تطوير خوارزمية لضغط الملفات النصية

نهلة عباس فليح

كلية العلوم-جامعة البصرة

الخلاصة :أن ضغط البيانات بعد حقلاً مهماً جداً في تطبيقات الحاسبات المختلفة ، بالاضافة الى كونه يقال من المساحة الخزنية التي تحتاجها البيانات للخزن على القرص فانه يقلل من حجم الحزمة المطلوبة النقل البيانات عبر شبكات الاتصالات ومن Digital Network . الرقمية ثم تقليل الوقت اللازم لنقل البيانات كما انه يقلل من حصول الاخطاء اثناء عملية النقل.

وبالنظر لأهمية الملفات النصية Text Files فقد تطرقت في هذا البحث الى موضوع ضغط الملفات النصية واستخدام خوارزمية الضغط المسماة خوارزمية LZW بعد تطويرها وربطها مع طريقة أخرى مقترحة الضغط انواع مختلفة من الملفات النصية .

اشتملت الخوارزمية المقترحة على برنامجين الأول يمثل عملية الضغط والثاني يمثل عملية فك الضغط

عملية الضغط تشمل مرحلتين :

1- استخدام خوارزمية LZW للحصول على ملف اقل حجم من الملف الاصلي يحتوي على شفرات للحروف المضغوطة.

٢ .استخدام خوارزمية الدمج المقترحة للتخلص من الثنائيات غير الضرورية في الملف الناتج من المرحلة الأولى اما برنامج فك الضغط فيعمل عكس عمل البرنامج الأول حيث يقوم أولا بفك الضغط من عملية الدمج ثم فك الضغط من خوارزمية LZW .

ان الخوارزمية المفترحة تعتبر الخوارزميات اللافقدية أي ان الملف الناتج من برنامج فك الضغط يكون مماثل للملف الاصلي بدون أي ضياع في البيانات

وقد تم تطبيق هذه الخوارزمية على مجموعة من الملفات النصية المختلفة اعتمادا على حجم الملف وطبيعة البيانات الموجودة فيه.

(1) ضغط البيانات Data compression :

يقصد بضغط البيانات تحويل البيانات المدخلة إلى بيانات أخرى ذات حجم اقل لتأخذ اقل مساحة خزنيه ممكنة مع الاحتفاظ بالمعلومات الضرورية لاعادتها إلى صيغتها الاصليه مرة أخرى. [1.2].

يدعى ملف الإدخال قبل عمليه الضغط بالملف الحقيقي (غير المضغوط (Uncompressed file بينما يدعى الملف بعد إنجاز عملية الضغط علية بالملف المضغوط (Compressed file) والذي يستخدم لاعادة البيانات إلى وضعها الأصلي مرة أخرى. وتستخدم عملية الضغط في مجالين [1.4]:

(١ – ١) مجال الخزن:

ازدادت الحاجة إلى هذه العملية في مجال الخزن بسبب ثورة المعلومات الهائلة وكثرة البيانات وقلة سعة الذاكرة القليلة محدوديتها، وبهدف الاحتفاظ بأكبر كمية ممكنة من البيانات مع السعة المحدودة للذاكرة يتم ضغط البيانات وخزنها بأقل مساحة ممكنة

(1-2) مجال النقل :

ازدادت الحاجة إلى مثل هذا الموضوع نتيجة النمو السريع والهائل في مجال الاتصالات والنقل وظهور شبكة المعلومات العالمية (شبكة الانتريت) حيث أن نقل كمية كبيرة من البيانات يتطلب قناة إرسال ذات حزمة واسعة وفترة زمنية طويلة .

ولهدف تقليل عرض الحزمة وتقليل زمن الإرسال يستخدم ضغط البيانات لتقليل حجم البيانات بدلا من إرسالها بوضعها الحالي.

2- ستراتيجيات ضغط البيانات (Data Compression Strategies)

يظهر الجدول (١) أدناه طريقتين مختلفتين يمكن تصنيف خوارزميات الضغط على أساسهما . حيث يمثل الجدول (١) (ا) تصنيف طرق الضغط الى فقديه Lossy أو لا فقدية Lossless.

تقنية ضغط البيانات اللافقدية Lossless تعني أن ملف البيانات الناتج من عملية حل الشفرة يكون نسخة طبق الأصل من ملف البيانات الأصلي وهذه التقنية ضرورية بشكل مطلق للعديد من أنواع البيانات فمثلا الشفرة القابلة للتنفيذ(الملفات التنفيذية Executable files) وملفات معالجات النصوص (Text files) والأعداد المجدولة وخزن الملفات لأغراض الأرشفة .. الخ . حيث لا يمكن تضييع أي ثنائية من هذا النوع من البيانات ، وبالمقارنة فأن ملفات البيانات التي تمثل الصور أو الإشارات الصوتية من الممكن أن لا تبقى بكامل بياناتها عند النقل أو الخزن حيث أن طريقة حفظ البيانات الفقدية قد تسبب مقدار غير ضار من التشويش الإضافي في الإشارات بعد عملية فك الضغط ولكنها تعتبر أكثر فعالية من الطرق اللافقدية بسبب نسبة الضغط العالية التي توفرها.

