# 平衡二叉排序树的插入

```
if(平衡二叉排序树 t == Ø)  {直接插入；}
if(关键码 key == p-> key) {已经存在；  return 0；}
if(parent_p == Ø)
      /* 父结点为空，即平衡二叉排序树中只有p一个结点 */
{
      if(key > p->key) { p->rlink = create(key); p->bf = 1;}
      else  { p->llink = create(key); p->bf = -1;}
      return 1;    /* 处理成功 */
}
else
      /* 平结点的父结点不为空，分平衡因子为0，-1,1分别处理 */
{
      if(parent_p->key == 0)         /*parent_p->bf == 0，分四种情况讨论*/
      {           /* ●代表parent_p，●代表p，○代表新插入结点，○其他结点*/
            if(key < p->key){
                  p->llink = create(key); p->bf = parent_p->bf = -1;
            }
            else if(p->key < key  && key < parent_p->key){
                  p->rlink = create(key); p->bf = 1; parent_p->bf = -1;
            }
            else if(parent_p->key < key && key < p->key){
                  p->llink = create(key); p->bf = -1; parent_p->bf = 1;
            }
            else{       /* key > p->key */
                  p->rlink = create(key); p->bf = parent_p->bf = 1;
            }
      }
      else if(parent_p->bf == -1)   /*parent_p->bf == -1，分三种情况讨论 */
      {
            if(key < p->key){
                  p->llink = create(key); LL_rotate(p);
            }
            else if(p->key < key && key < parent_p->key){
                  p-rlink = create(key); LR_rotate(p);
            }
            else{
                  parent_p->rlink = create(key); parent_p->bf = 0;
            }
      }
      else        /*parent_p->bf == 1，分三种情况讨论 */
      {
            if (key > p->key){
                  p->rlink = create(key); RR_rotate(p);

            }
            else if(parent_p->key < key && key < p->key){
                  p->llink = create(key); RL_rotate(p);
            }
            else{       /*key < parent_p->key 的情况 */
                  parent_p->llink = create(key); parent_p->bf =0;
            }
      }

      return 1;
}
```