Міністерство освіти і науки України
Національний університет водного господарства
та природокористування

Кафедра обчислювальної техніки

04-04-205

МЕТОДИЧНІ ВКАЗІВКИ

для виконання лабораторних та самостійних робіт
з дисципліни
"Захист інформації в комп'ютерних системах"
студентами напряму підготовки
6.050102 "Комп'ютерна інженерія"

Частина IV. Кодування інформації

Рекомендовано
методичною комісією
напряму підготовки
"Комп'ютерна інженерія"
Протокол № 7
від 03 березня 2017 р.

Рівне 2017

Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Захист інформації в комп'ютерних системах" студентами напряму підготовки 6.050102 "Комп'ютерна інженерія". Частина IV. / П. В. Ольшанський, – Рівне: НУВГП, 2017, – 32 с.

Упорядник П. В. Ольшанський, старший викладач кафедри обчислювальної техніки.

Відповідальний за випуск
Б.Б. Круліковський, кандидат технічних наук, доцент, завідувач кафедри обчислювальної техніки.

Зміст

Вступ

Захист інформації – розділ інформатики, який вивчає питання:

- кодування інформації для передачі по лініях зв'язку, для обміну між різними пристроями, для зберігання на різноманітних носіях (*стандарти зберігання, протоколи передачі, алгоритми стискання, мінімальні коди, коди з виправленням помилок*);
- шифрування інформації для конфіденційного обміну повідомленнями, для захисту документів від несанкціонованого читання чи використання (*криптологія= криптографія+ криптоаналіз+протоколи обміну*);
- забезпечення ідентифікації користувачів при роботі в комп'ютерних мережах (*парольний захист; протоколи обміну; соціальний інжиніринг*);
- захист документів від змін і гарантування їх аутентичності та недоторканості з допомогою електронного підпису; (*електронне голосування; електронні гроші; доведення з нульовим розголошенням; розподіл секрету; сліпий підпис);*
- захист комп'ютерів і комп'ютерних систем від несанкціонованого проникнення (хакерів, зловмисників, недобросовісного персоналу) та халатного відношення працівників до політики безпеки;
- захист комп'ютерів і комп'ютерних систем від проникнення шкідливих програм і документів – вірусів, троянців, черв'яків; від використання неперевірених та сумнівного походження програм і документів;
- ліквідація недоліків - люків та дірок в системному та мережевому програмному забезпеченні;

- корпоративна організація адміністративних заходів для зберігання, обміну, знищення документів в паперовому та електронному вигляді та доступу до них в межах підрозділу і підприємства в цілому;
- апаратний захист від збоїв, втрати важливої інформації, краху ОС внаслідок відключення живлення, стрибків напруги, попадання блискавки, виходу з ладу компонентів та ін.;
- захист програм від декомпіляції і розшифрування алгоритмів (*режими компіляції, витончене програмування*);
- захист інформаційних систем від промислового шпигунства – попадання важливої ділової, фінансової і технічної документації в руки конкурентів, шантажистів та зловмисників;
- юридичні, політичні та соціальні питання, пов'язані з доступом і використанням інформації, її зберіганням та передачею по лініях зв'язку;
- забезпечення передачі важливої інформації в несприятливих умовах – нестійкий зв'язок, зашумлення каналів, у військовий час, в критичних ситуаціях та ін.;
- забезпечення надійного знищення конфіденційної інформації;
- питання безпеки в комп'ютерних мережах.

Лабораторна робота №8

Тема: Частотний аналіз файлів даних заданого типу

Мета роботи. Проведення частотного аналізу символів (байтів) у файлах даного типу, встановлення характеру та природи розподілу, підготовка вхідних даних для наступної роботи (“Кодування методами Фано та Хаффмена”)

Теоретичні відомості

Частотний аналіз використовується на різних етапах криптоаналізу, в евристичних алгоритмах розпізнавання відомих типів файлів (з невідомим чи зміненим розширенням імені), для автоматичного розпізнавання мови та таблиці кодування документів (зокрема Web-сторінок), в алгоритмах стискання (архівації), у форматах зберігання графічних та мультимедійних даних, в транспортних та криптографічних протоколах, в програмах здатних до самомодифікації (самонавчання), для автозаповнення полів, в системах захисту для ідентифікації користувачів за їх індивідуальним стилем (почерком) під час сеансів роботи в системах загального доступу та ін.

Порядок виконання роботи.

Завдання 1. Підготувати в окремих каталогах (папках) набір файлів одного типу (з однаковим розширенням імені) загальною довжиною не менше 2 мегабайти згідно з варіантом.

Список варіантів

№	Розши-рення	Тип файлів
1	***.doc***	нестиснуті файли, створені в редакторі Word однієї версії однією мовою однієї тематики;

2	***.ppt***	файли презентацій PowerPoint;
3	***.jpg***	графічні (фото) файли JPEG (Joint Picture Expert Group) ;
4	***.gif***	графічні файли з можливістю анімації (Graphic Interlaced Format) ;
5	***.pdf***	об'єднують зображення та текст (частково чи повністю розпізнаний);
6	***.djvu***	об'єднують стиснуті зображення та текст;
7	***.rtf***	форматовані текстові файли (rich text format);
8	***.pas***	текстові файли з програмами мовою Паскаль;
9	***.mid***	музичні файли (ноти), використовують таблиці синтезу інструментів;
10	***.mp3***	звукові та музичні файли (зменшені .wav);
11	***.xls***	нестиснуті файли, створені в табличному процесорі Excel однієї версії;
12	***.dbf***	файли баз даних FoxPro, Clipper, dBase.

