

電機專題結報

錯誤編碼偵測與更正

指導教授：

王忠炫 教授

學生：

何智興

專題動機

在現在這個物聯網、互聯網的時代，全世界有數億台行動裝置像是手機、電腦等都同時連到網路上，發射、接收訊號，因此對於訊號的傳輸的要求也越來越高。通訊技術也從過去的 3G、4G 到現在即將運用的 5G 技術，就是希望能更準確、更有效率的傳遞每個裝置的訊號。

通訊又分成類比與數位這兩種形式：類比訊號就是大自然中一切的訊號，像是聲音、影像等訊號皆屬於類比。以聲音為例，我們將聲音的大小轉成電壓的大小，得到的是連續的電壓變化，而這種連續的訊號就稱為類比訊號。數位訊號就是將連續的類比訊號加工成由 0、1 所組成的不連續訊號，也是目前多數電子產品訊號的表示方式，而數位訊號的優點包括方便儲存、傳送、加密與解密和容易偵錯、除錯，而這也是我們有興趣的主題。

通訊為我們生活帶來巨大方便，其中對於訊號的偵錯與除錯的技術最為感興趣，因此想更加地了解其背後原理和技術。

先備知識

編碼理論分成兩大區塊，資料壓縮和錯誤控制。資料壓縮嘗試壓縮資料的大小，讓傳輸和儲存資料時更加地有效率，像是我們常見的 zip 檔、rar 檔皆是如此。錯誤控制則是增加位元到傳輸的資料之中來防止各種干擾，確保資料的穩定性。

在基本的通訊模型中，資料會在傳送端先經過編碼器之後傳入通道之中，到達接收端之後，經由解碼器還原出資料。在編碼端則是由三部分所組成的：source coding、channel coding 和 modulation: source coding 將資料轉換成數位的形式，而這樣的轉換是一對一的轉換，目的是消除冗餘的資訊，把不影響到資料傳輸結果的部分刪除，提升資料傳輸的效率；channel coding 則是增加冗餘資料來對抗有噪聲存在的環境；經 source coding 和 channel coding 之後，資料是離散的形式，不利於傳輸，因此需要 modulation 將資料轉成連續的波，傳入通道至解碼端，而解碼端的運作方式與編碼端則完全相反，然而解碼完後的資料仍然有可能會與初始資料不相同，是因為訊號的失真和通道中的噪聲所造成，因此解碼器只能還原出資料的估計值。

現在廣泛使用數位訊號而不是類比訊號來傳輸的原因是因為數位訊號在傳輸上比類比訊號更加的可靠，因為當不可避免的干擾出現時，可以被偵測、還原成原本的資料；而數位訊號的表示方式是先將資料量化成數個等級，而每組資料則是以最相近的等級來表示，一種最簡單的方式是以四個位元來表示一組資料，前面三個位元是以二進位的方式來表示量值大小，最後一個位元是用來表示資料是正值或是負值，

而每組資料被選中的機率相同，各組機率的加總和則為 1。最優化的編碼方式是讓每組資料平均的位元數越少越好，而前提條件是不能讓各組資料的表示方式重複，滿足這樣條件的最小位元數稱為 least significant bit(LSB)。

除了上述提到的二進位表示方式之外，常見的資料表示方式還有八進位和十六進位，八進位的表示方式從 0 到 7，十六進位的表示方式除了 0 到 9 之外，還加上了英文字母的 A 到 F。

介紹完編碼的方式之後，現在介紹干擾的部分，我們從個位元發生錯誤的機會是獨立的機率開始，而這種個位元發生干擾的機率是獨立的模型稱為 white noise，雖然在真實情況中，會有特定的位元較為容易發生干擾，而且往往這些干擾會互相影響，但我們從較為簡單的 white noise 開始探討干擾，再進一步地了解更複雜的情形。當一個位元發生錯誤的機率是 p ，而一組資料有 l 個位元，假設之中 n 個位元發生錯誤，機率為 $(l!)*(p^n)*((1-p)^{(l-n)})/(n!)*((l-n)!)$ 。

