アルゴリズムとデータ構造1　　演習問題 (7)　　回答

n 個の長方形の 2 次元配置を木で覚えるため, 下のように定義された木を用いる.

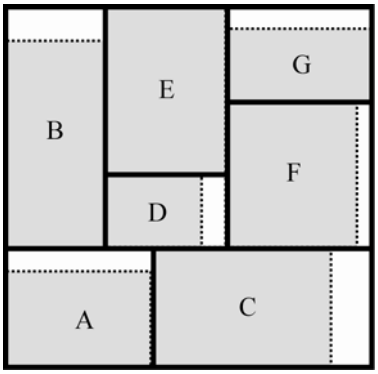
```
typedef  struct {
            char[9]          rect ;          /*  長方形の名前あるいは並び方を入れる欄  */
            int              height ;        /*  長方形の高さを入れる欄    */
            int              width ;         /*  長方形の幅を入れる欄   */
            int              x0 ;            /*  長方形の左下隅の点の x 座標を入れる欄    */
            int              y0 ;            /*  長方形の左下隅の点の y 座標を入れる欄   */
            NDCELL↑          lm_child ;      /*  長男へのポインタ. 長男が無いときは  NULL   */
            NDCELL↑          r_sibling ;     /*  次弟へのポインタ. 次弟が無いときは NULL   */
      } NDCELL ;
typedef        NDCELL↑        NODE ;        /*    NODE 型は, NDCELL 型のセルを指すポインタ  */

NODE           T ;                          /*    木 T は NODE 型のポインタ  */
```

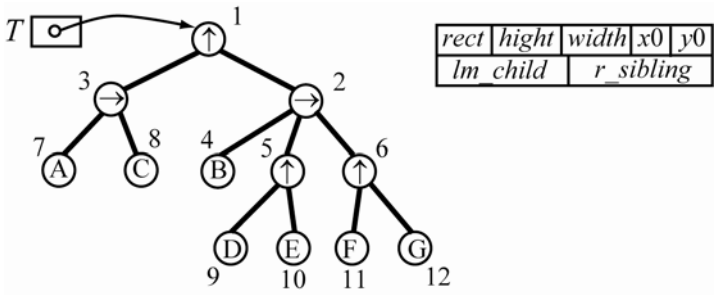
この木の各葉は, 与えられた *n* 個の長方形の一つに対応し, 英数字 10 文字で表された長方形の名前 (*rect*) , その高さ (*height*) および幅 (*width*) の情報が与えられている. 従って, この木には葉 (子を持たない点) が *n* 個ある. ただし, 長方形の名前の先頭の 2 文字には, “BU” あるいは “LR” のいずれも使われていないものとする.

葉以外の各点 *v* も (2 次元配置から定められる) 長方形 R(*v*) に対応し, この長方形 R(*v*) は, *v* を根とする部分木に含まれる葉に対応した長方形を囲む. 従って, 木の根 *r* に対応する長方形 R(*r*) は与えられた *n* 個の長方形全てを囲む. このような点に対応した長方形の高さおよび幅は与えられていないので, 計算によって求めねばならない.

葉以外の点 *v* には, その名前 (*rect*) として, “BU” あるいは “LR” のどちらかの文字が入っており, 長方形 R(*v*) がどのような構造になっているかを表している. 点 *v* の名前 (*rect*) に “BU” (Bottom Up の意味) が入っていると, *v* の子に対応した長方形が長男から順に下から上に配置されていることを示し, “LR” (Left to Right の意味) が入っていると, *v* の子に対応した長方形が長男から順に左から右に配置されていることを示す. “BU” および “LR” をこのような目的に使用するため, 長方形の名前の先頭に用いることができない.