Завдання 2. Частотний аналіз набору однотипних файлів можна проводити по одному з накопиченням результатів, або для спрощення алгоритму об'єднати послідовно групу файлів в один великий технічний робочий файл і провести аналіз одного великого файлу. Об'єднання файлів можна виконати за допомогою файлового менеджера або програмно. Зразок процедури, яка сканує послідовно файли ***.txt*** з каталогу ***\0*** на диску ***d:*** та об'єднує у файл ***work1.000.***

```
procedure TForm1.Button1Click(Sender: TObject);
var f1,f2: file of byte; b1:byte;   F:TSearchRec;
begin if FindFirst('d:\0\*.txt',faAnyFile,F)=0 then
   begin   ShowMessage('Wait.....');
     AssignFile(f1,'work1.000');     Rewrite(f1);
     AssignFile(f2,'d:\0\'+F.Name);  Reset(f2);
```

```
    while not eof(f2) do begin read(f2,b1);
                write(f1,b1); end; CloseFile(f2);
    while FindNext(F)=0 do
    begin  AssignFile(f2,'d:\0\'+F.Name); Reset(f2);
       while not eof(f2) do begin read(f2,b1);
          write(f1,b1) end; CloseFile(f2) end;
    CloseFile(f1); end;
 ShowMessage('Done, all files added in work1.000');end;
```

Завдання 3. Алгоритм найпростішого частотного аналізу полягає в послідовному перегляді байтів даних із файла та обчисленні кількості байтів із різними номерами в масиві ***Freq[0..255].*** Для подальшого використання обчислені кількості діляться на довжину файла та зберігаються в текстовому файлі. Зразок процедури, яка сканує послідовно файл ***work1.000*** та зберігає частоти символів (байтів) в текстовому файлі ***resuLt1.txt.***

```
procedure TForm1.Button2Click(Sender: TObject);
var f1:file of byte;  b1:byte;freq: array[0..255] of
integer;  Len,k:integer; f2:TextFile;  s:string;
Begin ShowMessage('Wait...');
   for k:=0 to 255 do freq[k]:=0;
   AssignFile(f1,'work1.000');Reset(f1);
   while not eof(f1) do begin read(f1,b1);  inc(len);
     inc(freq[b1]) end; CloseFile(f1);
   AssignFile(f2,'result1.txt'); Rewrite(f2);
   for k:=0 to 255 do
   begin Str(freq[k]/len:16:12,s); writeln(f2,s); end;
   CloseFile(f2);
   ShowMessage('Done, result1.txt is created') end;
```

Завдання 4. Проаналізувати характер розподілу частот для байтів у файлах заданого типу, вказати символи, які зустрічаються найчастіше та символи, що не зустрічаються жодного разу. Обчислити середнє значення, дисперсію та середньоквадратичне відхилення. Аналіз можна виконати за допомогою електронних таблиць, або математичного пакету, або програмно. Якщо файл ***resuLt1.txt*** містить нулі (тобто існують байти, що у файлах вказаного типу не зустрічаються жодного разу), вилучити ці дані з розгляду. Зразок процедури, яка за даними з файлу ***resuLt1.txt*** обчислює середнє значення, дисперсію та середньоквадратичне відхилення.

```
procedure TForm1.Button3Click(Sender: TObject);
var f2:TextFile; freq: array[0..255] of real;
    s:string;    k,n,code:integer;  r,M,V,SD:real;
begin  AssignFile(f2,'result1.txt');   Reset(f2);
   M:=0.0; n:=0;  for k:=0 to 255 do
   begin readln(f2,s);  val(s,r,code);   freq[k]:=r;
     if r<>0.0 then inc(n); M:=M+r; end; CloseFile(f2);
   if n<>0 then M:=M/n; Str(M:16:14,s); ShowMessage(
  'Середнє MeanValue ='+ s+'   n='+inttostr(n));
   V:=0.0; for k:=0 to 255 do
   begin r:=freq[k]; if r<>0.0 then V:=V+sqr(r-M); end;
   if n>1 then V:=V/(n-1); Str(V:16:14,s);
   ShowMessage('Дисперсія  Variance='+ s);
   SD:=sqrt(V); Str(SD:16:14,s); ShowMessage(
   'Середньоквадратичне відхилення StdDev='+s); end;
```

Зробити висновки про рівномірність розподілу різних байтів у файлах заданого типу.

Завдання 5. Для відповіді на питання, чи є знайдений розподіл частот байтів закономірним для файлів вказаного типу чи випадковим, **повторити пункти 1-4** для наступної групи файлів даного типу.

Завдання 6. Встановити зв'язок між рядами частот байтів в двох групах однотипних файлів.

Обчислити коефіцієнт кореляції між даними у файлах ***resuLt1.txt*** і ***resuLt2.txt*** можна за допомогою електронних таблиць, або математичного пакету, або програмно.

Зразок процедури, яка за даними файлів ***resuLt1.txt*** і ***resuLt2.txt*** обчислює коефіцієнт кореляції.

```
procedure TForm1.Button4Click(Sender: TObject);
var f1,f2:TextFile; freq: array[1..2,0..255] of real;
    s:string;k,code:integer; r,MX,MY,X,Y,XY,corel:real;
 begin  AssignFile(f1,'result1.txt');   Reset(f1);
   AssignFile(f2,'result2.txt');   Reset(f2);
   MX:=0.0;  MY:=0.0; for k:=0 to 255 do begin
   readln(f1,s);val(s,r,code);freq[1,k]:=r; MX:=MX+r;
  readln(f2,s);val(s,r,code);freq[2,k]:=r;MY:=MY+r end;
   CloseFile(f1);CloseFile(f2); MX:=MX/256; MY:=MY/256;
   Str(MX:16:14,s); ShowMessage('MeanValue X='+ s);
   Str(MY:16:14,s); ShowMessage('MeanValue Y='+ s);
   X:=0.0; Y:=0.0; XY:=0.0;for k:=0 to 255 do begin
      X:=X+sqr(freq[1,k]-MX); Y:=Y+sqr(freq[2,k]-MY);
     XY:=XY+(freq[1,k]-MX)*(freq[2,k]-MY);  end;
     corel:=XY/sqrt(X*Y); Str(corel:16:14,s);
ShowMessage('Коефіцієнт кореляції Corelation='+ s);end;
```

За величиною коефіцієнта кореляції зробити висновок про силу лінійного зв'язку двох рядів частот.

Завдання 7. Об'єднати знайдені частоти в один масив, відсортувати в порядку спадання разом з номерами відповідних байтів та зберегти у файлі ***res.txt*** для використання в наступній роботі (байти з нульовими частотами можна з подальшого розгляду вилучити).

