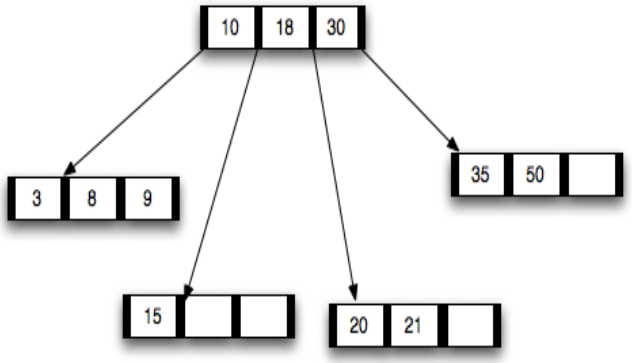
1) Giới thiệu cấu B-TREE.
2) Các thao tác trên cây B-TREE.



1. GIỚI THIỆU CẤU TRÚC B –TREE ?

CẤU TRÚC CÂY B-TREE?

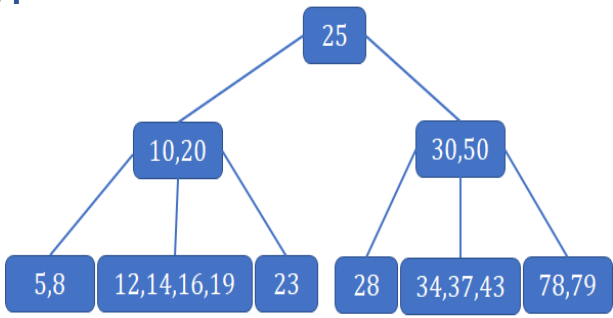
- B-Tree là cấu trúc dữ liệu phù hợp cho việc lưu trữ ngoài do R.Bayer và E.M.Mc Creight đưa ra năm 1972.
- B-Tree được tối ưu hóa cho các hệ thống đọc và ghi dữ liệu lớn.
- B-Tree là cây tự cân bằng (self-balancing). Khi thêm hoặc xoá 1 node thì cây sẽ đảm bảo chiều cao của cây càng thấp càng tốt.



1. GIỚI THIỆU CẤU TRÚC B –TREE ?

ĐỊNH NGHĨA B-TREE?

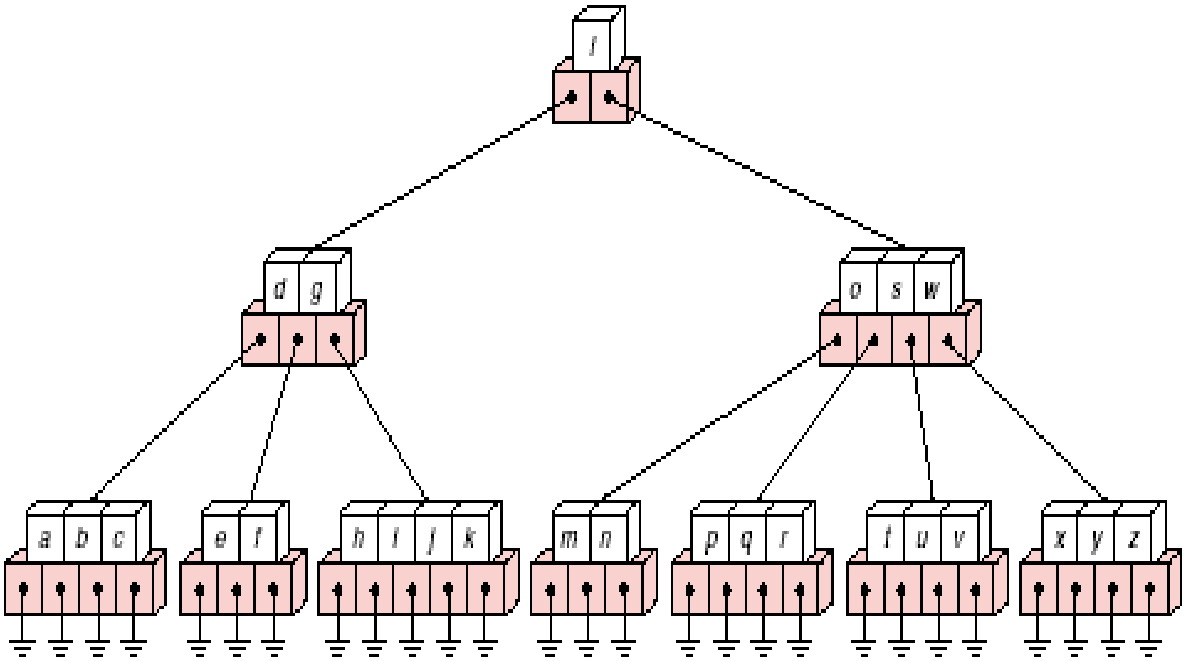
B-Tree: Cho số tự nhiên k > 0, **B-Trees** bậc m với **m = 2*k-1** là một cây thỏa mãn các tính chất :