在學會如何計算一組資料發生幾個錯誤位元的機率後，介紹一種最簡單的偵測錯誤編碼，就是在各組資料的最後加入一個位元，使得整組資料各位元相加後對 2 取餘數為 0，透過這種稱為 single parity check code，可以在一組資料中發生奇數個錯誤時偵測出來，而 single parity check code 的 redundancy rate 是 $1/(n+1)$ ，因此我們會直觀地認為當一組資料中包含越多的位元越有效率，但是相對地，它的可靠度就會降低，當然地，也不是越多的 redundancy bit 代表偵測到錯誤發生的能力一定越好，而是要設計出一種能有效地檢查方式，才能最有效

率，因此要增加幾個和如何設計每個 redundancy bit 又是一門學問了。

另一種常見的 codeword 是運用在電腦上面的 ASCII code，每組 codeword 由 7 個 bit 組成，因此共能表示 128 種符號，但電腦中一個 byte 有 8 個 bit，所以我們會在最後面用 single parity check code 來讓它具有 error detecting 的功能。

接著討論到較為複雜的情形，就是把 burst error 考慮進來之後，single parity check code 無法那麼有效地偵測出是否有錯誤，因此學習了 simple burst error detecting code 的方式來處理當 burst error 的長度小於一組資料長度的情況。Simple burst error detecting code 就是把每個位置的 bit 相加，再對 2 取餘數，因此還會再產生一組資料，使得每個位置的 bit 相加之後對 2 取餘數為 0，而這也是解碼端用來驗證是否有錯誤發生的方式。還有另一種方式，weighted code。就是當全部的符號數為質數(這裡表示為 n)時，我們給每個符號一個相對應的數字，從 0 到 $n-1$ ，接著讓各符號對應的值乘上它們的權重然後加總，而每個符號的權重就是看他們是從最後一個符號開始數的第幾個符號加上一，接著再用 n -(加總後的值對 n 取餘數)，並找出這個值所對應的符號，放在原本那些要傳送的符號後面，讓解碼端能夠用相同的方式來算出加總各符號的加權值對 n 取餘數是否為 0 的方式來偵測錯誤。

在介紹完 error detecting code 之後，接著介紹 error correcting code，而資料中提到 repetition code 和 Hamming code 這兩種方式。在

repetition code 中，以 three times repetition code 為例，就是每個 bit 總共重複了 3 次，是一種 single error correcting code，雖然大大地確保了資料的正確性，但是卻十分的沒有效率。Hamming code 是由 G 矩陣(generator matrix)和 H 矩陣(parity check matrix)所組成：G 矩陣的功能就是將 k 個 bit 的資料轉換成 n 個 bit(稱作 codeword)，因此 G 矩陣的大小為 $k \times n$ ；而 H 矩陣乘上 codeword 的轉置矩陣之後，能檢查這組 codeword 是否有受到干擾，如果乘完的結果為零向量，則這組 codeword 並未受到干擾而發生錯誤；若有發生錯誤，我們則能夠從 H 乘上 codeword 的轉置矩陣所得到的向量來推測最有可能是哪個位元發生了錯誤。而推測到底是哪個位元發生錯誤的方式，也可以透過 Venn diagram，如果一個圓之中有偶數個 1，那麼這個圓裡代表的位元原則是沒有問題，反之，若是有奇數個 1，那麼這個圓有些位元是錯的，在算出每個圓的對與錯之後，就能輕易地找到是哪個位元發生了錯誤；除此之外，也會發現，表示圓是否正確的向量(正確的圓以 0 表示，有錯誤位元的圓以 1 表示)與 H 乘上 codeword 的轉置矩陣所得到的向量相同。我們已知 H 乘上 codeword 的轉置矩陣所得到的向量不為零向量代表存在著錯誤，而更正的方式就是找出 H 矩陣中哪個列向量與 H 乘上 codeword 的轉置矩陣所得到的向量一樣，然後在 codeword 所對應的位元加上 1 然後再對 2 取餘數來修正；而(7,4)Hamming code 最多只能修正一個錯誤位元，這也是我們接著要討論的主題。