Зразок процедури, яка сортує об'єднані дані файлів ***resuLt1.txt*** і ***resuLt2.txt*** та виводить в текстовому вікні в рядках в порядку спадання за частотою номер байта, частоту та відповідний символ ASCII (нечитабельні символи позначаються ***).

```
procedure TForm1.Button5Click(Sender: TObject);
var f1,f2:TextFile;  byte_array:array[0..255] of byte;
    freq: array[0..255] of real;
    s:string;    k,code,L,q:integer;  r:real;
procedure Swap(i,j:integer);
var t:real; b:byte;
begin  t:=freq[i]; b:=byte_array[i];
  freq[i]:=freq[j]; byte_array[i]:=byte_array[j];
  freq[j]:=t; byte_array[j]:=b;
end;
begin AssignFile(f1,'result1.txt');  Reset(f1);
   for k:=0 to 255 do begin
 byte_array[k]:=k;readln(f1,s);val(s,freq[k],code);end;
 CloseFile(f1);AssignFile(f2,'result2.txt'); Reset(f2);
  for k:=0 to 255 do begin
    readln(f2,s);val(s,r,code);freq[k]:=(freq[k]+r)/2;
   end; CloseFile(f2);
   for L:=255 downto 1 do for q:=0 to L-1 do
        if freq[q]<freq[q+1] then Swap(q,q+1);
   for k:=0 to 255 do  begin    str(freq[k]:15:12,s);
```

```
    s:=s+'#'+IntToStr(byte_array[k]);
    if byte_array[k]<32 then s:=s+' ***'
         else  s:=s+' '+ansichar(byte_array[k]);
    Memo1.Lines.Add(s);  end;
  AssignFile(f2,'res.txt');   Rewrite(f2);
  for k:=0 to 255 do
  begin str(byte_array[k],s);     writeln(f2,s);
    str(freq[k]:15:12,s);     writeln(f2,s);
  end; CloseFile(f2); end;
```

Завдання 8. Оформити головну форму програми для виклику всіх процедур в потрібному порядку та виведення в текстові поля результатів обчислень. Підготувати звіт роботи та підвести загальні підсумки. До звіту додається електронна копія всіх файлів проекту.

Контрольні запитання.

1. Як встановити силу лінійного зв'язку між двома випадковими величинами (рядами значень)?

2. Яке відношення до даної роботи має закон великих чисел? Чи можна відносні частоти символів (байтів) спів ставити з ймовірностями появи їх у файлах даного типу?

3. Чому для частотного аналізу текстових даних потрібно підбирати тексти однією мовою?

4. Чому для частотного аналізу текстових даних потрібно підбирати тексти створені з використанням однієї таблиці кодування?

5. Як перевірити рівномірність розподілу випадкової величини ?

6. Чи може коефіцієнт кореляції набувати від'ємних значень ?

7. Чому для частотного аналізу краще використовувати нестиснуті файли?

8. Як з допомогою даної роботи перевірити ефективність роботи програми-архіватора чи алгоритму стискання графічної інформації?

9. Як з допомогою даної роботи можна перевірити роботу програми шифрування файлів?

10. Як за допомогою даної роботи перевірити якість генератора псевдовипадкових чисел?

Лабораторна робота №9.

Тема: Кодування методами Фано та Хаффмена

Мета роботи. Вивчення основ кодування файлів на основі проведеного у попередній роботі частотного аналізу символів (байтів) у файлах заданого типу, дослідження та аналіз ефективних методів кодування Фано та Хаффмена.

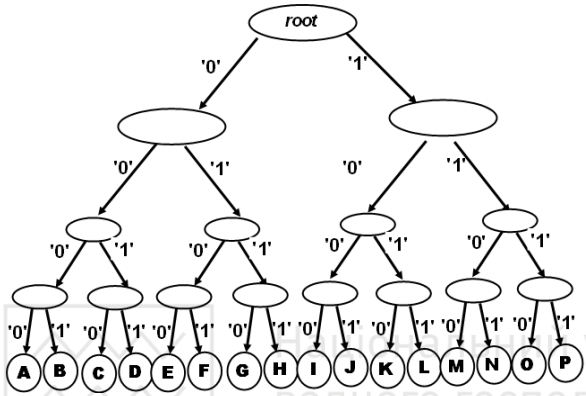
Теоретичні відомості

Завданням кодування є швидке, зручне і надійне передавання повідомлень по каналах зв'язку. Кодуючий пристрій співставляє кожному символу тексту, а іноді цілим словам і фразам певну комбінацію сигналів (прийнятну для передавання по даному каналу зв'язку), яку називають кодовим словом. Операцію переведення повідомлень в послідовність сигналів називають кодуванням, а зворотню – декодуванням.

У сучасних пристроях для зберігання та передачі інформації використовуються найпрстіші в реалізації двійкові та двійково-десяткові коди.

При передачі інформації розрізняють два рівні напруги сигналу, один з рівнів відповідає "1", інший – "0". Для зберігання інформації використовуються елементи з двома стійкими станами (тригери, конденсатори, магнітні домени та ін.), які відповідають "0" чи "1". Якщо повідомлення (символи) мають однакову імовірність появи чи частота їх появи не відома, то використовують двійкові коди однакової довжини. Довжина коду у випадку, якщо кількість повідомлень (символів) дорівнює степені 2, обчислюється за

формулою $L = \log_2 n$ і бінарне дерево кодів в цьому випадку повне , в інших випадках за формулою $L = \lceil \log_2 n \rceil + 1$. Приклад кодування двійковими числами при *n=16* наведено на **рис.1**.



Повідом-лення (символ)	Коди	довжина коду
A	0000	4
B	0001	4
C	0010	4
D	0011	4
E	0100	4
F	0101	4
G	0110	4
H	0111	4
I	1000	4
J	1001	4
K	1010	4
L	1011	4
M	1100	4
N	1101	4
O	1110	4
P	1111	4

Рис1. Дерево та таблиця двійкових кодів однакової довжини при ***n=16***

Код називається однозначно декодованим (або кодом без коми), якщо будь-яка послідовність символів (різної довжини) може бути розбита на кодові слова. Прикладом є префіксні коди, які мають таку основну властивість: жодне кодове слово не є початком (префіксом) іншого кодового слова. Префіксний код називається повним, якщо додавання до нього будь-якого кодового слова порушує його префіксність. Статичний алгоритм кодування двопрохідний: під час першого перегляду обчислюються частоти символів (див. попередню роботу), на другому етапі будуються таблиці (або дерево) кодів. Адаптивний алгоритм кодування використовує змінну таблицю частот, яка в процесі аналізу неперервного чи тривалого потоку вхідних даних відповідно змінює таблицю кодів. На початку роботи використовується типова для файлів данного типу таблиця частот.