- Tất cả các node lá là cùng mức.
- Tất cả các node có **tối đa** m con.
- Tất cả node, trừ node gốc và node lá, có **tối thiểu** **k+1** node con.
- Tất cả các node, trừ node gốc, có từ **k** cho đến **m-1** khóa (keys). Node gốc có từ **1** đến **m-1** khóa.
- Một node không phải lá và có **n** khóa thì phải có **n+1** node con.

1. GIỚI THIỆU CẤU TRÚC B –TREE ?

ĐỊNH NGHĨA B-TREE?

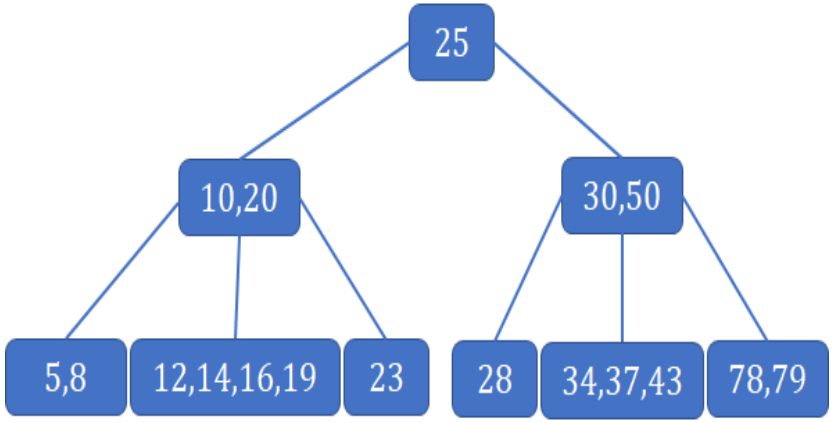


B-TREE BẬC 5 CÓ 3 MỨC

2. CÁC THAO TÁC TRÊN B –TREE ?

CÁC THAO TÁC TRÊN B-TREE?

- Khai cấu trúc của B-Tree
- Thêm node khóa X vào B-Tree
- Thao tác tìm kiếm trên B-Tree.
- Duyệt cây B-Tree.



CÁC THAO TÁC TRÊN B-TREE?