一種編碼方式最多能夠修正幾個位元的錯誤取決於 Hamming distance，就是兩個 codeword 之間有幾個位元數是不相同的，假設任

兩個 codeword 間的 Hamming distance 至少為 3，那麼這樣的編碼方式能夠偵測到 2 個位元的錯誤(算法：最小的 Hamming distance-1)，能夠更正 1 個位元的錯誤(算法：最小的((Hamming distance-1)/2)。而剛剛舉例的(7,4)Hamming code 能夠更正一個錯誤，能偵測部分的兩個錯誤，但有些部分兩個錯誤的 codeword 則會在修正之後產生更多的錯誤，而(8,4)Hamming code 就是在 codeword 的最後面多加一個位元，而這個位元的產生方式就是如果前面七個位元有偶數個 1，那麼就補 0；反之，如果有奇數個 1，那麼則補 1；(8,4)Hamming code 優於(7,4)Hamming code 的地方在於它除了能更正一個錯誤位元之外，它能在兩個位元發生錯誤時正確地偵測到，不會像(7,4)Hamming code 可能更正出偏差更多的 codeword。我們可以從這個例子發現到，增加 redundant bit 可以增加 error correcting 的能力，但是會降低傳輸有效資料的效率。

除了從幾何的角度判斷一組編碼方式能夠偵測和更正幾個錯誤位元之後，接著從代數的角度來理解。假設一種編碼方式，它的 information bit 有 k 個位元，codeword 有 n 個位元，我們令 $r=n-k$ ，那麼它的特徵向量共有 2^r 種表示方式，而當我們希望這種編碼方式能夠更正一個錯誤位元時，那麼 2^r 要大於等於 $n+1$ ，原因是因為當 codeword 沒有錯誤產生時，需要一組特徵向量來表示，加上當 codeword 有一個位元發生錯誤時，由於總共有 n 個位元，所以總共需要 n 組特徵向量來表示，因此更正一個錯誤位元至少需要 $n+1$ 組相異的特徵向量來表示。當 $n=2^r-1$ ， $k=2^r-r-1$ 時，那我們就可以稱之為 (n,k) Hamming code。

接著討論當我們希望能夠更正兩個錯誤位元的情況，當 codeword 沒有錯誤產生時，需要一組特徵向量來表示，加上當 codeword 有一個位元發生錯誤時，由於總共有 n 個位元，所以總共需要 n 組特徵向量來表示，再加上當 codeword 有兩個位元發生錯誤時，由於總共有 $n*(n-1)/2$ 種可能，所以總共需要 $n*(n-1)/2$ 組特徵向量來表示，因此要更正兩個錯誤位元， 2^r 要大於等於 $(n*(n-1)/2)+n+1$ 。