الطريقة الأخرى لتصنيف طرق ضغط البيانات توضح في الجدول (١) (ب) فأن معظم برامج الضغط تعمل بأخذ مجموعة من البيانات من الملف الأصلي وضغطها بواسطة برنامج الضغط ومن ثم كتابة المجموعة المضغوطة على ملف الإخراج ، على سبيل المثال إحدى الطرق المذكورة في الجدول طريقة CS&Q وهي مختصر طريقة Coarser Sampling and Quantization توضح على أنها طريقة ثابتة الإدخال والإخراج لذلك فأن عدد ثابت من الثنائيات يتم قراءتها في ملف الإدخال وعدد ثابت أصغر يتم كتابته على ملف الإخراج وهناك نوع آخر من التقنيات تسمح بقراءة عدد متغير من الثنائيات وكتابته على ملف الإخراج [13]

الطرق التقدية Lossy	الطرق اللاتقدية Lossles
CS&Q JPEGF MPEG	Run-length Huffman Delta LZW

الطريقة Method	حجم المجموعة Group size	
	الإدخال	
CS&O	ثابت	ثابت
Huffman	متغير	ثابت
Arithmetic Run-length ,LZW	متغير ثابت	متغير متغير

جدول (١) طرق تصنيف خوارزميات الضغط

الطرق الشائعة لضغط البيانات :

(3-1) طريقة تشفير طول السلسلة Run Length Ecoding (RLE)

بعض ملفات البيانات تحتوي أحيانا نفس البيانات مكررة عدة مرات فمثلا الملفات النصية (Text file تستخدم المجالات المتعددة (الفراغات) للجمل المنفصلة والفقرات البعيدة عن بداية السطر وجداول التنسيق والمخططات وغيرها

.... كما يمكن معالجة الإشارات المشفرة رقميا التي لها نفس القيمة المكررة فمثلا صورة السماء ليلا التي تحتوي الرمز المكرر الذي يمثل الخلفية الغامقة السواد أو الإشارات الصوتية التي تحتوي على وقفات بين الفقرات، ويمكن ضغط مثل هذه الأنواع باستخدام طريقة طول السلسلة التي يمكنها التخلص من هذه التكرارات [1.2)

(3-2) طريقة هوفمان : Hoffman coding

طور هذه الطريقة العالم هوفمان عام ١٩٥٢ وتتضمن هذه الطريقة تحليل أولي للبيانات لتحديد احتمالية ظهور كل عنصر فيتم تشفير البيانات الأعلى احتمالية بأقل عدد ممكن من الثنائيات والأقل احتمالية بعدد أكبر من الثنائيات وعند دمج الثنائيات يتم فصل كل رمز عن الرمز الآخر بوضع فواصل بين الثنائيات الممثلة للشفرات وهي إحدى الطرق ذات النوع القاموسي أي يتم تكوين قاموس أولي للشفرات ومن ثم أجراء عملية الضغط [1,2,3].

(٣-٣) طريقة LZW

قد تضم بعض أنواع الملفات المختلفة تكراراً من البيانات الغير متجاور وللتخلص من هذا التكرار بهدف تقليص حجم الملف تستخدم خوارزمية LZW ذات الإمكانية العالية في تقليل حجم الخزن للبيانات المتكررة المتجاورة وغير المتجاورة وقد استخدمت هذه التقنية في ضغط الملفات النصية والملفات الصورية وفي هذا البحث تم استخدام هذه الطريقة الضغط الملفات النصية Text files ذات الأحجام المختلفة [15]

(٣-٣-١) آلية LZW

(٣-٣-١-١) الضغط بطريقة LZW

تكون آلية عمل الضغط بطريقة LZW كالتالي :

يدخل المشفر الرموز الواحد بعد الآخر ويجمعهم في الخيط 1 ، بعد إدخال كل رمز ودمجه مع 1 يتم البحث في القاموس عن الخيط 1 ، طالما نجد 1 في القاموس فالمعالجة تستمر ، في مرحلة معينة إضافة الرمز التالي يسبب فشل عملية البحث ، أي إن الخيط 1 موجود في القاموس لكن الخيط IX (دمج 1 مع X) ليس موجودا في تلك اللحظة يقوم المشفر بالتالي :

1. يخرج مؤشر القاموس الذي يشير إلى 1 .

٢. يخزن الخيط IX) الذي يدعى ألان phrase) في مدخل القاموس المعرف التالي.

3.ابتداء الخيط 1 بالقيمة X .

و الشكل (1) يمثل خوارزمية الضغط بطريقة LZW .

(3-3-1-2) فك الضغط بطريقةLZW (LZW decoding)

من أجل فهم طبيعة عمل محلل شفرة LZW يجب علينا إن ندرك الخطوات الثلاث التي يقوم بها المشفر في كل مرة يكتب فيها شي ما على شريط الإخراج وهي كالاتي :-

1. إخراج مؤشر القاموس الذي يشير إلى الخيط .