❖ **Khai báo:**

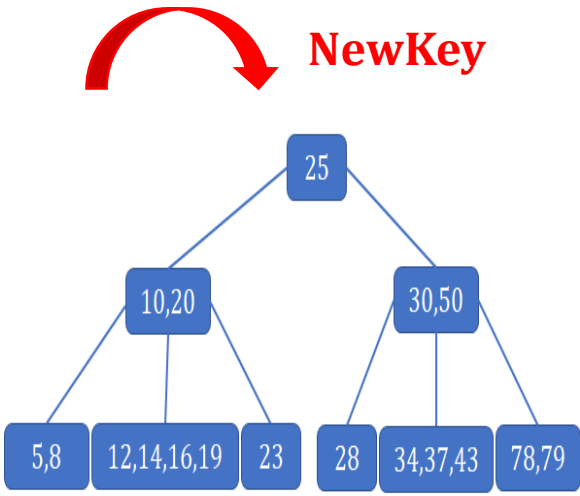
```
typedef struct
{
   int count;                  // số khoá của node hiện hành
   int Key[Order-1];           // mảng lưu trữ các  khoá của node
   int *Branch[Order]; // các con trỏ chỉ đến các cây con, Order-Bậc của cây
} BNode;                       //Kiểu dữ liệu của node
typedef struct BNode *pBNode;     // con trỏ node
pBNode *Root;                              // con trỏ node gốc
```

2. CÁC THAO TÁC TRÊN B –TREE ?

THÊM NODE KHÓA X VÀO B-TREE?

Quá trình thêm một khoá (NewKey) vào B-Tree có thể được mô tả như sau:

- Tìm node **NewKey** nếu có trên cây thì kết thúc công việc này tại node lá (không thêm vào nữa).
- Thêm **NewKey** vào node lá, nếu chưa đầy thì thực hiện thêm vào và kết thúc.

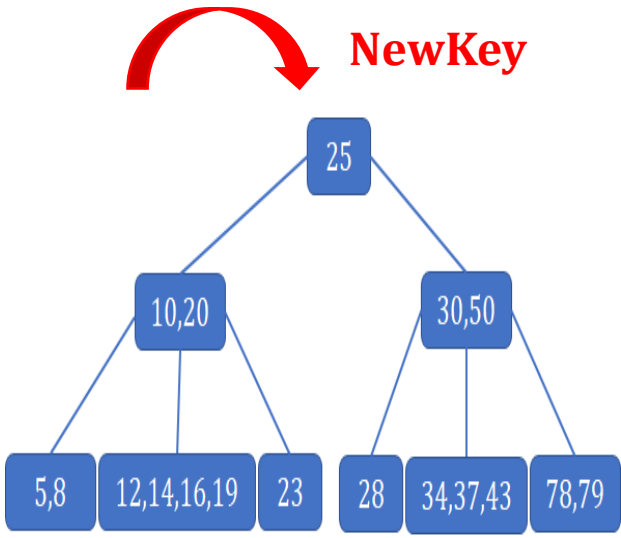


Node đầy là node có số khoá= (bậc của cây)-1

2. CÁC THAO TÁC TRÊN B –TREE ?

THÊM NODE KHÓA X VÀO B-TREE?

- Khi node được thêm vào bị đầy, node này sẽ được tách thành 2 node cùng mức, khoá median sẽ được đưa vào node mới.
- Khi tách node, khoá median sẽ được dời lên node cha, quá trình này có thể lan truyền đến node gốc
- Trong trường hợp node gốc bị đầy, node gốc sẽ bị tách và dẫn đến việc tăng trưởng chiều cao của cây

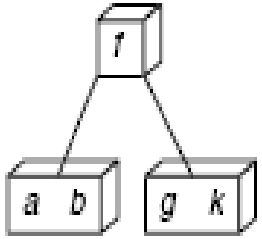


2. CÁC THAO TÁC TRÊN B –TREE ?

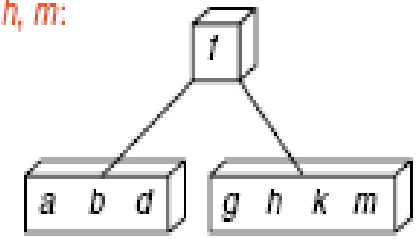
VÍ DỤ MINH HỌA THÊM NODE KHÓA X VÀO B-TREE?