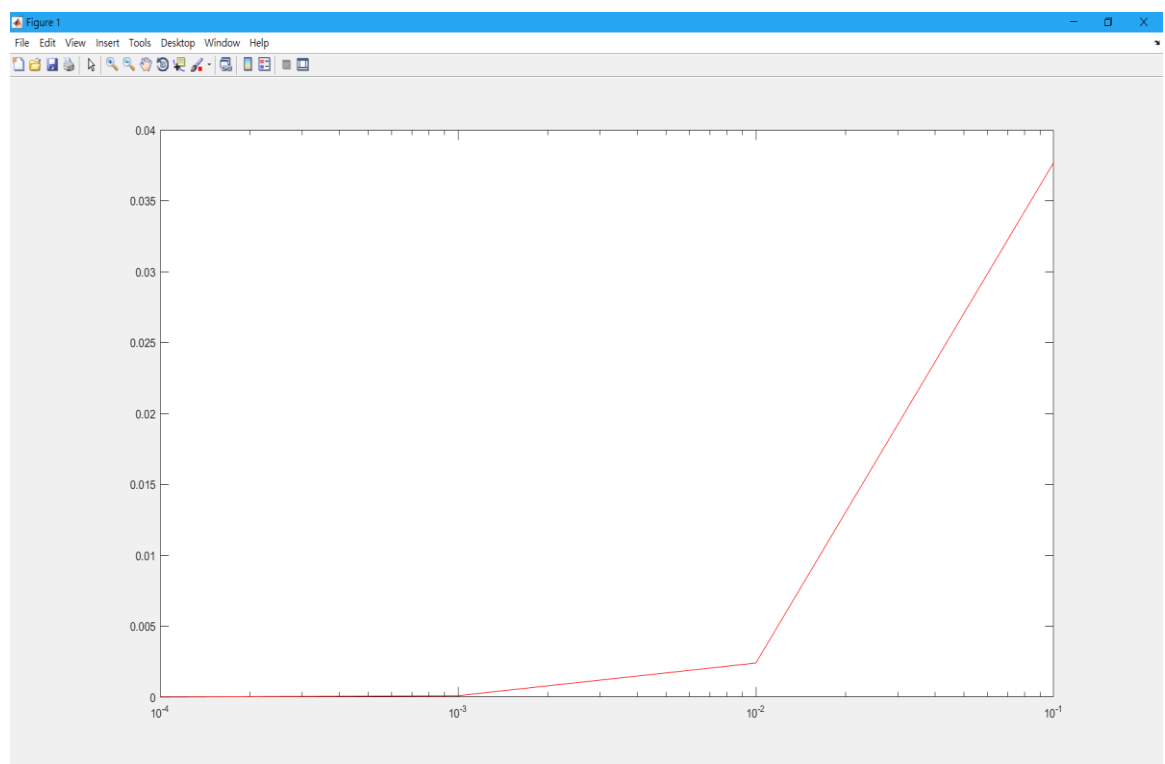
模擬與成果

透過 C++ 模擬出一個包含編碼、解碼、產生隨機錯誤等過程的 (7,4)Hamming code，盡可能地讓模擬貼近現實的狀況。首先，程式先讀入 G 矩陣(大小：4*7)，然後呼叫 G_identity 這個函式判斷這個 G 矩陣是否為一個合法的矩陣，而一個合法的矩陣，矩陣的左方或是右方之中會包含一個單位矩陣，G_identity 函式會回傳一個值，如果這個單位矩陣存在於 G 矩陣的左方，回傳 1；若在 G 矩陣的右方，回傳 2；若 G 矩陣不存在單位矩陣，則回傳-1。接著用 find_H 函式透過 G 矩陣來算出 H 矩陣(大小：3*7)，當 G 矩陣的單位矩陣在 G 矩陣的左方時，H 矩陣的單位矩陣在 H 矩陣的右方；相反地，當 G 矩陣的單位矩陣在 G 矩陣的右方時，H 矩陣的單位矩陣在 H 矩陣的左方，而 H 矩陣扣除單位矩陣的部分則是 G 矩陣扣除單位矩陣部分的轉置矩陣。

接著建立在解碼端所需要用到的資料，因為要從 codeword 轉回去變成 information bit 需要乘上 G 的轉置矩陣再乘上 G 乘 G 轉置矩陣的反矩陣，所以先用 G_G_trans 函式來算出 G 乘 G 轉置矩陣，再將結果傳至 find_inverse 函式之中，求出它的反矩陣。為了降低運算時間和成本，先利用 build_syndrome 這個函式，對 H 乘上加入經干擾的 codeword 的轉置向量所可能產生的特徵向量，建立對應的修正方式，讓 H 乘上每組 codeword 後得到的特徵向量，能迅速的從這個已經建立的表找到對應的修改方法，而不用再跟每個 H 矩陣的列向量逐一

比較，提升效率。

使用者能夠自己輸入每個位元發生錯誤的機率為多少，測試在各機率下一萬組資料經過這樣模擬的錯誤率，進行比較，而這一萬組資料的 information bit 是由電腦隨機產生所形成，產生完每組 information bit 之後送至 to_codeword 的函式和 G 矩陣相乘得到對應 codeword，接著將各組的 codeword 送至 rand_generator 來決定哪些位元發生錯誤，發生的機率則是先前使用者所輸入的機率，而這個過程也是隨機的。把這些已經發生隨機錯誤的 codeword 送至 get_syndrome 的函式中，得到每組 codeword 的特徵向量，然後利用先前 build_syndrome 這個函式所建立的表，找到每組 codeword 的修正向量，進行修正。每組 codeword 修正完之後，用 back_to_information_bit 的函式算出解碼後的 information bit，接著與原始產生的 information bit 進行比較，看看是否相同。最後，在一萬組資料測試完之後，算出錯誤率。



