```
//**************************************************//
//                                                  //
//        IRA.cpp                                   //
// Implementation file  for class IRA               //
//**************************************************//
//
//   Author:   Hui Jin
//   Modification:    Mar 10, 2000 First version
//   This file defines the class of Irregular Repeat Accumulate code.
//   the code essentially is a concatenate of LDGM code and Accumulate
code.



#include <math.h>

#include "IRA.h"


IRAcode::IRAcode(int info_len, int check_len, int _LEdgenum, int nv[],
degree_sequence * _pLV, degree_sequence * _pLC, \
                 double _sigma, RandomGenerator * _Rand, int
*_Interleaver)

{

   Rand= _Rand;



   infobits_len=info_len;
   checkbits_len=check_len;

   pLV =  _pLV;
   pLC =  _pLC;


   codebits_len = infobits_len+checkbits_len;
   rate = (infobits_len*1.0) /(codebits_len*1.0);

   LEdge_num = _LEdgenum;
   REdge_num= 2*checkbits_len-1;


   sigma=_sigma;

   y_r=new double [codebits_len];
```

```
    RVarNode= new var_node [checkbits_len];
    LVarNode= new var_node [infobits_len];


    CheckNode=new check_node [checkbits_len];

    LMessage=new double [LEdge_num];
    RMessage=new double [REdge_num];

    Lbelief =new int [LEdge_num];
    Rbelief =new int [REdge_num];


    Interleaver=new int [LEdge_num];
    for(int i=0; i<LEdge_num; i++) Interleaver[i]= _Interleaver[i];


    getYrAWGN();

    initLVNode(nv);


    initRVNode();
    initCNode();

    initMessage();
    initM0();

}


void IRAcode::initLVNode(int nv[])
{

    int deg_index=0;
    int edge_index=0;

        //for simplicity
    int edge[LEdge_num];
    for(int j=0; j< LEdge_num; j++) edge[j]=Interleaver[j];

    int i;

    int up_range= nv[0];
    for(i=0; i<infobits_len-1; i++)
       {
         //shoulbe be >=, since 0 is included.
         if(i>=up_range)
           {
             deg_index++;
```

```
            up_range+= nv[deg_index];
           }


        LVarNode[i].init(i, pLV->d[deg_index],0);
        LVarNode[i].init_Edgeid(&edge[edge_index], pLV->d[deg_index]);
        edge_index+= pLV->d[deg_index];

      }

   //the last bit has to make the edge number is exactly LEdge_num;
      i= infobits_len-1;
      int last_deg= LEdge_num-edge_index;


      LVarNode[i].init(i, last_deg, 0);
      LVarNode[i].init_Edgeid(&edge[edge_index], last_deg);

 // debug
 //  for(i=0; i<infobits_len; i++)
 //    LVarNode[i].display_edgeid();
}


void  IRAcode::initRVNode()
{
   int edge[REdge_num];
   for(int j=0; j< REdge_num; j++) edge[j]=j;

   int edge_index=0;
   int i;

   for(i=0; i<checkbits_len-1; i++)
      {
        RVarNode[i].init(i, 2, 1);
        RVarNode[i].init_Edgeid(&edge[edge_index], 2);
        edge_index+=2;
       }
//the last bit has degree 1.
   i=checkbits_len-1;
   RVarNode[i].init(i, 1, 1);
   RVarNode[i].init_Edgeid(&edge[edge_index], 1);


}

void IRAcode::initM0()
{
   double r;
   int j;
```

```
   for(int i=0; i< codebits_len; i++)
     {
       r=y_r[i];
           //if it belongs to the left var node.
       if(i<infobits_len)
           LVarNode[i].init_m0(r, sigma);

       else //if belongs to right.
          {
             j=i-infobits_len;
            RVarNode[j].init_m0(r, sigma);
           }

      }
}



void IRAcode::initCNode()
{
  //local variables
   int j;
   int i;
   int deg_index=0;
   int edge_index=0;

//LEFT PART, as similar to init_LVNode().
//just take care the last node, to make edges used up in all nodes.
 /*   The following main idea is, divide all the variable nodes into
    several ranges. [0, LC.fn[0]*infobits_len].
                    [LC.f[0] *infobits_len,
(LC.fn[0]+LC.fn[1])*infobits_len],
                     ...
    If node with id I belongs to the kth range, then it has degree
    LC.degree[k-1].
     */
 //for simplicity

   int edge[LEdge_num];
   for(j=0; j< LEdge_num; j++) edge[j]=j;

   double up_range=pLC->fn[0]* checkbits_len;
   for(i=0; i<checkbits_len-1; i++)
      {
        //in case there's zero in it.
        while(i>=floor(up_range))
           {
             deg_index++;
             up_range+= pLC->fn[deg_index]* checkbits_len;
            }
```

```
          (i==0) ? CheckNode[i].init(i, pLC->d[deg_index],1): \
                   CheckNode[i].init(i, pLC->d[deg_index],2);


          CheckNode[i].init_LEdgeid(&edge[edge_index], pLC-
>d[deg_index]);
          edge_index+= pLC->d[deg_index];



      }
   // To make the edge number the same.
   //the last bit will have left degree LEdge_num-edge_index;
       i= checkbits_len-1;
       int last_d= LEdge_num-edge_index;

       CheckNode[i].init(i, last_d,2);
       CheckNode[i].init_LEdgeid(&edge[edge_index], last_d);




//RIGHT PART, as similar to init_RVNode();
     int r_edge[REdge_num];
     for(j=0; j< REdge_num; j++) r_edge[j]=j;

   edge_index=0;
   CheckNode[0].init_REdgeid(&r_edge[0],1);
   edge_index++;

   for(i=1; i<checkbits_len; i++)
      {
         CheckNode[i].init_REdgeid(&r_edge[edge_index], 2);
         edge_index+=2;
       }


  //debug
 //     for(i=0; i<checkbits_len; i++)
 //    CheckNode[i].display_edgeid();
}


void IRAcode::update_message()
{
    int i;
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.