1. a, g, f, b:

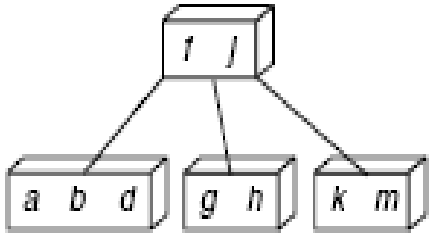
2. k:



3. d, h, m:



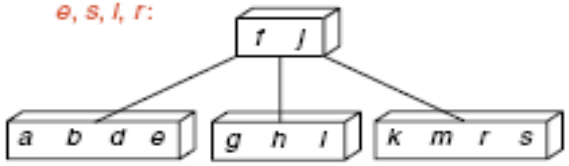
4. j:



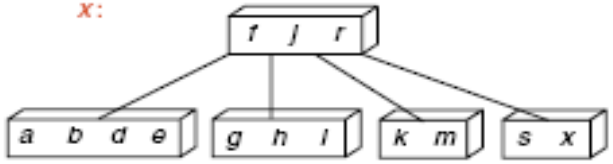
2. CÁC THAO TÁC TRÊN B –TREE ?

VÍ DỤ MINH HỌA THÊM NODE KHÓA X VÀO B-TREE?

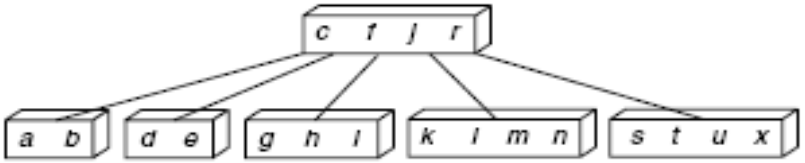
5. *e, s, l, r*:



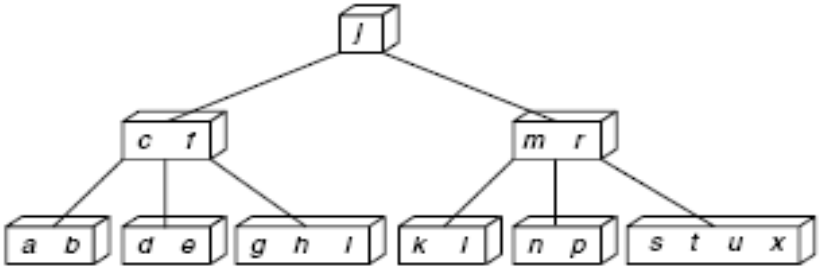
6. *x*:



7. *c, l, n, t, u*:



8. *p*:



2. CÁC THAO TÁC TRÊN B –TREE ?

THÊM NODE KHÓA X VÀO B-TREE?

```
if(Root == NULL)
   Root = makeroot(k);
else
{
   s = search(k, &position, &timthay);
      if (timthay) cout<<"Không thêm vào được";
    else insert (s, k, position);
}
```

2. CÁC THAO TÁC TRÊN B –TREE ?

THÊM NODE KHÓA X VÀO B-TREE?

- Thêm khóa k vào vị trí position của nút lá s (s và position do phép toán search() trả về)
- Nếu nút lá s chưa đầy: gọi phép toán insnode để chèn khóa k vào nút s
- Nếu nút lá s đã đầy: tách nút lá này thành 2 nút nửa trái và nửa phải

void insert (pBNode **s, int k, int position)**

2. CÁC THAO TÁC TRÊN B –TREE ?

THÊM NODE KHÓA X VÀO B-TREE?