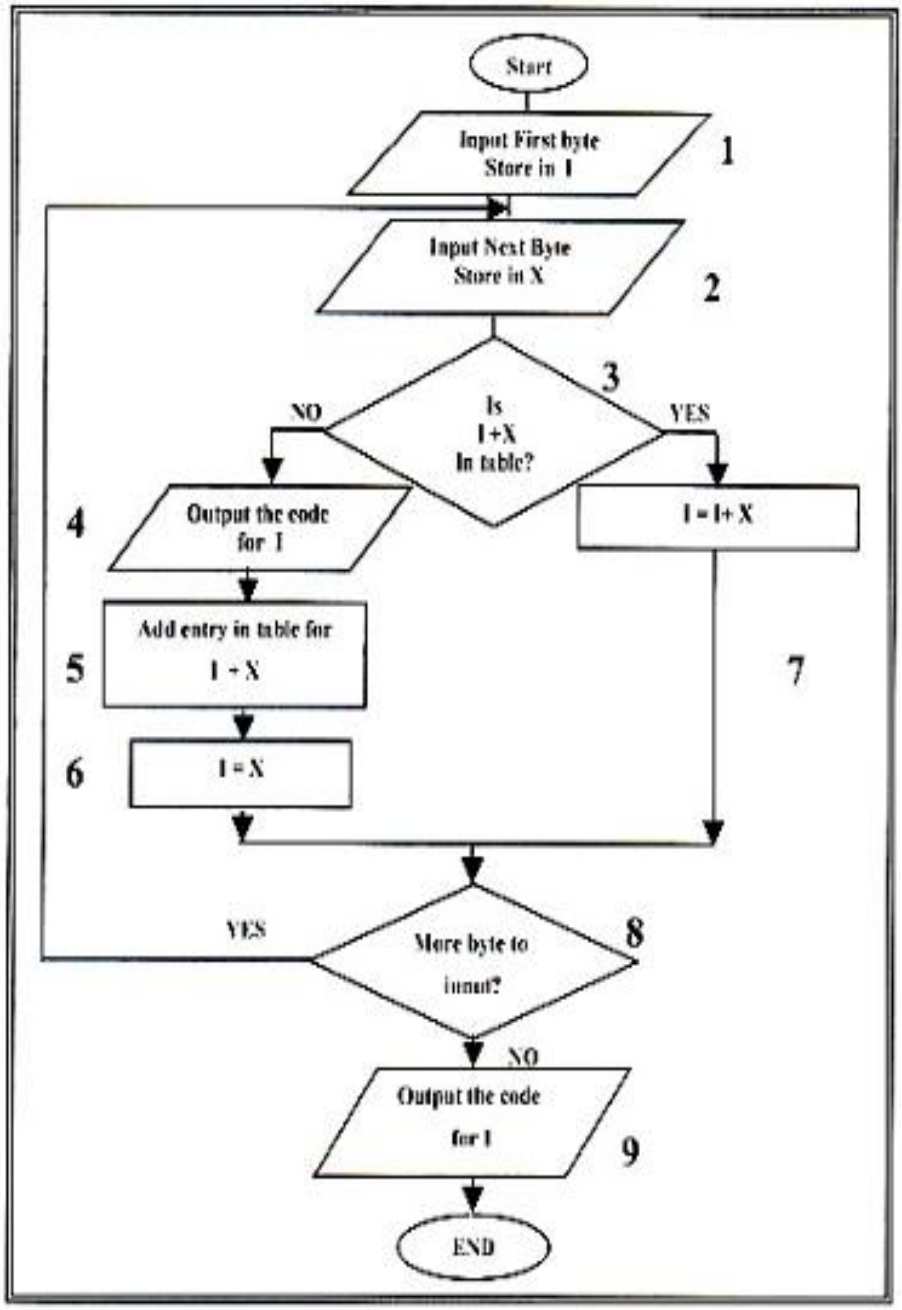
2.خزن IX في المدخل المعرف التالي في القاموس.

3.ابتداء الخيط | بالرمز X.