Код Фано. Ідея економного кодуваня Р.М.Фано (R.M.Fano) полягає в аналізі вхідного потоку даних та побудові на основі частот символів економного дерева кодів. Схожий алгоритм був запропонований також К.Шеноном (Claude Shannon), тому в літературі алгоритм часто називають кодуванням Шенона-Фано.

Повідомлення (символ)	Відносна частота
A	0,24
B	0,13
C	0,11
D	0,09
E	0,08
F	0,07
G	0,05
H	0,04
I	0,04
J	0,03
K	0,03
L	0,03
M	0,02
N	0,02
O	0,01
P	0,01

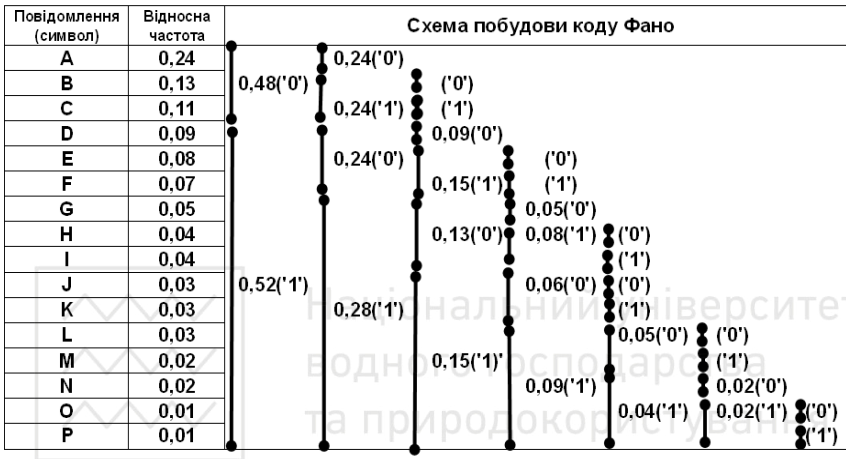


Рис 2. Приклад розбиття множини повідомлень та побудови таблиці кодів Фано

Алгоритм побудови кодів Фано.

1) Частоти символів відповідного алфавіту впорядковуються (на практиці найчастіше реалізується побайтовий алгоритм з символами від #0 до #255). Порядок сортування в даному алгоритмі не принциповий, хоч при цьому коди змінюються на протилежні. Разом з частотами потрібно зберегти і відповідні їм коди (для визначеності сортування виконане в порядку спадання частот). Відповідні дані розміщені у файлі ***res.txt*** з попередньої роботи.

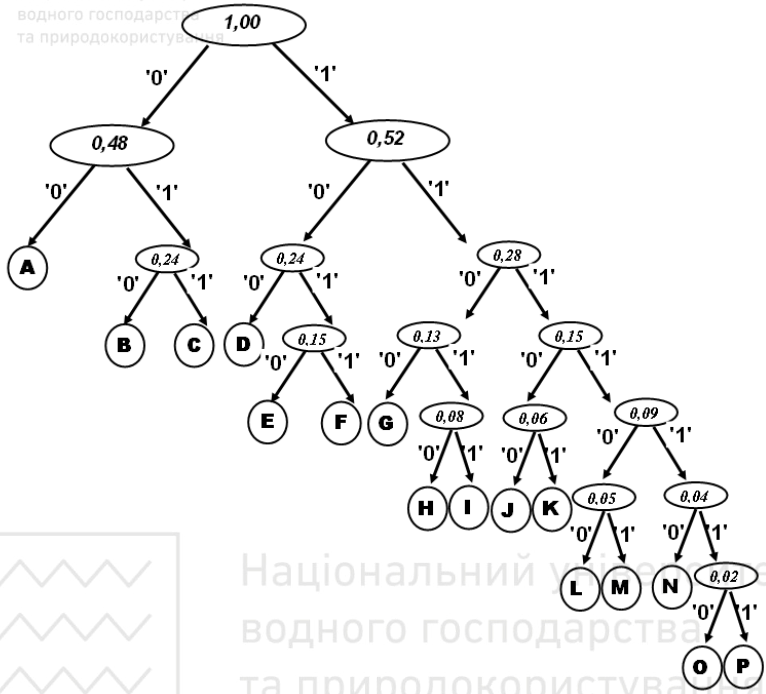
Далі, спочатку для всього масиву елементів, а потім для частин, на які він розбивається, повторюються такі кроки (поки не залишиться два елемента):

1) масив частот розбивається на дві послідовні частини, таким чином щоб суми частот в обох частинах точно чи наближено співпадали;

2) до кодів першої частини дописується символ ”0”, до кодів другої –“1”;

4) якщо залишається два символи, до кодів першого дописується символ ”0”, до кодів другого –“1”.

Властивості кодів зручно перевіряти чи демонструвати з допомогою відповідного бінарного дерева. Якщо властивість префіксності не виконується, то на графі проміжні вершини можуть відповідати кодовим словам. Для коду Фано це неможливо, бо побудова кодового слова закінчується одночасно із досягненням кінцевої вершини (рис.3). Отже, код Фано є префіксним.



Повідомлення (символ)	Відносна частота	Коди Фано	довжина коду	F*L
A	0,24	00	2	0,48
B	0,13	010	3	0,39
C	0,11	011	3	0,33
D	0,09	100	3	0,27
E	0,08	1010	4	0,32
F	0,07	1011	4	0,28
G	0,05	1100	4	0,2
H	0,04	11010	5	0,2
I	0,04	11011	5	0,2
J	0,03	11100	5	0,15
K	0,03	11101	5	0,15
L	0,03	111100	6	0,18
M	0,02	111101	6	0,12
N	0,02	111110	6	0,12
O	0,01	1111110	7	0,07
P	0,01	1111111	7	0,07
			сума=	3,53

Рис 3. Дерево та таблиця кодів Фано з прикладу на рис.2.

