INCLUDEの第三回
公開日:04/5/29

【第三回】:可逆圧縮に挑戦！

皆さん、こんにちは。いよいよ軌道に乗ってきた「INCLUDE」第三回です。今回は、実用性第一位の「可逆圧縮」を体験してみましょう！

さて、皆さん、第一回の「音声圧縮」を見た方なら、「可逆圧縮」という言葉はご存知でしょう。まず、知らない方に説明しておきましょう。なお、知ってる方も、復習のつもりで見てください。

ここに、一枚のデジカメ写真のデータがあります。(あるんです！)それで、このデータ、絵の形をしていますけど、中身はただの数字の羅列なのです。

たとえば

0101010101001011010100100101010100010101001010 10・・・

みたいに。

そして、それをCD-Rなり、フロッピーなり、MOなりに保存しますけど、ところで、考えてみてください。ただずらずらとデータを保存していくのは、なんだかもったいない気がしますね。ゴミ箱にごみを捨てるときだって、ただお菓子の箱をポンと捨てたりするのは実に無駄が多いです。では、皆さんは効率よくごみを捨てるとき、どうしますか？そう。

「GUSHAー！！！！！！！！！！！！！」

そうです。「つぶす」ですよね。グシャリとつぶせば、スペースが無駄にならずにすみます。

では、データにも「GUSHA」はできるのでしょうか？まぁ、答えは「Yes」そして「No」です。(この言い回し、どこかで聞いたような・・・)まず、Yesといったのは、いらないCDなりフロッピーなりを見つけてきて、足で踏み潰して下さい。(自己責任でお願いします。)ほら、小さくなったでしょう。では、Noのほう。そのつぶれたCDをドライブに入れて、データを読み取ってください。えっ？入らない？そうでしょうね。だって、形が変っちゃったんですから。入るわけがありません。

そうです。データをつぶすのは、その「つぶし方」が、問題になってくるんです。「データ」である利点、そう、それは、数字の羅列であること。

つまり、データの圧縮とは、データを、つまりは数字の羅列を、どうやって無駄なく記録するかと言うことなんです。そして、普通は、その数字の順番をうまく組替えたり、「1Byte＝256種類のデータ」という「上限」を利用して、出現頻度を逆手にとってサイズを減らしたり、するわけですね。そして、つぶしたデータが、「完全に」元に戻る圧縮、つまり、スポンジのような弾力性のある圧縮方法が「可逆圧縮」なのです。「可逆」とは、逆が在り得る、つまり、元に戻ると言う意味です。

圧縮の利点は、ただサイズが小さくなるだけではありません。CDーRのような一度しか書き込めないメディアに、毎日のデータのバックアップをしているあなた、そうです。無駄に記録すると、お金がかかってしまいますね。それを小さいデータで記録できたら・・・特に、「ビットマップファイル」と呼ばれるファイルの中には、100分の一位にまで小さくなることがあるので、お金がかからずにすみます。また、デジカメのメモリーカードも、結構寿命が早いです。これも、書き込む量が減れば、寿命を延ばすことにもつながるわけです。

とりあえず、何よりも感じておいてほしいことは、所詮、データは数字の羅列に過ぎないと言うこと、そして、それらは計算で出すことができるものである、と言うことです。まず、何も考えずに、このことを頭に刻んでおいてください。

さて、説明はこれぐらいにして。

いよいよ実習に入ります。例によって、圧縮は、著作権・特許問題の絡む世界なので、僕のオリジナルの方法を説明しましょう。

【パターン圧縮】

とりあえず。圧縮において問題になってくることは、「圧縮できないときもある」と言うことです。

圧縮できないときは、そのデータを見捨てる・・・なんてことがあっては、データが元に戻りませんので、大変です。「田嶋、君の命は無駄にはしない。すまん！」では困るのです。(田嶋君はどうでもいいです。友情出演ですから。)こういうときは、素直に圧縮できないから、できないものをそのまま記録する、つまり、圧縮しないと言うことにすれば、問題なく元に戻ります。しかし、一方で、圧縮できないときが多いと、圧縮できる機会が少なく、圧縮してもサイズが小さくならない、ということにもなります。

よい圧縮方法とは、「圧縮できるとき」と「圧縮できないとき」を区別することが、圧縮自体の負荷にならないような圧縮方法です。まぁ、見ていればわかりますので、まず「パターン圧縮」というものを紹介しましょう。