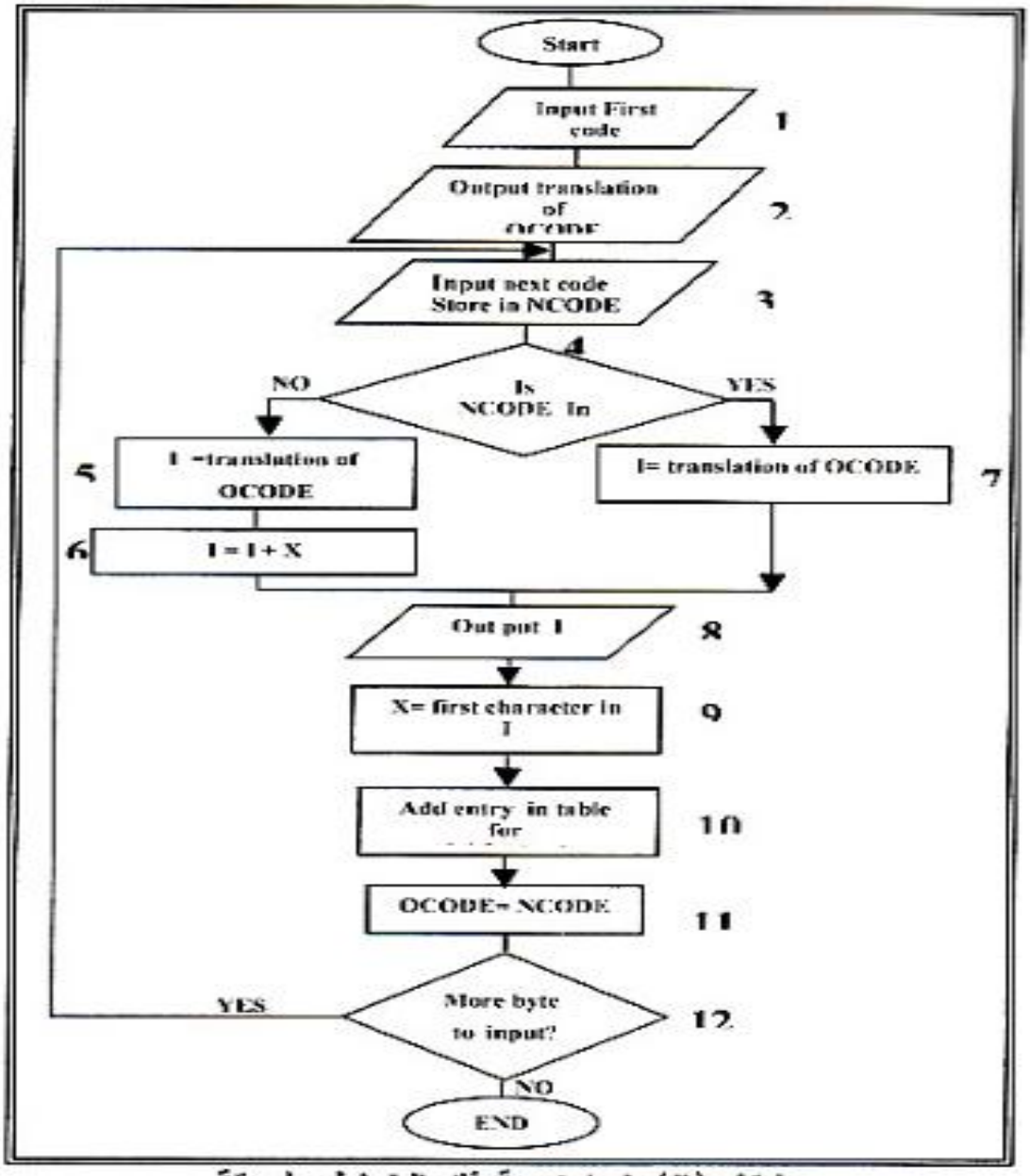
محلل الشفرة يبدأ بأول المدخل للقاموس التي ابتدأت بكل رموز الأبجدية (عادة 256 رمز) ، ثم بعد ذلك يقرأ شريط الإدخال الذي يحتوي مؤشرات للقاموس) ويستخدم كل مؤشر لاستعادة الرموز غير المضغوطة من قاموسه وكتابتها في شريطه الإخراجي ، كذلك يبني قاموسه الخاص

بنفس الطريقة في المشفر (هذه الحقيقة عادة يعبر عنها بالقول إن المشفر وحلال الشفرة متزاملان أو يعملان في lock step) في أول خطوة تحليل شفرة حلال، الشفرة يدخل أول مؤشر OCODE ويستخدمه لاستعادة عنصر القاموس 1 الذي يكتب في شريط إخراج حلال الشفرة الخيط XI يحتاج إن يخزن في القاموس ولكن الرمز X لازال غير معروف ، وهو الذي سيكون أول رمز في الخيط المستعاد والتالي في القاموس

في أي خطوة أخرى بعد أول خطوة ، حلال الشفرة يدخل المؤشر التالي NCODE ويستعيد الخيط الثاني لـ من القاموس ويكتبه في شريط الإخراج ويفصله ويرمز له X ويخزن الخيط IX في مدخل القاموس المعرف التالي (بعد الفحص للتأكد إن الخيط IX هو غير موجود في القاموس)، بعد ذلك المحلل يحرك J إلى] وهو جاهز للخطوة التالية. والشكل (٢) يمثل خوارزمية فك الضغط باستخدام طريقة [5-1]



شكل (١) خوارزمية الضغط بطريقة LZW



شكل (٢) خوارزمية فك الضغط بطريقة

وفيما يلي مثال تفصيلي يوضح عمل هذه الخوارزمية : [1]

مثال : لو كان لدينا النص التالي :

the/rain/in/Spain/falls/mainly/on/the/plain

ان خوارزمية الضغط تستخدم متغيرين (char) و (string)، الرمز (char) يحمل رمز واحد مثل بايت واحد بين القيمة (0, 255) ، بينما المتغير string خيط رمزي متغير الحجم، مثل مجموعة من رمز أو أكثر.