Програмно код Фано реалізується у вигляді рекурсивної процедури:

```
procedure Fano(high,low:integer);
var k,median:integer; sum,half_sum:real;
begin if low=high+1 then
  begin code1[high]:=code1[high]+'0';
        code1[low]:=code1[low]+'1' end
  else
  begin
   if freq[high]=freq[low] then median:=(low+high)div 2
   else begin  sum:=0.0;
    for k:=high to low do sum:=sum+freq[k];
    half_sum:=sum/2; sum:=0.0; k:=high;
    while sum<half_sum  do
    begin sum:=sum+freq[k];inc(k) end;
    if abs(sum-half_sum)>abs(sum-freq[k-1]-half_sum)
    then median:=k-2  else median:=k-1; end;
    for k:=high to median do code1[k]:=code1[k]+'0';
    if high<median then Fano(high,median);
    for k:=median+1 to low do code1[k]:=code1[k]+'1';
    if median+1<low then Fano(median+1,low);
  end; end;
```

Код Фано не завжди дає найбільше стискання розмірів кодованих файлів. Відповідь про властивості найекономнішого методу кодування та алгоритм його побудови дав в 1952 році американський математик Девід Хафмен. [Huffman D.A. A Method for the Construction Minimum-Redunancy Codes, Proc. IRE, 40 , N 9, 1952, p.1098].

Код Хаффмена. Мінімальним (мінімально надлишковим) називається код, який при даному наборі частот дає мінімальне значення ефективної довжини кодів.

Властивості

1) Повідомлення (символи) з більшою імовірністю появи в потоці не можуть мати коди довші, ніж менш імовірні повідомлення.

$$\text{Якщо } p_k > p_m \text{ , то } l_k \le l_m$$

2) Два найменш ймовірні повідомлення мають коди однакової довжини, які відрізняються останнім бітом.

Алгоритм побудови кодів Хаффмена

1) Частоти символів відповідного алфавіту впорядковуються в порядку спадання частот. Відповідні дані розміщені у файлі ***res.txt*** з попередньої роботи. Таблиця кодів спочатку заповнюється пустими кодами.

Далі повторюються такі кроки (поки не залишаться дві частоти):

2) Розглядаються 2 групи символів з найменшими частотами. До початку кодів групи символів, яка відповідає передостанній частоті, дописується ‘0’, до початку кодів групи символів, яка відповідає останній частоті, дописується ‘1’.

3) Дві останні частоти додаються і їх сума вставляється в масив частот, так щоб масив був відсортований як раніше за спаданням. Якщо довжина масиву частот більша 2 – перехід до п.2.

4) якщо залишається дві групи символів, до кодів першої дописується попереду символ ”0”, до кодів другої –“1”.