橫軸代表解碼成 information bit 後的錯誤率

縱軸代表每個位元發生錯誤的機率

收穫與展望

雖然在這個專題中學習到了許多編碼的基本原理以及如何去偵錯與更正，但是這些僅僅是編碼理論中最入門的小小一部份，仍然有許多更複雜和新穎的理論與模型需要去了解，才能更加了解編碼的全貌。

透過模擬模型的過程，將知識轉換成實際可運用的程式，除了開始思考如何降低運算成本、節省時間之外，對於設計程式的邏輯有很大改變，像是學會先建立資料庫，再去抓取資料，而不是每次重新運算，讓整個設計有更佳的效率。這樣的實務經驗並不是能從書本中獲得的，而是需要動手操作才能獲得的寶貴經驗。

有了這次專題的經驗之後，對於學問的研究也有了不一樣的認知，了解到做研究的過程中往往許多的知識都需要靠自己來發掘，然後了解，和一般地上課考試有很大地不一樣，不會有問題列在考卷上，而是要透過自己不斷的思考，而這也是讓我覺得很有趣的地方，因此如果之後還有機會，會想要繼續修專題，探索更多的知識。

程式碼

```
1 #include<iostream>
2 #include<cstdlib>
3 #include<time.h>
4 #include<math.h>
5 #include<vector>
6
7 using namespace std;
8 int r=1,s=0,S_count=0;
9 vector<vector<vector<int>>>> S;
10
11 /*****判斷I在G的前置後*****/
12 int G_identity(vector<vector<int>> > G,int gm,int gn){
13     int flag1=0,flag2=0;
14     for(int i=0;i<gm;i++){
15         for(int j=0;j<gm;j++){
16             if((i==j)&&(G[i][j]!=1)) flag1++;
17             else if((i!=j)&&(G[i][j]!=0)) flag1++;
18         }
19     }
20     for(int i=gn-gm;i<gn;i++){
21         for(int j=0;j<gm;j++){
22             if(((i+gm-gn)==j)&&(G[j][i]!=1)) flag2++;
23             else if(((i+gm-gn)!=j)&&(G[j][i]!=0)) flag2++;
24         }
25     }
26     if(flag1==0) return 1;
27     else if(flag2==0) return 2;
28     else return -1;
29 }
30 }
```

```
31
32 /*****找出現*****/
33 vector<vector<int>> find_H(vector<vector<int>> > G,int gm,int gn,int flag){
34     vector<vector<int>> > H;
35     for(int i=0;i<(gn-gm);i++){
36         vector<int> l;
37         l.resize(0);
38         for(int j=0;j<gm;j++) l.push_back(2);
39         H.push_back(l);
40     }
41
42     if(flag==1){
43         for(int i=gm,k=0;i<gn,k<gn-gm;i++,k++){
44             for(int j=0;j<gm;j++) H[k][j]=G[j][i];
45         }
46         for(int i=0;i<gn-gm;i++){
47             for(int j=gm;j<gn;j++){
48                 if(i==(j-gm)) H[i][j]=1;
49                 else H[i][j]=0;
50             }
51         }
52     }
53
54     else if(flag==2){
55         for(int i=gn-gm,k=0;i<gn,k<gn-gm;i++,k++){
56             for(int j=0;j<gm;j++) H[i-gn+gm][j+gn-gm]=G[j][k];
57         }
58         for(int i=0;i<gn-gm;i++){
59             for(int j=0;j<gn-gm;j++){
60                 if(i==j) H[i][j]=1;
61                 else H[i][j]=0;
62             }
63         }
64     }
65 }
```

```

61         }
62     }
63 }
64 }
65 return H;
66 }
67
68 /*****建立2^r種syndrome*****/
69 void build_syndrome(int g_count,int k,vector<int> qq){
70     if(k<g_count){
71         qq[k]=0;
72         build_syndrome(g_count,k+1,qq);
73         qq[k]=1;
74         build_syndrome(g_count,k+1,qq);
75     }
76     else if(k==g_count){
77         S_count++;
78         for(int i=0;i<g_count;i++) S[0][S_count-1][i]=qq[i];
79     }
80 }
81
82 /*****產生隨機錯誤*****/
83 int rand_generator(int b,double prob){
84     double comp=(double)rand()/(RAND_MAX+1.0);
85     if(comp<prob) b=(b+1)%2;
86     return b;
87 }
88
89 /*****information bit change to codeword*****/
90 vector<int> to_codeword(int gm,int gn,double prob,vector<int> s,vector<vector<int>> G){

