C から入る C++ 最終レポートのヒント

1 (3) を解くための解説

1.1 edgeMap とは何か

Node と Pin と Edge の関係が図 1 のようになっていたことを思いだそう. ここで, 出力の Edge が

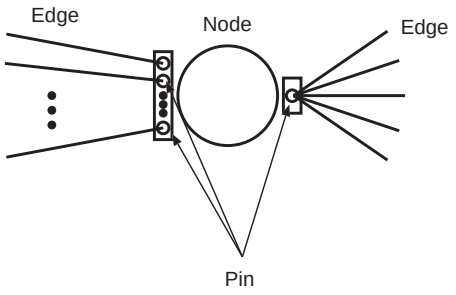

図 1: Node と Pin と Edge.

複数ある場合を考えよう. 半加算器を作成すると, このように出力の Edge は複数存在する.

このとき, 一つの Pin に複数の Edge が存在するわけであるが, この複数の Edge を格納するクラスが edgeMap クラスである. 実際に common.h の Pin クラスの定義を見ると, 図 2 のように Pin クラスのデータメンバに edgeMap クラスのインスタンス edge が存在しているのがわかる.

```
class Pin{
  public:
    ...
    edgeMap edge;
    ...
```

図 2: Pin クラスは edgeMap をメンバに持つ

この edgeMap は, 資料で扱った「線形リスト」や「二分木」のように, どんどんデータを追加できるようなデータ構造になっている (実際には map は二分木で実現されている).

1.2 edgeMap の使われ方 (1) ～データの追加～

EdgeMap のインスタンス edge へと値を追加する際, common.h の edgeAdd 関数が使われている (図 3). edgeCount が key として, Edge クラスのポイ

```
void edgeAdd(Edge* edg){
  edge.insert(edgeMap::value_type(
edgeCount++, edg )); /* 一行 */
}
```

図 3: edgeMap への Edge の追加.

ンタ edg が data として挿入されていることがわかる. このとき, 整数の edgeCount が edgeCount++ によって 1 づつ増加されながら値が挿入されることから, このときの模式図は図 4 のようになる. 第

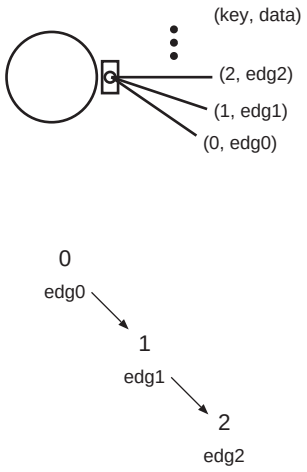

図 4: Edge が追加される様子.

十二回資料の図 4 のように, 二分木クラスを線形リストのように利用していることがわかるだろう.

1.3 edgeMap の使われ方 (2) ～値の伝搬～

edgeMap が存在する Node で論理値が変更されると, その出力は Edge の先にある Node に伝搬される. このとき, 全ての Edge に対して出力を伝搬させるためにイテレータが用いられている.

simulation.cpp に, 図5のようなコードがある. や

```
edgeMap::iterator eitr; // イテレータの定義
for( eitr = NodeObj− >outPin[0]− >edge.begin();
     eitr != NodeObj− >outPin[0]− >edge.end();
     eitr++ ){
     // for 文で全てのエッジをスキャン

    Node* dnode = eitr− >second− >dstNode;
        //伝搬先のノード
    if(dnode != 0) // 伝搬先の Node が存在すれば
        scheduleEvent(new execEvent(time+1,
        NodeObj− >out, eitr− >second− >dstPin,
        dnode)); //新たに発生したイベントを登録
}
```

図 5: イテレータを用い, 全ての Edge に値を伝搬させる.

や複雑であるが, イテレータと for 文を用いて全ての Edge をスキャンしているのがポイントである. なお, 図 6 にあるように, この関数内では edgeMap

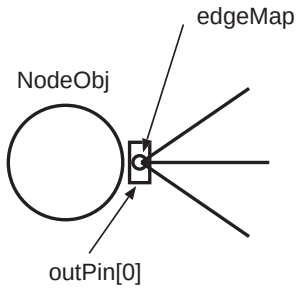

図 6: NodeObj− >OutPin[0]− >edge について.

は「NodeObj− >outPin[0]− >edge」 で表されることに注意.

また,「eitr− >second」であるが, これはイテレータが指す値のうちの data を表す. すなわち, 我々が作成した BTree の場合は「btree.Itr()− >getData()」に相当する.

1.4 どのように edgeMap を置き換えるか? ～準備～

以上でみた edgeMap を BTree で置き換えるのが課題 (3) である. そのための準備を以下に示す.

(1) 課題 (3) を実行するためのフォルダを作成し, そこに新たなプロジェクトを作成する.

(2) プロジェクトファイルのあるフォルダに以下のファイルをコピーする.
BTree.h, BTree.cpp, common.h, common.cpp, simulation.h, simulation.cpp, simMain.cpp
ただし, BTree.cpp は課題 (1), (2) を終らせた後のものとする.
そして, これらのファイルを全てプロジェクトに加える.

(3) BTree クラスから Edge クラスを参照できるように, Btree.h の上部に「class Edge;」と記述する.

(4) common.h から BTree クラスを呼び出すため, common.h の「#include <map>」の次の行に「#include "BTree.h"」と記述する.

(5) まずここでコンパイルが通るか確認. コンパイルが通ったら, 変更を開始する.

最後に, 注意点を幾つかあげる.

- 一つのファイルのみをコンパイルし, 検証したいときがある. 例えば, BTree.cpp のみをコンパイルしたいときは, BTree.cpp を BCC Developer 内でダブルクリックした後,「プロジェクト→コンパイル」を実行する.
- BTree クラスでのイテレータの使い方などは btree_test.cpp が参考になるだろう.