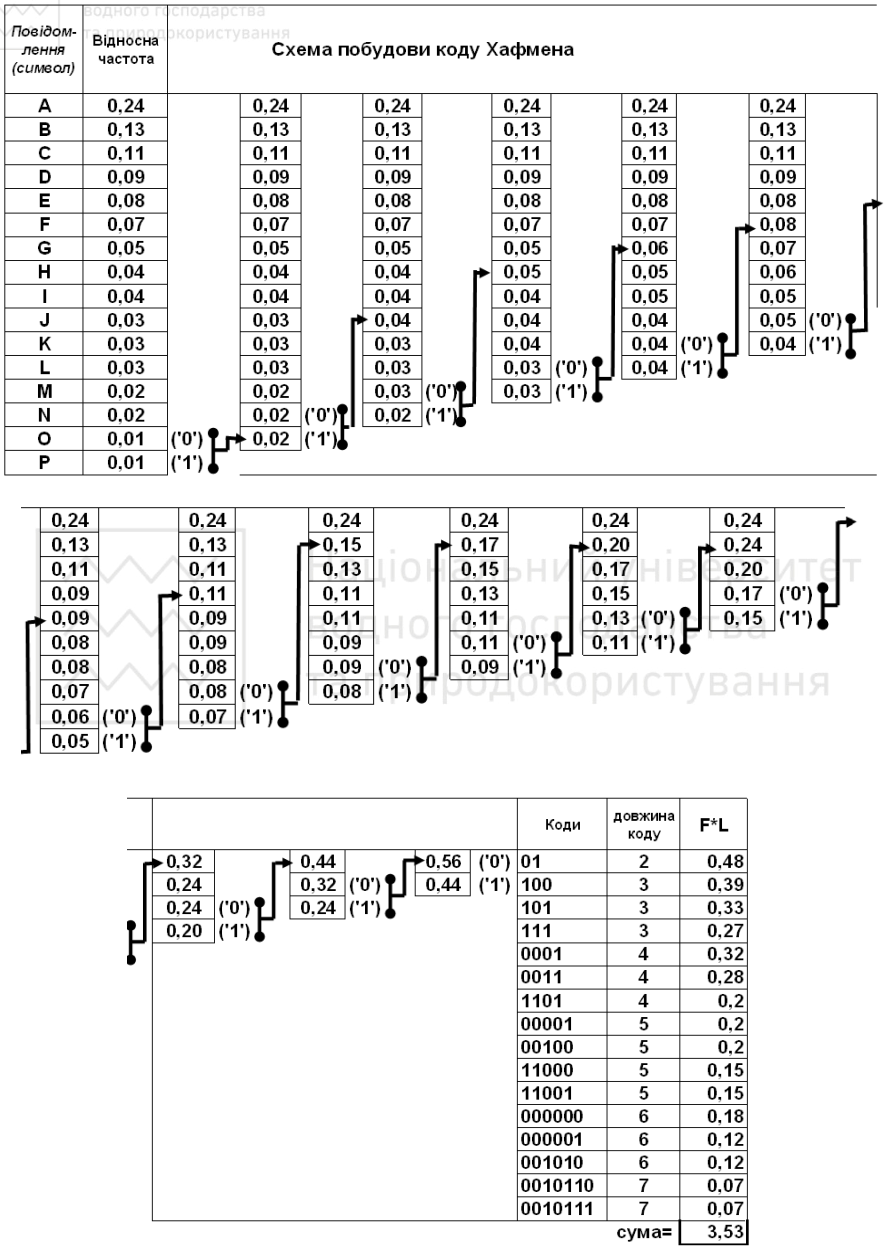


Рис. 4. Приклад побудови таблиці кодів Хафмена

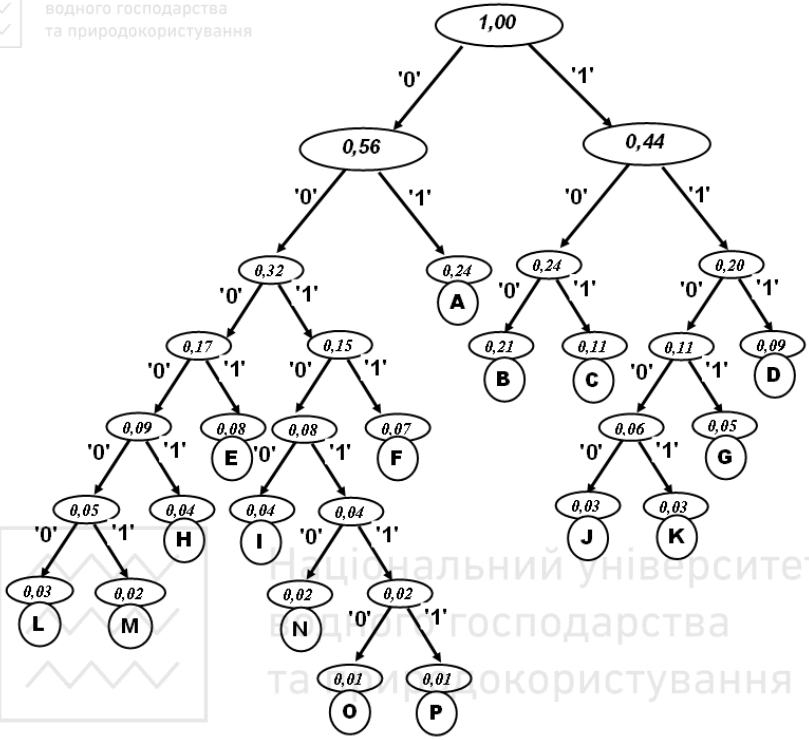


Рис 5. Дерево кодів Хафмена з прикладу на рис.4

Процедура для побудови коду Хафмена (коди обчислюються без явної побудови бінарного дерева і без зворотнього ходу) має вигляд:

```
procedure Huffman;
var len2,k,k1,i:integer; sum2:real;
    index,index1:array[0..255] of byte;
begin
  for k:=0 to 255 do
  begin  code2[k]:=''; index[k]:=k  end;
  freq2:=freq;   index1:=index;
  for len2:=255 downto 1 do
```

```
  begin
    for k:=0 to 255 do
    begin
      if index[k]=len2-1 then code2[k]:='0'+code2[k];
      if index[k]=len2   then code2[k]:='1'+code2[k];
    end;
    if len2>1 then
    begin
      sum2:=freq2[len2-1]+freq2[len2];   k:=len2-1;
      while (k>0)and(freq2[k-1]<sum2) do
      begin freq2[k]:=freq2[k-1];dec(k); end;
      freq2[k]:=sum2;
      for i:=0 to 255 do
         if (index[i]=len2)or(index[i]=len2-1) then
                              index1[i]:=k ;
      if k<>len2-1 then
       for k1:=k to len2-2 do
         for i:=0 to 255 do
           if index[i]=k1 then index1[i]:=index[i]+1;
      index:=index1;
    end
  end;
end;
```

Порядок виконання роботи

В якості вхідних даних використовується файл ***res.txt*** з попередньої роботи, в якому для заданих згідно з варіантом типів файлів збережено імовірності (частоти)появи байтів, відсортовані в порядку спадання разом з номерами байтів.

Завдання 1. Розмістити на головній формі компоненти ***Memo1, OpenDialog1*** та відповідну кількість кнопок ***Button***. Побудувати на основі даних з файлу ***res.txt*** таблицю кодів Фано для файлів заданого типу та зберегти у файлі ***code1.txt.***

Зразок процедури для побудови кодів Фано.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  file_text:TextFile; byte_array:array[0..255] of byte;
  freq: array[0..255] of real;
  code1:array[0..255] of string;
  s:string; k,code:integer; len1:real;
procedure Fano(high,low:integer);
          {наведена вище}
begin
   AssignFile(file_text,'res.txt'); Reset(file_text);
   for k:=0 to 255 do
   begin
     readln(file_text,s);val(s,byte_array[k],code);
     readln(file_text,s); val(s,freq[k],code);
   end;
   CloseFile(file_text);
   for k:=0 to 255 do code1[k]:='';    Fano(0,255);
   // обчислення середньої довжини при кодуванні файла
довжиною 1000000 байтів
   len1:=0.0;
   for k:=0 to 255 do
       len1:=len1+length(code1[k])*freq[k];
   len1:=len1*1000000/8; Str(len1:10:0,s);
```

```
   Memo1.Lines.Add('ефективна довжина при кодуванні
файла обсягом 1 Мбайт='+ s+' байтів');
   // запис номерів байтів та їх кодів
   // у файл в порядку спадання частот
  AssignFile(file_text,'code1.txt');Rewrite(file_text);
   for k:=0 to 255 do
   begin
     str(byte_array[k],s); writeln(file_text,s);
     writeln(file_text,code1[k]);
   end;
   CloseFile(file_text);Memo1.Lines.Add(
   'Таблиця кодів Фано збережена у файлі code1.txt');
end;
```

Завдання 2. Побудувати основі даних з файлу ***res.txt*** таблицю кодів Хаффмена для файлів заданого типу та зберегти у файлі ***code2.txt.***

Зразок процедури для побудови кодів Хаффмена.