في الصندوق 1 في الشكل (1) أعلاه البرنامج يبدأ بأخذ أول رمز من ملف الإدخال ووضعه في المتغير string الجدول (٢) يمثل هذه الخطوة في السطر الأول وهذا يتبع بخوارزمية تكرارية لكل بايت مضاف من ملف الإدخال سيسيطر عليها في الصندوق (8)، في كل مرة يتم قراءة بايت من ملف الإدخال الصندوق (2) ، ويخزن في المتغير char . بعد ذلك يتم بحث جدول الرموز لتحديد فيما إذا كان دمج المتغيرين string + char حدد لهما شفرة أم لا الصندوق (3). إذا لم يكن هنالك تطابق فان ثلاث خطوات يتم تنفيذها كما موضح في الصندوق 6.5.4 في الصندوق 4 يتم كتابة string على ملف الإخراج، في الصندوق 5 شفرة جديدة تهيئ في الجدول للخيط الذي تم دمجه سابقا (char) + string في الصندوق 6 المتغير string يأخذ قيمة المتغير char، كمثال على هذه الخطوة يوضح في الصف 10−2 من الجدول (٢)، إما عندما يوجد التطابق في جدول الرموز الصندوق (3) دمج كل من string + char يخزن في المتغير string بدون أي فعل آخر الصندوق (7) لذلك فان وجود أي تطابق في جدول الرموز يؤدي إلى عدم حدوث أي فعل مثل هذه الحالة يمكن توضيحها في السطر 11 حيث إن التسلسل(string + char = in) يؤشر على انه لديه

شفرة سابقة في جدول الرموز في السطر 12 الرمز التالي من ملف الإدخال (/) يضاف الى التسلسل وجدول الرموز يبحث عن الخيط in ، حيث إن التسلسل السابق غير موجود في الجدول فان البرنامج يضيفه إلى الجدول، ويخرج شفرة الرموز السابقة (الشفرة (262)، هذا التسلسل من الأحداث يستمر حتى لا يبقى لدينا رمز آخر في ملف الإدخال.

إما في خوارزمية فك الشفرة الموضحة في الشكل (٢) أعلاه فيتم قراءة كل شفرة من الملف المضغوط ومقارنتها مع جدول الرموز لتحديد ترجمته الرمز المرتبط بكل شفرة في حال معالجة كل شفرة فان جدول الرموز يتم تحديثه لذلك فهو يحتوي على شفرات مطابقة لما تم تكوينه في برنامج الضغط.

هنالك تعقيد بسيط في خوارزمية فك الضغط وهو بالتأكيد ناتج من استلام خوارزمية فك الضغط الرمز غير موجود في جدول الرموز حيث يتم معالجة هذه الحالة في الصندوق 4,5,6

	STRING + CHAR	*CHAR*	*In table?*	*Output*	*Add to Table*	*New STRING*
Comments						
1	t	t			t	First character-no action
2	h	th	no	t	256=th	h
3	e	he	no	h	257=he	e
4	/	e/	no	e	258=e/	/
5	r	/r	no	/	259=/r	r
6	a	ra	no	r	260=ra	a
7	i	ai	no	a	261=ai	I
8	n	in	no	i	262=in	n
9	/	n/	no	n	263=n/	/

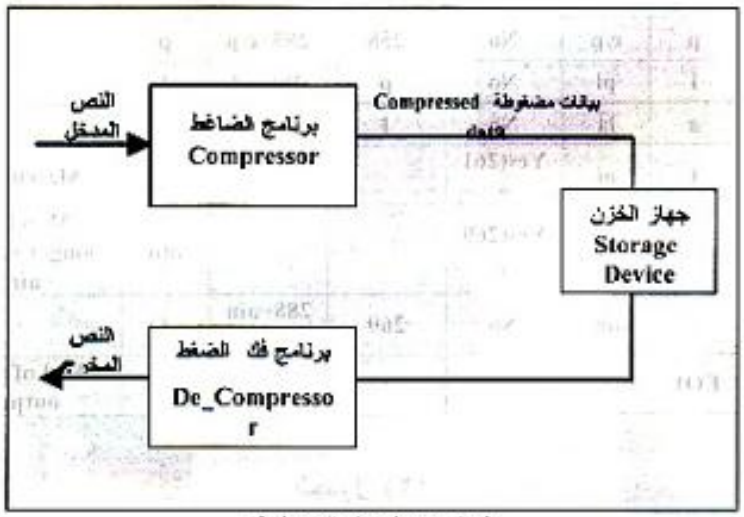
10	i	/i	no	/	264=/i	i	
11	n	in	Yes(262)			in	First match found
12	/	in/	No	262	265=in/	/	
13	S	/S	No	/	266=/S	s	
14	p	Sp	no	S	267=Sp	p	
15	a	pa	no	p	268=pa	a	
16	i	ai	Yes(261)			ai	Matches ai , ain not in table
17	n	ain	no	261	269=ain	n	ain add to table
18	/	n/	Yes(263)			n/	
19	f	n/f	no	263	270=n/f	f	
20	a	fa	No	f	271=fa	a	
21	l	al	no	a	272=al	l	
22	l	ll	No	l	273=ll	l	
23	s	ls	No	l	274=ls	s	
24	/	s/	No	s	275=s/	/	
25	m	/m	No	/	276=/m	m	
26	a	ma	No	m	277=ma	a	
27	i	ai	Yes(261)			ai	Matches ai
28	n	ain	Yes(269)			ain	Matches longer string , ain
29	l	ainl	No	269	278=ainl	l	
30	y	ly	no	l	279=ly	y	
31	/	y/	No	y	280=y/	/	
32	o	/o	No	/	281=/o	o	
33	n	on	No	o	282=on	n	
34	/	n/	Yes(263)			n/	
35	t	n/t	No	263	283=n/t	t	
36	h	th	Yes(256)			th	Matches th, *the* not in table