下に示した2次元配置を表すこのような木 *T* をその右に示すが, 木の図では, “BU” および “LR” はそれぞれ上向きおよび右向きの矢印で表している. 例えば, 木の根 1 には上向きの矢印 (BU) が入っているので, 根に対応する全体の長方形は, 長方形 A, C からなる長方形の上に, 長方形 B, E, D, G, F からなる長方形があることを示す. また, 根の一番末の子の点 2 には右向きの矢印 (LR) が入っているので, 左から順に 3 つの長方形が配置されていることを示す. なお, 右の図において実線で表された長方形が必ずしも点に対応した長方形ではない. 例えば, 長方形 A, C を含む点 3 に対応した長方形 R(3) は, 長方形 A, C を含む実線の長方形ではなく, その右辺は長方形 C の点線で描かれた右辺と考える.





このような木 *T* で表された 2 次元配置を描画するため, 葉以外の点に対応した長方形の高さと幅, ならびに各点に対応した長方形の左下隅の点の座標を計算する関数を作りたい. この関数は, 次のように考えれば, 容易に作れる.

まず, 木の根 r に対応する長方形 R(r) の左下隅の点の座標を原点とする. すなわち, 根 r の $x0$ および $y0$ を 0 と定める. そうすると, 根 r に最初に訪問する際, その欄 $x0$ および $y0$ に入れるべき値を指定できる.

そこで, 点 v を根とする部分木を深さ優先探索する再帰的関数 *DFStree*(v, x, y) を考える. ここで, 引数の x および y は, それぞれ点 v の欄 $x0$ および $y0$ に入れるべき値を示すものとする.

この再帰的関数 *DFStree*(v, x, y) では, 次のようにして, 点 v を根とする部分木に含まれる葉以外の点に対応した長方形の高さと幅, ならびに点 v を根とする部分木内の各点に対応した長方形の左下隅の点の座標を計算できることが分かる.

1. 点 v の欄 $x0$ および $y0$ に, それぞれ入力された整数 x および y を入れる.
2. 点 v が葉であれば, 処理を終了.
3. 点 v が葉でなく, 欄 *rect* に “BU” が入っていると, 点 v に対応した長方形は, v の子に対応した長方形を下から上に積み上げたものになっている. そこで, v の子に対応した長方形を左下詰めにすることを考えると, 以下の関係が成り立つことが分かる.
   - v に対応した長方形の高さは, v の子に対応した長方形の高さの和.
   - v に対応した長方形の幅は, v の子に対応した長方形の幅の最大値.

   さらに, v の子 w に対応した長方形の左下隅の点の x 座標 ($x0$) および y 座標 ($y0$) の値に関して, 以下の関係が成り立つ.
   - x 座標値 ($x0$) は, v に対応した長方形の左下隅の点の x 座標値と同じ.
   - y 座標値 ($y0$) は, w が v の長男の場合, v に対応した長方形の左下隅の点の y 座標値と同じ. そうでない場合, w の直ぐ上の兄の点を u とすると, u に対応した長方形の左下隅の点の y 座標値に, u に対応した長方形の高さを加えた値.
4. 点 v が葉でなく, 欄 *rect* に “LR” が入っていると, 点 v に対応した長方形は, v の子に対応した長方形を左から右に並べたものになっている. そこで, v の子に対応した長方形を左下詰めにすることを考えると, 以下の関係が成り立つことが分かる.
   - v に対応した長方形の高さは, v の子に対応した長方形の高さの最大値.
   - v に対応した長方形の幅は, v の子に対応した長方形の幅の和.

   さらに, v の子 w に対応した長方形の左下隅の点の x 座標 ($x0$) および y 座標 ($y0$) の値に関して, 以下の関係が成り立つ.
   - x 座標値 ($x0$) は, w が v の長男の場合, v に対応した長方形の左下隅の点の x 座標値と同じ. そうでない場合, w の直ぐ上の兄の点を u とすると, u に対応した長方形の左下隅の点の x 座標値に, u に対応した長方形の幅を加えた値.
   - y 座標値 ($y0$) は, v に対応した長方形の左下隅の点の y 座標値と同じ.