```
procedure TForm1.Button2Click(Sender: TObject);
var f1,f2:TextFile;
    byte_array:array[0..255] of byte;
    code2:array[0..255] of string;
    freq,freq2: array[0..255] of real;
    s:string;    k,code:integer;   len1:real;
procedure Huffman;
           {наведена вище}
begin
   AssignFile(f1,'res.txt');   Reset(f1);
   for k:=0 to 255 do
   begin
```

```
      readln(f1,s); val(s,byte_array[k],code);
      readln(f1,s); val(s,freq[k],code);
   end;
   CloseFile(f1);   Huffman;
  // обчислення середньої довжини при
  // кодуванні файла довжиною 1000000 байтів
   len1:=0.0;
   for k:=0 to 255 do
                len1:=len1+length(code2[k])*freq[k];
   len1:=len1*1000000/8;   Str(len1:10:0,s);
   Memo1.Lines.Add('ефективна  довжина  при  кодуванні
файла обсягом 1 Мбайт='+ s+' байтів');
  // запис номерів байтів та їх кодів
  // у файл в порядку спадання частот
   AssignFile(f2,'code2.txt');  Rewrite(f2);
   for k:=0 to 255 do
   begin str(byte_array[k],s); writeln(f2,s);
         writeln(f2,code2[k]) end;
   CloseFile(f2); Memo1.Lines.Add(
'Таблиця кодів Хаффмена збережена у файлі code2.txt');
end;
```

Завдання 3. Закодувати один із файлів заданого типу з використанням створених таблиць кодів. Порівняти довжину оригінального та закодованого файлів. Оцінити ступінь стискання.

Для спрощення алгоритму кодування відбувається в два етапи: спочатку створюється файл з символьних нулів і одиниць, а потім символи порціями по 8 перетворюються в байти. В останньому байті

закодованого файлу вказується довжина останньої порції для однозначності наступного декодування.

Зразок процедури 1 для кодування файлів

```
procedure TForm1.Button3Click(Sender: TObject);
var  file_text:TextFile; file_byte:file of byte;
     file_char:file of char;
    codetable:array[0..255] of string;
    byte1,k:byte; char1:char; code,s,s1:string;
begin // читання номерів байтів та їх кодів
  //  з файла code1.txt в порядку спадання частот
   Memo1.Lines.Add('Задайте назву файла з кодами
                   (code1.txt чи code2.txt) ');
   OpenDialog1.Execute;
AssignFile(file_text,OpenDialog1.Filename);
Reset(file_text);
   for k:=0 to 255 do
   begin
     readln(file_text,s);readln(file_text,s1);
    // запам'ятаємо в масиві в порядку кодів
     codetable[StrToInt(s)]:=s1;
   end;
   CloseFile(file_text);
Memo1.Lines.Add('Задайте назву файла для кодування
               (з розширенням відповідно до варіанту)');
   OpenDialog1.Execute;
   AssignFile(file_byte,OpenDialog1.Filename);
   Reset(file_byte);
   s:=InputBox('(test1.chr чи test2.chr)','Задайте
    назву для проміжного двійково-символьного
```

```
    файла','test1.chr');
   AssignFile(file_char,s);  Rewrite(file_char);
   while not eof(file_byte) do
   begin
      read(file_byte,byte1); code:= codetable[byte1];
      for k:=1 to length(code) do
      begin char1:=code[k];write(file_char,char1); end
   end;
   CloseFile(file_byte);CloseFile(file_char);
   Memo1.Lines.Add
   ('Двійково-символьний файл '+s+' створено');
end;
```

Зразок процедури 2 для кодування файлів

```
procedure TForm1.Button4Click(Sender: TObject);
var
     f_char:file of char; f_byte:file of byte;
     L1,byte1,k:byte; s,ss:string; char1:char;

   function BynaryStringToByte(s:string):byte;
   var i,power2,byte1:byte;
   begin
     byte1:=0; power2:=1;
     for i:=8 downto 1 do
     begin if s[i]='1' then byte1:=byte1+power2;
                            power2:=power2*2;end;
     BynaryStringToByte:=byte1;
   end;
begin
```

```
   Memo1.Lines.Add('Задайте назву для проміжного
двійково-символьного файла (test1.chr чи test2.chr)');
   OpenDialog1.Execute;
   AssignFile(f_char,OpenDialog1.Filename);
   Reset(f_char);
   s:=InputBox('(test1.cod чи test2.cod)','Задайте
   назву для закодованого файла','test1.cod');
   AssignFile(f_byte,s); Rewrite(f_byte);  ss:='';
   while not eof(f_char) do
   begin    read(f_char,char1); ss:=ss+char1;
     if length(ss)=8 then
     begin byte1:=BynaryStringToByte(ss);
           write(f_byte,byte1); ss:=''; end;
   end;
   byte1:=0;
   if ss='' then write(f_byte,byte1) else
   begin
     L1:=length(ss); for k:=L1+1 to 8 do ss:=ss+'1';
     byte1:=BynaryStringToByte(ss);
     write(f_byte,byte1); write(f_byte,L1);
   end;
   CloseFile(f_char);CloseFile(f_byte);
   Memo1.Lines.Add
  ('Кінцевий закодований файл '+s+' створено');
end;
```

Завдання 4. Декодувати закодовані файли та порівняти з оригінальним.

Зразок процедури для декодування файлів

```
procedure TForm1.Button5Click(Sender: TObject);
var  file_text:TextFile;file_byte:file of byte;
     file_char:file of char;IsFind:Boolean;
     codetable:array[0..255] of string;
     k,minL,maxL:byte; char1:char;  s,s1:string;
begin // читання номерів байтів та їх кодів
      // з файла code1.txt в порядку спадання частот
   Memo1.Lines.Add('Виберіть  файл  з  кодами  (code1.txt
чи code2.txt)');
   OpenDialog1.Execute;
   AssignFile(file_text,OpenDialog1.Filename);
   Reset(file_text); minL:=100;maxL:=0;
   for k:=0 to 255 do
   begin
     readln(file_text,s);readln(file_text,s1);
     if length(s1)>maxL then maxL:=length(s1);
     if length(s1)<minL then minL:=length(s1);
     // запам'ятаємо в масиві в порядку кодів
     codetable[StrToInt(s)]:=s1;
   end;
   CloseFile(file_text); Memo1.Lines.Add('Виберіть
   назву закодованого двійково-символьного файла
     (test1.chr чи test2.chr)');
   OpenDialog1.Execute;
   AssignFile(file_char,OpenDialog1.Filename);
   Reset(file_char);
   s:=InputBox('(test1.dcd чи test2.dcd)','Задайте
         назву для розкодованого файла','test1.dcd');
   AssignFile(file_byte,s); Rewrite(file_byte);
```

```
    while not eof(file_char) do
   begin
      s:=''; IsFind:=False;
      for k:=1 to minL do
      begin read(file_char,char1);s:=s+char1;end;
      repeat
        k:=0;
        repeat if codetable[k]=s then begin
         IsFind:=True; break end;
         inc(k)
        until k=0;
        if IsFind then break;
        if not eof(file_char) then
        begin read(file_char,char1);s:=s+char1;end;
      until  length(s)>maxL;
      if IsFind then write(file_byte,k) else
              Memo1.Lines.Add('Помилка декодування')
   end;
  CloseFile(file_byte);CloseFile(file_char);
  Memo1.Lines.Add('Файл розкодовано')
end;
```

Зразок процедури для порівняння файлів