```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 1 Sel: 0 Lines: 432 Length: 9877 Insert Done parsing in 0.765 seconds


```

C:\Users\ASUS-NB\Desktop\Project.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
Project.cpp
211 }
212 }
213 }
214 }
215
216 /*****get syndrome*****/
217 vector<int> get_syndrome(vector<vector<int> > H,vector<int> codeword,int gm,int gn){
218     vector<int> syndrome;
219     syndrome.resize(0);
220     for(int i=0;i<(gn-gm);i++) syndrome.push_back(0);
221     for(int i=0;i<(gn-gm);i++){
222         int ans=0;
223         for(int j=0;j<gm;j++){
224             ans=ans+H[i][j]*codeword[j];
225         }
226         syndrome[i]=ans%2;
227     }
228     return syndrome;
229 }
230
231 /*****return to information bit*****/
232 vector<int> back_to_information_bit(int gm,int gn,vector<vector<int> > inv,vector<vector<int> > G_transform, vector<int>
233     vector<int> information,temp;
234     information.resize(0);
235     temp.resize(0);
236     for(int i=0;i<gm;i++){
237         information.push_back(0);
238         temp.push_back(0);
239     }
240     for(int i=0;i<gm;i++){

```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 1 Set: 0 Lines: 432 Length: 9877 Insert Done parsing in 0.765 seconds