37	e	the	No	256	284=the	e	*the* added to table
38	/	e/	Yes			e/	
39	p	e/p	No	258	285=e/p	p	
40	l	pl	No	p	286=pl	l	
41	a	la	No	l	287=la	a	
42	i	ai	Yes(261)			ai	Matches ai
43	n	ain	Yes(269)			ain	Matches longer string ain
44	/	ain/	No	269	288=ain/	/	
45	EOF	/		/			End of file, output

الجدول (٢)

(٤) النظام المقترح

في البند السابق تم شرح تقنية LZW التي استخدمت في هذا البحث حيث تم تصميم نظام للضغط اعتمد بشكل أساسي على هذه التقنية بالاضافة الى مرحلة اخرى لاكمال عمل هذا النظام . ان الهيكل العام لهذا النظام يتكون من برنامج الضاغط Compressor وجهاز للخزن وبرنامج فك الضغط De_compressor وكما موضح في الشكل (٣)



شكل(٣) الهيكل العام لنظام الضغط

(٤-١) برنامج الضاغط: Compressor program

ان برنامج الضاغط يتكون من مرحلتين :

1. المرحلة الأولى : الضغط باستخدام تقنية LZW .

٢. المرحلة الثانية : دمج البايتات للتخلص من الثنائيات الغير

ضرورية الناتجة من المرحلة الأولى

وفيما يلي تفصيل لكل مرحلة.

(٤-١-١) المرحلة الأولى : مرحلة الضغط باستخدام تقنية LZW

في هذه المرحلة استخدمت خوارزمية LZW المذكورة في الفصل السابق مع استخدام قاموس بحجم (١٢) ثنائية وبذلك يتوفر لنا إمكانية تمثيل ٤٠٩٦ رمز ، حيث استخدمت ثلاثة أنواع من الملفات :

1. **الملف الأول** : يمثل ملف الإدخال الذي نرغب بتقليص حجمه وهو من Text file.

2. **الملف الثاني:** يمثل القاموس (جدول الرموز) وهو عبارة عن ملف من القيود كل قيد فيه يتكون من حقلين : الحقل الأول يمثل الرمز والحقل الثاني يمثل شفرة الرمز.

3. **الملف الثالث:** وهو ملف الاخراج وكل قيد فية عبارة عن قيمة من نوع Integer تمثل قيمة الإخراج أي الملف المضغوط) . يتم اولاً تكوين جدول الرموز القاموس الحاوي على الرموز الاساسية فقط (255-0) ومن ثم تبدأ عملية الضغط بقراءة الرمز الأول واخراج الشفرة الممثلة له ثم تكرار دمج الرموز وتحديث القاموس في كل مرة يتكون فيها رمز غير موجود سابقاً واخراج شفرات الرموز الموجودة لكتابتها على ملف الاخراج، كما تم تفصيله اعلاه والشكل (٤) يمثل نهج معالجة للضغط بخوارزمية LZW

```
procedure LZW_coding;
  var
    Ch,str,str1:st;
    I,, x:st;
    Le,i:integer;
  Begin
    Creat_code_table;
    Assign(f1,'out.dat');
    Rewrite(f1);
    Assign(t,'input.dat');
    Reset(t);
    While not eof(t) do
     Begin
       Readln(t,l);
       Ch:=copy(l,1,1);
       Str:=ch;
       Le:=length(l);
       For i:=2 to le do
        Begin
          Ch:=copy(l,i,1);
          Str1:=str+ch;
          Search(str1,str,flag,code);
          If flag then
             Str:=str1
          Else
            Begin
               R1.num1:=code;
               Write(f1,r1);
               Add(str1);
               Str:=ch;
            End;{else}
        End;{for}
      End;{while}
    Search(str1,str,flag,code);
    R1.num1:=code;
    Write(f1,r1);
    Close(f1);
    Close(t);
  End; {LZW_coding}
```

شكل (٤) نهج معالجة للضغط بخوارزمية LZW

(٤-١-٢) المرحلة الثانية : مرحلة الدمج

ان الملف الناتج من المرحلة الأولى يحتوي على قيم تمثل شفرة الرموز المضغوطة وكل قيمة تمثل بنوع بياني هو (integer) ، وكما هو معروف فان النوع integer يمثل بـ (٢ بايت) أي (١٦ ثنائية) ، ولائنا حددنا القاموس بـ (١٢) ثنائية اذن ستكون لكل قيمة ٤ ثنائيات غير مستغلة ولاجل الوصول الى امثل نسبة ضغط يتم التخلص من هذه الثنائيات كالتالي :

