**压缩程序测试**

1 测试目标

测试压缩程序的压缩、解压功能在处理不同类型的文件时是否可以正常工作，以及压缩的效率。

2 测试方法

采用黑盒测试法，对不同类型、大小的文件进行压缩操作，并观察压缩是否成功、压缩后文件的大小。

3 测试用例

1、压缩功能测试

测试目的：压缩功能的有效性。测试前提是文件大小大于1kb，此测试共设计了 2 个用例，如下表1，表2所示。

表1测试用例 1

|  |
| --- |
| 编号： compress01 |
| 项目： 压缩功能用例测试 |
| 内容： 压缩10kb大小的markdown文件 |
| 前提条件：文件大小大于1kb |
| 测试步骤： |
| 1.运行程序选择压缩对并输入文件路径地址 |
| 2.对文件进行压缩 |
| 预期结果：压缩成功。 |
| 结果和结论：压缩成功，压缩后文件大小为8kb。 |

表2测试用例 2

|  |
| --- |
| 编号： compress02 |
| 项目： 压缩功能用例测试 |
| 内容：压缩大小为9kb的文本文件 |
| 前提条件：文件大小大于1kb。 |
| 测试步骤： |
| 1. 运行程序选择压缩并输入文件路径 |
| 2. 对文件进行压缩 |
| 预期结果：压缩成功。 |
| 结果和结论：压缩成功，压缩后大小变为5kb。 |

表3测试用例 3

|  |
| --- |
| 编号： compress03 |
| 项目： 压缩功能用例测试 |
| 内容： 压缩4mb大小的PNG文件 |
| 前提条件：文件大小大于1kb |
| 测试步骤： |
| 1.运行程序选择压缩对并输入文件路径地址 |
| 2.对文件进行压缩 |
| 预期结果：压缩成功。 |
| 结果和结论：压缩成功，但压缩后文件大小不变，效率为0 |

2.解压功能测试

测试目的：测试程序是否能正确解压被压缩过的文件。测试前提是被解压的文件是被本程序压缩的，此测试共设计了2 个测试用例，如下见表3，表4。

表4测试用例 4

|  |
| --- |
| 编号： decompress01 |
| 项目： 解压功能用例测试 |
| 内容： 对压缩过的md文件进行解压 |
| 条件：被解压的文件是被本程序压缩过的，即后缀为huf |
| 步骤： |
| 1. 运行程序选择解压后输入文件路径 |
| 2. 对文件进行解压 |
| 预期结果：解压成功 |
| 测试结果和结论：一致 |

表5测试用例5

|  |
| --- |
| 编号： decompress02 |
| 项目： 解压功能用例测试 |
| 内容：解压被压缩过的文本文件 |
| 前提条件：文件是被本程序压缩过的，即后缀为huf |
| 测试步骤： |
| 1. 运行程序选择解压并输入文件路径 |
| 2. 解压文件 |
| 预期结果：文件解压成功。 |
| 测试结果和结论：一致。 |

3.压缩性能的分析

Huffman编码是程序的核心，压缩的性能主要依赖与Huffman编码，在一些符号出现频率明显高于其他符号的文件中效果特别好。例如，在英语文本文件中，某些字母如 'e', 't', 'a', 'o' 的出现频率比 'z', 'q', 'x', 'j' 高得多。Huffman编码利用这种不均匀的频率分布来实现较好的压缩效果。此外某些像素值比其他值更常见的灰度图像中，Huffman编码也能表现得非常好。

符号分布均匀的文件对于每个符号出现频率大致相等的数据，Huffman编码效果较差。而对于已经压缩过的数据，（如 JPEG 图像、MP3 音频文件或 ZIP 文件），通常没有多少冗余可以让Huffman编码利用。对这些文件的压缩效果很差。最后，在压缩非常小的文件时，Huffman树本身的开销（需要与压缩数据一起存储，以便解压缩）可能超过任何压缩增益。