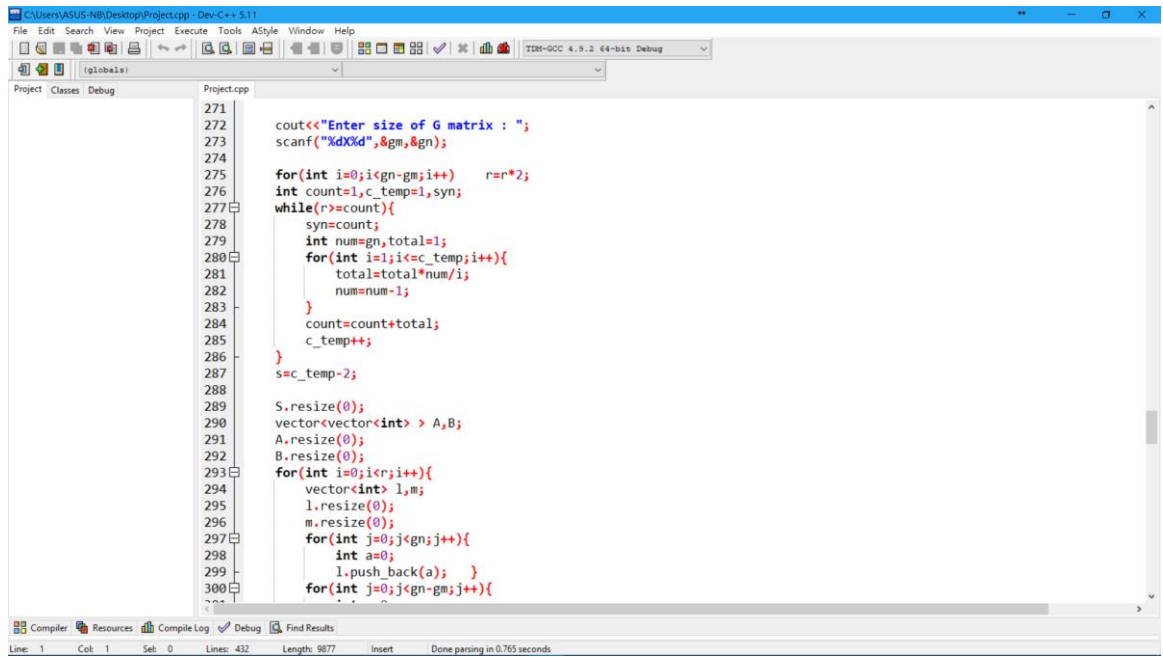
```

C:\Users\ASUS-NB\Desktop\Project.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
Project.cpp
241     int ans=0;
242     for(int j=0;j<gm;j++){
243         ans=ans+codeword[j]*G_transform[j][i];
244     }
245     temp[i]=ans;
246 }
247 for(int i=0;i<gm;i++){
248     int ans=0;
249     for(int j=0;j<gm;j++){
250         ans=ans+(int)temp[j]*inv[j][i];
251     }
252     information[i]=((int)(ans+0.5))%2;
253 }
254 return information;
255 }
256
257 int main()
258 {
259     srand(time(NULL));
260     int gm=0,gn=0,set=0,matrix_flag=-1;
261     double prob=0.0;
262     vector<vector<int> > G,H,syndrome,G_transform;
263     vector<int> inform,org_inform,codeword;
264     vector<vector<int> > inv,mul;
265     G.resize(0);
266     H.resize(0);
267     inform.resize(0);
268     org_inform.resize(0);
269     codeword.resize(0);
270     mul.resize(0);

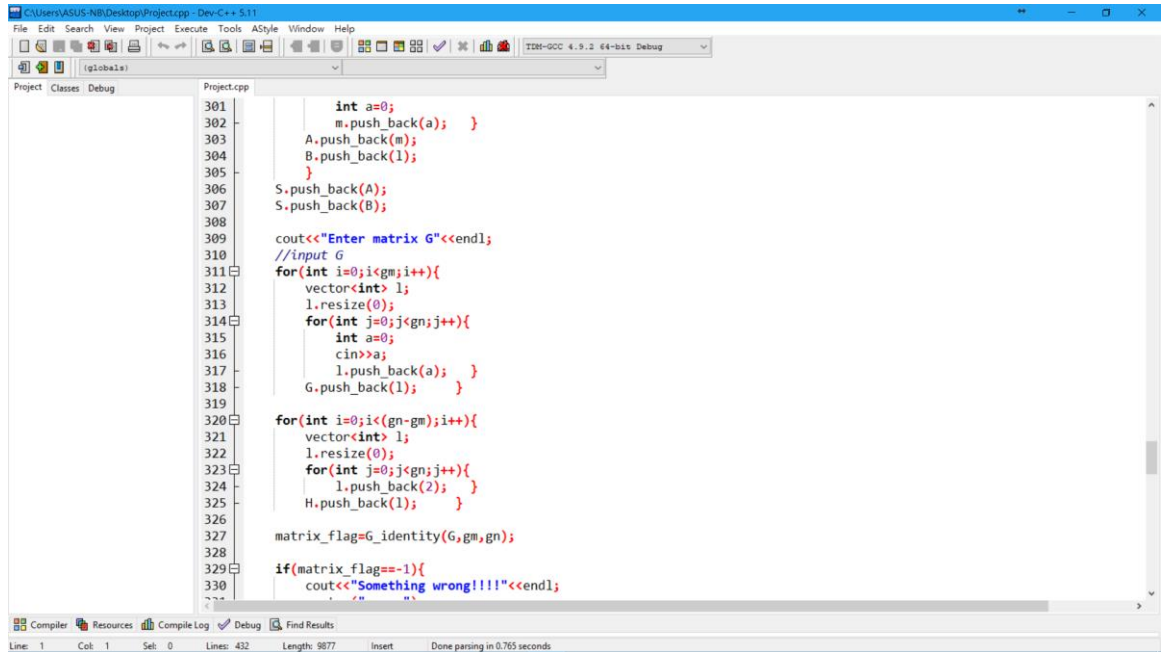
```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 1 Set: 0 Lines: 432 Length: 9877 Insert Done parsing in 0.765 seconds



```
271     cout<<"Enter size of G matrix : ";
272     scanf("%dX%d",&gm,&gn);
273
274     for(int i=0;i<gn-gm;i++)    r=r*2;
275     int count=1,c_temp=1,syn;
276     while(r>=count){
277         syn=count;
278         int num=gn,total=1;
279         for(int i=1;i<=c_temp;i++){
280             total=total*num/i;
281             num=num-1;
282         }
283         count=count+total;
284         c_temp++;
285     }
286     s=c_temp-2;
287
288     S.resize(0);
289     vector<vector<int> > A,B;
290     A.resize(0);
291     B.resize(0);
292     for(int i=0;i<r;i++){
293         vector<int> l,m;
294         l.resize(0);
295         m.resize(0);
296         for(int j=0;j<gn;j++){
297             int a=0;
298             l.push_back(a);
299         }
300         for(int j=0;j<gn-gm;j++){
```



```
301         int a=0;
302         m.push_back(a);
303         A.push_back(m);
304         B.push_back(l);
305     }
306     S.push_back(A);
307     S.push_back(B);
308
309     cout<<"Enter matrix G"<<endl;
310     //input G
311     for(int i=0;i<gm;i++){