```
procedure TForm1.Button6Click(Sender: TObject);
var f1,f2:file of byte;b1,b2:byte;L1,L2,diffr:integer;
begin
  Memo1.Lines.Add('Виберіть оригінальний файл для
порівняння');
  OpenDialog1.Execute;
  AssignFile(f1,OpenDialog1.Filename);Reset(f1);
```

```
  Memo1.Lines.Add('Виберіть закодований файл для
порівняння');
  OpenDialog1.Execute;
  AssignFile(f2,OpenDialog1.Filename);Reset(f2);
  L1:=0;L2:=0;diffr:=0;
  while (not(eof(f1)))and(not(eof(f1))) do
  begin
    read(f1,b1);inc(L1); read(f2,b2);inc(L2);
    if b1<>b2 then inc(diffr);  end;
  while not(eof(f1)) do begin read(f1,b1);inc(L1); end;
  while not(eof(f2)) do begin read(f2,b2);inc(L2); end;
  CloseFile(f1);   CloseFile(f2);
  if L1=L2 then Memo1.Lines.Add('Довжини рівні')
       else Memo1.Lines.Add('L1-L2='+IntTostr(L1-L2));
  if diffr=0 then Memo1.Lines.Add('Файли співпадають')
  else Memo1.Lines.Add('Не співпадають
                '+IntTostr(diffr) +'байтів');end;
```

Завдання 8. Оформити головну форму програми для виклику всіх процедур в потрібному порядку та виведення в текстові поля результатів обчислень. Підготувати звіт роботи та підвести загальні підсумки. До звіту додається електронна копія всіх файлів проекту.

Контрольні запитання.

1. Як перевірити префіксність коду?

2. Як перевірити повноту коду?

3. Чим адаптивний алгоритм стискання (кодування) відрізняється від статичного?

4. Які типи файлів використовуються в роботі?

5. Який результат кодування методом Фано алфавіту із двох символів?

6. Який результат кодування методом Хафмена алфавіту із двох символів?

7. Як найефективніше передавати (зберігати) таблицю кодів Хафмена?

8. В яких випадках використовується статичний алгоритм кодування (стискання)?

9. В яких випадках використовується адаптивний алгоритм кодування (стискання)?

10. В яких графічних форматах використовується алгоритм Хафмена?

11. Яка максимально можлива висота бінарного дерева (довжина коду) в алгоритмі кодування Фано?

12. Яка максимально можлива висота бінарного дерева (довжина коду) в алгоритмі кодування Хафмена?

13. Назвіть основні типи алгоритмів кодування.

14. Які типи файлів Turbo Pascal та Object Pascal Ви знаєте?

15. Назвіть стандартні процедури для роботи з файлами.

16. Чому запропоновані алгоритми кодують файли будь-якого типу?

17. Які стандартні функції Object Pascal використовуються для обчислення середньої швидкості кодування?

18. Запропонуйте вдосконалений варіант реалізованого в роботі алгоритму для прискорення кодування.

Література

О.В.Вербицький. Вступ до криптології.- 240с, Видавництво науково-технічної літератури, Львів,1998.

К.Шеннон. Работы по теории информации и кибернетике, - Изд. иностр. лит., М., 1963.

Н.Смарт. Криптография, -М.,Техносфера, 2005

В.А.Мухачев, В.А.Хорошко. Методы практической криптогафии. - К., ПолиграфКонсалтинг, 2005.

Нильс Фергюсон, Брюс Шнайер, Практическая криптография.- М, Диалектика, 2005.

М.Н.Арщинов, Л.Е.Садовский. Коды и математика.- М, Наука,1983.

Методичні вказівки

100-64 *Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Основи захисту та кодування інформації" студентами спеціальностей 7.080200, 8.080200 "Прикладна математика". Частина І./* ***П.В. Ольшанський****, - Рівне: УДУВГП, 2004, -52 с.*

100-65 *Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Основи захисту та кодування інформації" студентами спеціальностей 7.080200, 8.080200 "Прикладна математика". Частина ІІ./* ***П. В. Ольшанський****, - Рівне: УДУВГП, 2004, -60 с.*

04-04-190 *Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Захист інформації в комп'ютерних системах" студентами напряму підготовки 6.050102 "Комп'ютерна інженерія". Частина І. /* ***П. В. Ольшанський****, – Рівне: НУВГП, 2017, – 44 с.*

04-04-191 *Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Захист інформації в комп'ютерних системах" студентами напряму підготовки 6.050102 "Комп'ютерна інженерія". Частина ІІ. /* ***П. В. Ольшанський****, – Рівне: НУВГП, 2017, – 32 с.*

04-04-192 *Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Захист інформації в комп'ютерних системах" студентами напряму підготовки 6.050102 "Комп'ютерна інженерія". Частина ІІІ. /* ***П. В. Ольшанський****, – Рівне: НУВГП, 2017, – 28 с.*

04-04-193 *Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Технології захисту інформації" студентами напряму підготовки 6.050101 "Комп'ютерні науки". Частина І. / П. В. Ольшанський, – Рівне: НУВГП, 2017, – 44 с.*

04-04-194 *Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Технології захисту інформації" студентами напряму підготовки 6.050101 "Комп'ютерні науки". Частина ІІ. /* ***П. В. Ольшанський****, – Рівне: НУВГП, 2017, – 32 с.*

04-04-195 *Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Технології захисту інформації" студентами напряму підготовки 6.050101 "Комп'ютерні науки". Частина ІІІ. /* ***П. В. Ольшанський****, – Рівне: НУВГП, 2017, – 28 с.*

04-04-204 *Методичні вказівки для практично-семінарських занять та самостійної роботи з дисципліни "Захист інформації" студентами спеціальності 029 "Інформаційна, бібліотечна та архівна справа (Документознавство та інформаційна діяльність)". Частина І. Боротьба з комп'ютерними вірусами. /* ***П. В. Ольшанський****, – Рівне: НУВГП, 2017, – 36 с.*