まず、データがあるとします。そのデータが、「001001001001001001001001001001」だったとします。さて、これをそのまま記録したら、30バイトの容量になります。ですが、

「おやっ？「001」というのが10個あるように見えるぞ？」

それに気づいたあなた、えらい！そうです。このデータは、「001」が10個、連続して続いたものなのです。さて、これに「GUSHA」してみましょうか。

まず、「001」を「1番」と決めます。それ以外は「0番」です。1番か0番か、と言うのは、2通りの選択です。ですので、「1ビット」ですよね。では、これを利用して、以下のようにデータを組替えます。

「1番・1番・1番・1番・1番・1番・1番・1番・1番・1番」

こうすると、1番のときが10個、つまり、10回、1番であることを記録すればいいのですから、1＊10＝10ビット、ですんでしまいます。つまり、圧縮が成功した、ということです。やったね！

しかし、つかの間の喜びはさておき、問題はここからです。先ほど、「001」は1番と決めたのは誰ですか？正直に白状してください。そう、あなたですね！(えっ？僕ですって？はい、そうですが・・・刑事さん、許してください。僕はただそう決めただけなんですから。これって、罪にはならないですよね？刑:ばか者！お前がそんなことを勝手に決めたりするから、ほかの人が・・・以下略)。かってに「001は1番」と決めた以上、そうでないときはどうするんだ、ということになってきます。たとえば、次のようなファイルだったらどうしましょう。

0101001010101110101001010101010011010101010110101 01

もう、お手上げですね。001なんて1つしか見つかりません。「他の皆さん、さようなら」になってしまいます。それでは困り者ですね。

だから、圧縮できないときがあるんです。そうしたときに、どうするか？

「そう、それが問題だ。(笑)」

こういうときは、素直に圧縮できませんと言いましょう。つまり、先ほど「0番」と決めた「その他」

に分類すればいいのです。そう、0番かどうかは1ビットで指定できますから、圧縮できないデータを1バイトずつ区切ってしまいましょう。そして、圧縮できないことを示す「0番」をくっつけてしまいます。

0番・01010010・0番・10101110・・・以下略。

こうすることによって、「圧縮できないとき」も、データを記録できるようになります。これで、「可逆圧縮」は完全に完成しました。おしまい。

・・・。喜んでいた田嶋君。しかし、ここで新たな問題が・・・。

圧縮できないから、「0番」としましたが、そうすると、圧縮できるはずの、唯一の「1番」が、0番の中に入って、圧縮できなくなっています。つまり、圧縮できるはずなのに、圧縮できないことになってしまうのです。こうなると、まったく圧縮できない部分しかなくなり、圧縮してもサイズが減らない・・・さっき述べたとおりになってしまいました。これでは「可逆」圧縮であっても「価値があり」ません。サイズが減るから意味のある圧縮方法なのです。

実を言うと、この方法、こういった「圧縮できないとき」にめぐり合うことが多く、ぜんぜんサイズが減らなかったのでお蔵入りになってしまった方法なのです。まあ、全貌は明らかにしますけど・・・。もっと効率よくなる改造をできる方は、私の方までご連絡ください。(メールアドレスは、各サイト下の、「サイトマップ・ポリシー・管理人」の「管理人」ページにあります。)

・・・と言うことで、全貌公開!

先ほどの「1番」の部分を、16番まで増やします。そうすると、16通りの場合ができるので、4ビットの指定になります。そして、「001」の部分を「xxxxxxxx」つまり、8つの01まで増やします。そうすると、16種類の中に入れば、8ビットが4ビットになり、半分を「GUSHA」することができます。また、その他のときは、4ビットの無駄が出るものの、しっかりとデータはもとに戻ります。

これを思いついた当時は、勇んで作ってみたんですが・・・あんまり16種類の中に入る機会がないんですよね・・・。だから、さっきみたいにサイズが減るどころか、その他の指定の分が多く、無駄にサイズが増えてしまって・・・だめだめです。

まあ、そのだめ部分を修正し、拡張したのが、【論理圧縮】です。いつか書きますので、楽しみにしていてください。

と言うことで、おしまい!

ダウンロード

さて、記事に使ったプロジェクトを公開します。

PDFドキュメント・・・パターン圧縮の説明書

VisualC++6.0プロジェクト・・・圧縮シミュレーションのプロジェクト

メール等の受付

当サイトの管理人は、MORIOです。

質問やご要望、ご感想、苦情などは、メールで受け付けております。以下のアドレス宛に送ってくださいませ。

master@morik.net

form 2006/1/9