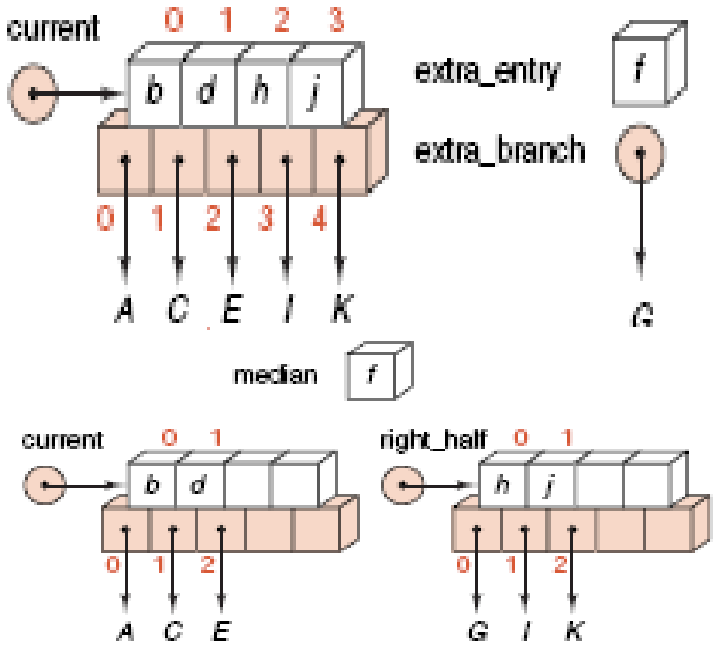
```
void insert (pBNode s, int f, int position)
{
   pBNode current, right_half, P, extra_branch;
   int pos, extra_entry, median;
   //khởi động cac tri trước khi vào trong vòng lặp tách các node dãy
   current = s;
   extra_entry = f;
   extra_branch = NULL; // vi current la node la nen gan
                        //  extra_branch la NULL
   pos = position;
   p = father (current);
   // Vong lap tach cac node day current
```

2. CÁC THAO TÁC TRÊN B –TREE ?

THÊM NODE KHÓA X VÀO B-TREE?

```
while (p != NULL && current -> count == Order)
{
   split(current, extra_entry, extra_branch, position, right_half, median);
   // Gan lai cac tri sau lan  tach node truoc
   current = p;
   extra_entry = median;
   extra_branch = right_half;
   pos = nodesearch (p, median);
   p = father (current);
}
```

Position=2,order=5



2. CÁC THAO TÁC TRÊN B –TREE ?

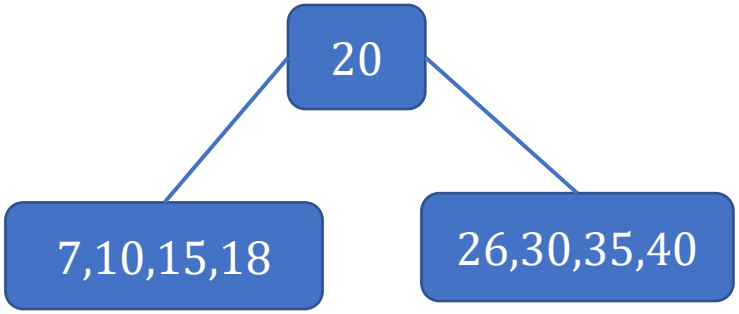
THÊM NODE KHÓA X VÀO B-TREE?

```
// Truong hop node current chua day va current khong phai la node goc
   if(current - > count+1 < Order)
{
   //chen extra_entry va extra_branch tai vi tri position cua node current
   insnode (current, extra_entry, extra_branch, pos);
   return;
}
//Truong hop node current la node goc bi day, tach node goc nay va tao node goc moi
split (current, extra_entry, extra_branch, pos, right_half, median);
Root = makeroot (median);    // tao node goc moi
// Gan lai hai nhanh cay con cua node goc moi la current va right_half
Root -> Branch[0] = current;
Root -> Branch[1] = right_half;
}
```

2. CÁC THAO TÁC TRÊN B –TREE ?

MINH HỌA THÊM NODE VÀO B-TREE?

- Thêm khóa X=22



- Thêm khóa X=20.
- Sau đó thêm 40 10 30 15
- Sau đó thêm 35 7 26 18 22

2. CÁC THAO TÁC TRÊN B –TREE ?

MINH HỌA THÊM NODE VÀO B-TREE?

- Thêm khóa X=20.
- Sau đó thêm 40 10 30 15
- Sau đó thêm 35 7 26 18 22
- Sau đó thêm 42, 13, 46, 27
- Sau đó thêm 38, 24, 45

Vẽ hình minh họa các bước.

