==========

B*-Tree の挿入アルゴリズム

==========

```
1. proc insert_record(record)
2.      Input:
3.          record :  挿入するレコード
4.      entry = (key 値, record)
5.      insert (ルートノードのポインタ, entry)


1. proc insert (P,record)
2. Input:
3.          P   : 対象ノードへのポインタ(最初は root)
4.          record :  挿入するレコード
5.      output
6.       newEnt :  親ノードに挿入するエントリ（なければ NULL)
7.      N = P のさすノード
8.      if N が non-leaf ノードである：
9.           K_i <  entry の key 値  ≤ K_i+1    となる i を探す
10.           newEnt_fromchild = insert(P_i, entry) //再帰
11.           if N に空きがある：
12.               insert_entry(N, newEnt_fromchild)
13.               return null;
14.           else : //N に空きがない場合
15.               newEnt = divide_nonleaf(N)
16.               if  ノード N が root ノードである：
17.                       新しい root ノード R を作成
18.                       insert_entry(R, newEnt)
19.                       return null
20.               else:
```

```
21.                     return newEnt
22.     else : //N が leaf ノードである
23.             if N に空きがある
24.                     insert_entry(N, record)
25.                     return null
26.             else
27.                     newEnt = divide_leaf(N)
28.                     if  ノード N が root ノードである：
29.                             新しい root ノード R を作成
30.                             insert_entry(R, newEnt)
31.                             return null
32.                     else
33.                             return newEnt
```

==========

B*-Tree のノード分割アルゴリズム

==========

```
proc divide_nonleaf ( N )
    input
        N： 分割する non-leaf ノード
    output
        newEnt： 親ノードに挿入するエントリ
    process:
        ノード M を作る
          N の d+2 番目から 2d+1 番目までの key 値を M へ移動
          N の d+2 番目から 2d+1 番目までのポインタを M へ移動
          N の子孫の一番右のリーフノードから M の子孫の一番左の
            リーフノードへ兄弟ポインタを貼る
        newEnt = <N の d+1 番目の key 値, M へのポインタ>
        N の d+1 番目の key 値を削除
        return newEnt;
```

proc divide_leaf (N)
　input：
　　N： 分割する leaf
　output
　　newEnt： 親ノードに挿入するエントリ

　process：
　　ノード M を作る
　　　N の d+1 番目から 2d+1 番目までのエントリを M へ移動
　　　N から M へ兄弟ポインタを張る
　　newEnt = <M の 1 番目の key 値, M へのポインタ>
　　return entEnt;

==========

B*-Tree のエントリ追加アルゴリズム (non-leaf)

==========

insert_entry (newEnt)
　input:
　　newEnt： 挿入するエントリ。key 値とポインタを持つ。
　process:
　　$K_i \leq$ newEnt の key 値 $< K_{i+1}$ となる i を探す
　　i 番目と i+1 番目の間に空のエントリを追加する
　　（i+1 番目以降は 1 つずつ順番がずれる）
　　i+1 番目のポインタに i+2 番目のポインタを代入する
　　i+2 番目のポインタに newEnt のポインタを代入する
　　i+1 番目の key 値に newEnt の key 値を代入する