以上より, 点 v を根とする部分木を深さ優先探索する再帰的関数 *DFStree*(v, x, y) では, 長方形の高さおよび幅は, 帰りがけ順に計算でき, 長方形の左下隅の点の座標は, 行きがけ順に決定できることが分かる.

この関数は以下のように書ける.

アルゴリズムとデータ構造1　　演習問題 (7)　　回答

```
typedef  struct {
            char[9]          rect ;          /* 長方形の名前あるいは並び方を入れる欄 */
            int              height ;        /* 長方形の高さを入れる欄 */
            int              width ;         /* 長方形の幅を入れる欄 */
            int              x0 ;            /* 長方形の左下隅の点の x 座標を入れる欄 */
            int              y0 ;            /* 長方形の左下隅の点の y 座標を入れる欄 */
            NDCELL↑          lm_child ;      /* 長男へのポインタ. 長男が無いときは NULL */
            NDCELL↑          r_sibling ;     /* 次弟へのポインタ. 次弟が無いときは NULL */
        } NDCELL ;
typedef     NDCELL↑          NODE ;          /* NODE 型は, NDCELL 型のセルを指すポインタ */

NODE        T ;                              /* 木 T は NODE 型のポインタ */
```

```
void  DFStree( NODE v, int x, int y )
    /* v が指す点を根とする部分木を探索し, その子孫の各点に対して, 以下の値を計算する */
    /* v が葉の場合, 欄 x0 および y0 に, それぞれ入力された整数 x および y を入れる */
    /* v が葉でない場合, v に対応する長方形の高さと幅, ならびにその左下隅の点の座標を計算する */
    /* v は NULL ではなく, 長方形の 2 次元配置を表した木の点を指していなければならない */
{
    NODE   w ;                              /* w は NODE 型で, v の子を指すための局所変数 */

    (↑v).x0 := x ;                          /* 点 v の左下隅の点の x 座標値を x にする */
    (↑v).y0 := y ;                          /* 点 v の左下隅の点の y 座標値を y にする */

    w := (↑v).lm_child ;                    /* w を v が指す点の長男を指すようにする */
    if ( w ≠ NULL ) {                       /* v が指す点が葉でないならば, 以下を実行 */
        (↑v).height := 0 ;                  /* v が指す点に対応する長方形の高さの初期化 */
        (↑v).width := 0 ;                   /* v が指す点に対応する長方形の幅の初期化 */
        while( w ≠ NULL ) {                 /* v が指す点の子 (w が指す点) に対して以下を実行 */
            DFStree( w, x, y ) ;            /* w が指す点に対して DFStree を実行 */
            UpdateSize( v, w, &x, &y ) ;    /* v が指す点に対応する長方形の高さと幅を更新し, */
                                            /* 同時に, w の次弟に対応する長方形の左下隅の座標を計算する */
            w := (↑w).r_sibling ;           /* w を w が指す点の次弟を指すようにする */
        }
    }
 } /* DFStree */
```

```
void  UpdateSize( NODE v, NODE w, int↑ px, int↑ py )
    /* v は長方形の 2 次元配置を表した木の点を指し, w はその子を指す */
    /* v が指す点に対応する長方形の形状情報を, w が指す点に対応する長方形のそれを用いて更新する */
    /* また, w の次弟に対応する長方形の左下隅の座標を計算し, ↑px および ↑py に入れる */
    /* v も w も NULL ではなく, w が指す点に関する DFStree は終了している */
{
    if ( (↑v).rect = "BU" ) {               /* v が指す点の子に対応する長方形は下から上に積まれている */
        (↑v).height := (↑v).height + (↑w).height ;
        if ( (↑v).width < (↑w).width )  (↑v).width := (↑w).width ;
        ↑py := ↑py + (↑w).height ;
    }
    else {                                  /* v が指す点の子に対応する長方形は左から右に並んでいる */
        if ( (↑v).height < (↑w).height )  (↑v).height := (↑w).height ;
        (↑v).width := (↑v).width + (↑w).width ;
        ↑px := ↑px + (↑w).width ;
    }
} /* UpdateSize */
```