2. CÁC THAO TÁC TRÊN B –TREE ?

MINH HỌA THÊM NODE VÀO B-TREE?

- **Thêm khóa (order 5): 1 12 8 2 25 5 14 28 17 7 52 16 48 68 3 26 29 53 55 45.**
- **Thêm khóa (order 5): 3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56**

2. CÁC THAO TÁC TRÊN B –TREE ?

TÁCH NODE B-TREE?

- Tách node đầy current, phép toán này được gọi bởi phép toán Insert
- Current là node đầy bị tách, sau khi tách xong nút current chỉ còn lại một nửa số khóa bên trái extra_entry, extra_branch và position là khóa mới, nhánh cây con và vị trí chèn vào nút current
- Node right_half là nút nửa phải có được sau lần tách, node right_half chiếm một nửa số khóa bên phải
- Median là khóa ngay chính giữa sẻ được chèn vào node cha

2. CÁC THAO TÁC TRÊN B –TREE ?

TÁCH NODE B-TREE?

```
void split (pBNode current, int extra_entry, pBNode extra_branch,
        int position, pBNode &right_half, int &median)
{
   pBNode p;
   p = new pBNode; //cap phat node nua phai
   //truong hop chen extra_entry va extra_branch vao node   nua phai
   if(position > Order/2)
   {
   copy(current, Order/2+1, Order – 2, p);
   insnode (right_half , extra_entry, extra_branch, position- Order/2 -1);
current->numtrees = Order/2+1; //so nhanh cay con con   lai cua node nua trai
   median = current -> key[Order/2];
   right_half = p ;
   return;
}
```

2. CÁC THAO TÁC TRÊN B –TREE ?

TÁCH NODE B-TREE?

```
// truong hop extra_entry la median
if(position == Order/2)
{
   copy(current, Order/2, Order-2, p);
   current->numtrees = Order/2+1;   //so nhanh cay con con lai cua node nua trai
   //Dieu chinh lai node con dau tien cua node nua phai
   current -> Branch[0] = extra_branch;
   median = current -> key[Order/2];
   right_half = p;
   return;
}
```

2. CÁC THAO TÁC TRÊN B –TREE ?

TÁCH NODE B-TREE?

```
//Truong hop chen extra_entry va extra_branch vao node nua trai
if(position < Order/2)
{
copy(current, Order/2, Order-2, p);
current->numtrees = Order/2;    //so nhanh cay con con lai cua node nua trai
   median = current -> key[Order/2- 1];
   insnode(current, extra_entry, extra_branch, position);
   right_half = p;
   return;
}
}
```

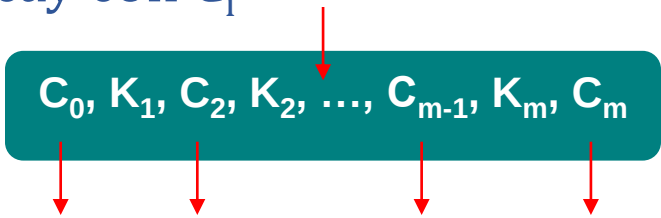
2. CÁC THAO TÁC TRÊN B –TREE ?

THÊM VÀO NODE LÁ B-TREE?

```
void insnode (pBNode current, int extra_entry, pBNode extra_branch, int position)
{
  //doi cac nhanh cay con va cac khoa tu vi tri  position tro ve sau xuong mot vi tri
  for(int i = current->count; i >= position+1; i--)
    {
    current -> Branch[i+1] = current -> Branch[i];
    current -> key[i] = current -> key[i - 1];
    }
    // gan khoa extra_entry vao vi tri position
    current -> key[position] = extra_entry;
    // Gan nhanh extra_branch la nhanh cay con ben phai cua khoa extra_entry
    current -> Branch[position + 1] = extra_branch;
    //tang so khoa cua node current len 1
    current -> count +=1;
}
```

TÌM KIẾM NODE B-TREE?