. كل قيمتين تمثل بـ (٤ بايت) اذن سيكون لدينا ٣٢ ثنائية ، ٢٤ منها تكون مستغلة والباقي غير مستغل ، وبذلك يمكن تحويل كل ٤ بايت الى ثلاثة بايت وكما موضح أدناه :

ان التمثيل الثنائي للاعداد التالية (2503, 2241) هو

2503	0	0	0	0	1	0	0	1	1	1	0	0	0	1	1	1
2241	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	1

يتم أولا استقطاع ١٢ ثنائية من العدد الأول ثم ١٢ ثنائية الأخرى من العدد الثاني فيتكون لدينا ٢٤ ثنائية كما يلي:

1	0	0	0	1	1	0	0	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1

ومن ثم يتم استقطاع كل ثمانية ثنائيات لتكوين ثلاثة اعداد وكما يلي:

199	1	1	0	0	0	1	1	1
25	0	0	0	1	1	0	0	1
140	1	0	0	0	1	1	0	0

إن ملف الإدخال هو ملف من نوع integer وهو الملف الناتج من المرحلة الأولى ، أما ملف الإخراج فسيكون من نوع Byte وهو الملف النهائي العملية الضغط . والشكل (٥) يوضح نهج معالجة لهذه الطريقة.

```
Procedure p;
   Var
    Stn: sw;
    X2: sts;
    I, J, n1: integer;
    X1: sttt;
   Begin
    Assign( f1, 'out.dat');
    Reset( f1 );
    Assign( f2,'end.dat');
    Rewrite( f2 );
    While not eof( f1 ) do
     Begin
      Stn:='';
      I:=1;
      While (i<= 2) and not eof( f1 ) do
       Begin
        Read( f1, r1 );
        N1:= r1.num;
        Dc_bi( n1, st1);
        X2:= copy ( st1, 5,12 );
        Stn:= stn + x2;
        I:= i+1
       End;
      If (I<3) and ( eof( f1 ) ) then
       Begin
        N1:= 0;
        Dc_bi( n1, st1);
        X2:= copy ( st1, 5,12 );
        Stn:= stn + x2;
        I:= i+1
       End;
      J:= 1;
      For i:= 1 to 3 do
       Begin
        X1:= copy( stn, j, 8 );
```

شكل (٥) نهج معالجة يمثل المرحلة الثانية للضغط

(٤-٢) برنامج فك الضغط De-compressor program:

(كما مرت عملية الضغط بمرحلتين فان مرحلة فك الضغط تمر بمرحلتين ايضا وهي:

1. مرحلة فك الضغط الناتج من المرحلة الثانية (مرحلة دمج البايتات)

2. مرحلة فك الضغط الناتج من خوارزمية LZW .

وفيما يلي شرح مفصل لكل مرحلة

(٤-٢-١) المرحلة الأولى :

في هذه المرحلة يستخدم الملف الناتج من برنامج الضغط كملف ادخال ، حيث يتم قراءة كل ثلاثة قيم معاً فيكون لدينا ٢٤ ثنائية كما في المثال التالي:

199	1	1	0	0	0	1	1	1
25	0	0	0	1	1	0	0	1
140	1	0	0	0	1	1	0	0

1	0	0	0	1	1	0	0	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1

ومن ثم يتم استقطاع كل 12 ثانية لوحدها وكالاتي :

1	0	0	1	1	1	0	0	0	1	1	1
1	0	0	0	1	1	0	0	0	0	0	1

ثم يتم إضافة أربعة أصفار لكل خيط فيتكون لدينا خيط رمزي طوله ١٦ثنائية كالتالي:

2503	0	0	0	0	1	0	0	1	1	1	0	0	0	1	1	1
2241	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	1

والشكل (٦) يمثل نهج المعالجة لهذه الخوارزمية.

(٤-٢ - ٢) المرحلة الثانية :

يعتبر ملف الاخراج الناتج من المرحلة الأولى كملف إدخال إلى المرحلة الثانية وفيها تتم عملية فك الضغط الناتج من خوارزمية LZW كالتالي :

يتم أولاً قراءة أول قيمة من ملف الادخال والبحث عن رمزها في جدول الرموز ومن ثم كتابة هذا الرمز على ملف الاخراج ثم يتم قراءة الرمز الثاني والبحث عنه في جدول الرموز فاذا كان غير موجود نستخرج الرمز التابع له ويدمج مع الرمز السابق فإذا كان موجود يستخرج الرمز التابع له من القاموس ثم نكتب الرمز على ملف الإخراج ونضيف مدخل في الجدول للرمز الجديد المتكون ونستمر بالعملية حتى نهاية الملف في هذه المرحلة لا نحتاج إلى الاحتفاظ بجدول الرموز من عملية الضغط وانما يتم تكوينه من خلال إضافة الرموز الجديدة إلى الجدول السابق و إكماله كما في عملية الضغط والشكل (٧) يمثل نهج معالجة لهذه المرحلة .

```
Procedure de_1;
   Var
     Sn: sw;
     I, j: integer;
     St1:string[8];
     X1: string[12];
     X2: string[16];
   Begin
     Assign( f2,'end.dat');
     Reset( f2 );
     Assign( f1, 'out.dat');
     Rewrite( f1 );
     While not eof( f2 ) do
       Begin
         Sn:='';
         For i:= 1 to 3 do
           Begin
             Read( f2, r2 );
             Dc_bi( r2.item, st1);
             Sn:= sn + st1;
           End;
         J:=1;
         For i:= 1 to 2 do
           Begin
             X1:= copy( sn, j, 12 );
             J:= j + 12;
             X2:= '0000' + x1;
             bi_int( x2, s1 );
             R1.num:= s1;
             Write( f1, r1 );
           End;
       End
     Close( f1 );
     Close( f2 );
   End;
```