312         vector<int> l;
313         l.resize(0);
314         for(int j=0;j<gn;j++){
315             int a=0;
316             cin>>a;
317             l.push_back(a);
318         }
319         G.push_back(l);
320     }
321     for(int i=0;i<(gn-gm);i++){
322         vector<int> l;
323         l.resize(0);
324         for(int j=0;j<gn;j++){
325             l.push_back(2);
326         }
327         H.push_back(l);
328     }
329     matrix_flag=G_identity(G,gm,gn);
330     if(matrix_flag==-1){
331         cout<<"Something wrong!!!!"<<endl;
332     }
```

```

C:\Users\ASUS-NB\Desktop\Project.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
Project.cpp
331     system("pause");
332     return 0;
333 }
334 else H=find_H(G,gm,gn,matrix_flag);
335
336 for(int i=0;i<gm;i++){
337     vector<int> l;
338     l.resize(0);
339     for(int j=0;j<gm;j++){
340         l.push_back(0);
341     }mul.push_back(l);
342 }
343 for(int i=0;i<gm;i++){
344     vector<int> l;
345     l.resize(0);
346     for(int j=0;j<gm;j++){
347         l.push_back(0);
348     }inv.push_back(l);
349 }
350 mul=G_G_trans(G,gm,gn);
351 inv=find_inverse(mul,gm);
352 vector<int> qq;
353 for(int i=0;i<gn-gm;i++) qq.push_back(0);
354 build_syndrome(gn-gm,0,qq);
355
356 vector<int> a,b;
357 for(int i=0;i<(gn-gm);i++){
358     a.push_back(0);
359 }
360 for(int i=0;i<gn;i++){
361     b.push_back(0);
362 }
363 for(int i=0;i<r;i++){
364     build_H(H,gm,gn,i,a,b,10000,0);
365 }
366 cout<<"Enter error probability : ";
367 cin>>prob;
368 for(int i=0;i<gn;i++){
369     vector<int> l;
370     l.resize(0);
371     for(int j=0;j<gm;j++) l.push_back(0);
372     G_transform.push_back(l);
373 }
374 G_transform=find_G_transform(G,G_transform,gm,gn);
375 int total_error=0;
376
377 for(int i=0;i<l;i++){
378     vector<int> l;
379     l.resize(0);
380     for(int j=0;j<gn;j++) l.push_back(0);
381     codeword.push_back(l);
382 }
383
384 for(int i=0;i<l;i++){
385     vector<int> l;
386     l.resize(0);
387     for(int j=0;j<(gn-gm);j++) l.push_back(0);
388     syndrome.push_back(l);
389 }
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2
```