- Xét trong hình trên, khoá cần tìm là X. Với m đủ lớn ta sử dụng phương pháp tìm kiếm nhị phân, nếu m nhỏ ta sử dụng phuong pháp tìm kiếm tuần tự. Nếu X không tìm thấy sẽ có 3 trường hợp sau xảy ra:
  - $K_i < X < K_{i+1}$ Tiếp tục tìm kiếm trân cây con C_i
  - $K_m < X$. Tiếp tục tìm kiếm trên C_m
  - $X < K_1$. tiếp tục tìm kiếm trên C_0



- Quá trình này tiếp tục cho đến khi node được tìm thấy. Nếu đã đi đến node lá mà vẫn không tìm thấy khoá, việc tìm kiếm là thất bại

2. CÁC THAO TÁC TRÊN B –TREE ?

TÌM KIẾM NODE B-TREE?

- **Cài đặt phép toán nodesearch :** Trả về vị trí nhỏ nhất của khóa trong nút current bắt đầu lớn hơn hay bằng k. Trường hợp k lớn hơn tất cả các khóa trong nút current thì trả về vị trí current -> count

```
int nodesearch (pBNode current, int k)
{
   int i;
   for(i=0;i<current->count &&
         current->key[i] < k; i++);
   return (i);
}
```

2. CÁC THAO TÁC TRÊN B –TREE ?

TÌM KIẾM NODE B-TREE?

- Tìm khóa k trên B-Tree. Con trỏ current xuất phát từ gốc và đi xuống các nhánh cây con phù hợp để tìm khóa k có trong một nút current hay không.
- Nếu có khóa k tại nút current trên cây:
  - Biến found trả về giá trị TRUE
  - Hàm search() trả về con trỏ chỉ nút current có chứa khóa k
  - Biến position trả về vị trí của khóa k có trong nút current này

2. CÁC THAO TÁC TRÊN B –TREE ?

TÌM KIẾM NODE B-TREE?

- Nếu không có khóa k trên cây:
  - Lúc này current=NULL và q(nút cha của current) chỉ nút lá có thể thêm khóa k vào nút này được.
  - Biến found trả về giá trị FALSE
  - Hàm search() trả về con trỏ q là nút lá có thêm nút k vào
  - Biến position trả về vị trí có thể chèn khóa k vào nút lá q này

2. CÁC THAO TÁC TRÊN B –TREE ?

TÌM KIẾM NODE B-TREE?

```
pBNode search(int k, int &position, int &found)
{
   int i;
   pBNode current, q;
   q = NULL;
   current = Root;
   while (current !=NULL)
   {
      i = nodesearch (current, k);
      if(i< current->count && k == current->key[i]) //tim thay
      {
         found = TRUE;
         position = i;            // vi trí tìm thay khoa k
         return(current);     // node co chua khoa k
      }
```

2. CÁC THAO TÁC TRÊN B –TREE ?

TÌM KIẾM NODE B-TREE?

```
q = current;
      current = current ->Branch[i];
   }
   //Khi thoát khỏi vòng lặp trên là không tìm thấy, lúc này current=NULL,
   //q là node là có thể thêm khóa k vào node này, position là vị trí có thể chèn khóa k
   found = FALSE;
   position = i;
   return (q); //trả về node lá
} //end search()
```

2. CÁC THAO TÁC TRÊN B –TREE ?

DUYỆT B-TREE?

```
void traverse(pBNode proot)
{
  if(proot == NULL)  return; //dieu kien dung
    else      // de qui
    {   //vong lap duyet nhanh cay con Branch[i] va in khoa key[i] cua  node proor*/
    for(i = 0; i < proot -> count; i++)
      {
        traverse (proot ->Branch[i]);
        printf ("%8d", proot -> key[i]);
      }
    traverse (proot -> Branch[proot -> count]); //duyet nhanh cay con cuoi cung cua node
    }
}
```