شكل (٦) خوارزمية فك الضغط_ المرحلة الاولى

```
Procedure de;
  var
   Ocode,ncode:integer;
   Flag:boolean;
   Str,item,ch:st;
   Chr1:st;
   Chr: string[1];
  Begin
   Creat;
   Assign(f1,'out.dat');
   Reset(f1);
   Assign(f2,'out1.dat');
   Rewrite(f2);
   Read(f1,r1);
   Ocode:=r1.num;
   Search(ocode,item);
   R2.item1:=item;
   Write(f2,r2);
   While not eof(f1) do
    Begin
     Read(f1,r1);
     Ncode:=r1.num;
     Search1(ncode,flag);
     If  not flag then
      Begin
        Search(ocode,item);
        Str:= item;
        Str:=str+chr;
      End{if}
     Else
      Begin
        Search(ncode,item);
        Str:=item;
      End;{else}
     R2.item1:=str;
     Write(f2,r2);
     Chr:=copy(str,1,1);
     Search( ocode, item);
     Chr1:=item+chr;
     Add(chr1);
     Ocode:=ncode;
    End;{while};
   Close(f1);
```

شكل (٧) خوارزمية فك الضغط المرحلة الثانية

(٥) النتائج والتطبيق : تم تطبيق هذا النظام على عدد مختلف من الملفات فكانت النتائج كما في الجدول ادناه.

	حجم الملف الأصلي (Byte)	حجم الملف المضغوط بعد المرحلة الأولى & نسبة الضغط	حجم الملف المضغوط بعد المرحلة الثانية	نسبة الضغط الكلية
١	١٥١٩	٩٨٣ ٣٥،٢٨٦	٧٣٧	٥١،٤٨١
٢	٣٢١٦	١٨٢٥ ٤٣،٢٥٢	١٣٦٨	٥٧،٤٦٢
٣	٥١٠٠	٢٢٤٣ ٥٦،٠١٩	١٦٨٢	٦٧،٠١٩
٤	٩٧٣	٧٢١ ٢٥،٨٩٩	٥٤٠	٤٤،٥٠١
٥	٢٥١	١٦٥ ٣٤،٢٦٣	١٢٣	٥٠،٩٩٦
٦	٩٧	٧٥ ٢٢،٦٨٠	٥٦	٤٢،٢٦٨
٧	٥٤	٣٨ ٢٩،٦٢٩	٢٨	٤٨،١٤٨

جدول (٣-١) نتائج تطبيق خوارزمية الضغط

من الجدول (١٣) السابق نلاحظ ان النسبة المئوية للضغط التي تقاس

بالمعادلة:

النسبة المئوية للضغط = ١٠٠ (١- نسبة الضغط)

نسبة الضغط =حجم المخرجات

حجم المدخلات

تختلف باختلاف الملفات المستخدمة اعتمادا على نسبة التكرار في كل ملف.

المصادر

1. Guye, Blelloch " Introduction to Data Compression Carregre Mellon University, Blelloch, 2001, @cs.cmn.ddu.

2. Salomon, David " Data compression "Springer –Verlay– New yourk, 1997.

3.Apractical introduction to compression " http://compression .ca. data

4.Leen, Todd K. CSES69 Data and signal compression ", 2001, tlea@cse.og, edu.

5– السيف ، خليل ، عبد الكاظم، اسراء " خوارزمية LZW لضغط الصوت عند الزمن الحقيقي جامعة الموصل, ٢٠٠١ .

Develop of an algoritham to compressed text files

Nahla Abbas Flayh

Basrah university-Science collage

Computer science department

Abstract

Data compression is consider a very important field of the various applications of computers. Besides decreasing the storage capcity needed by the data in order to be saved on a disk , it also decreased the volume of the required package to transmit the data via the digital network, and, hence, decreases the amount of time that is necessary to transmit the data. Moreover it decreases the errors during the process of transmission.

Due to the importance of text files this paper dealt with the compressed text files using compression algorithm called LZW algoritham, after developing and linking it to another suggested method to compress different types of text files.

The proposed method includes two programes decompression. compression

The compression passes through two stages:

(1The use of well known LZW algorithm to get a file that is smaller than the orginal one and that contain codes to be compressed later.

(2The use of propose concat algoritham to get ride of the unnecessary bits of the propaget file of the first stage.

The decompression programme works as the apposite to compression programme since it decompressed the file resulted of concat algorithm and, then decompressed the compression from LZW algorithm. The suggest algorithm is a

lossles method, so the file that result from decompression is the same file that compressed in size and contain.

This method has been applied to a number of different text files with different size and data type.