Relatório do TP 1

Nome: Vítor Fitzherbert Souza
Turma: TN

Introdução

Este programa tem o objetivo de implementar o algoritmo LZ78 de compressão de arquivos, bem como decodificar os arquivos compactados para possibilitar a recuperação das informações originais.

Esse algoritmo, em alto nível, vai construindo um dicionário à medida que processa o texto do início para o fim, substituindo trechos que já ocorreram anteriormente no texto por seus respectivos índices nesse dicionário, a fim de economizar espaço. Para a descompressão, basta recriar a construção desse dicionário com base no arquivo compactado, substituindo os índices pelos seus trechos correspondentes.

Implementação

O programa se divide em 3 funções principais: `comprimir`, `descomprimir` e `main`, responsáveis respectivamente pela compressão de arquivos, descompressão, e leitura dos parâmetros de entrada. A função `main` é a mais simples, usando a função de biblioteca `getopt` para obter a partir da linha de comando os nomes dos arquivos de entrada e saída, realizando a abertura esses arquivos, e por fim chamando a função `comprimir` ou `descomprimir` dependendo da escolha do usuário. A função `main` também é responsável por determinar o nome padrão do arquivo de saída caso ele não seja informado, conforme as regras dadas na especificação do trabalho.

Compressão

A função `comprimir` usa uma árvore de prefixos para armazenar o dicionário usado pelo algoritmo LZ78. Cada nó da árvore corresponde a uma string já encontrada no texto, armazenando o índice pelo qual essa string poderá ser substituída nas ocorrências futuras, e um `std::map` contendo os filhos do nó, indexados por caractere. A string correspondente a um nó filho é herdada de seu nó pai, porém acrescida de um caractere ao seu final, sendo esse o caractere usado como índice no `std::map` do nó pai. No início da execução, a árvore contém apenas um nó, correspondendo à string vazia.

Após a inicialização da árvore, é repetido até o final do arquivo o processo de ler sequencialmente caracteres do texto até chegar a uma string que não está presente no dicionário. Isso é feito com um percurso na árvore, que começa no nó raiz, e para cada caractere lido, caminha para o nó filho indexado por esse caractere no `std::map`, até que se chegue a um nó que não possui um filho indexado pelo caractere lido. Quando isso acontece, o nó faltante é adicionado à árvore, efetivamente adicionando uma nova string ao dicionário, e o processo se recomeça no nó raiz. Ao fim de cada iteração desse processo, sabemos que ao remover o último caractere da string lida obtemos uma string que já fazia parte do dicionário, e portanto pode ser representada por um

índice. Assim, a string lida inteira pode ser reduzida a apenas esse índice e seu último caractere, realizando-se assim a compressão do texto.

Descompressão

A árvore de prefixos usada na função `comprimir` é útil para permitir, a partir de uma string, o fácil acesso ao seu índice no dicionário, mas para o processo de descompressão precisamos fazer o caminho inverso, logo uma representação diferente do dicionário é usada: um vetor onde cada elemento guarda um caractere e o índice de seu nó pai. Assim, partindo de um índice qualquer, podemos caminhar para o pai do nó que tem esse índice, e repetir esse movimento até que o nó raiz seja atingido, acumulando os caracteres lidos no caminho para obter a string correspondente a aquele índice. Na função `descomprimir`, esse processo é usado para substituir os índices no arquivo comprimido por suas respectivas strings, à medida que o dicionário criado na fase de compressão vai sendo reconstruído por meio dos caracteres não indexados.

Benchmarks

Arquivo	Descrição	Tamanho	Tamanho (comprimido)	Taxa de compressão
bee_movie.txt	Roteiro do filme “Bee Movie”	89952	56892	36,8%
constituicao1988.txt	Constituição de 1988	651790	290402	55,4%
dom_casmurro.txt	Livro “Dom Casmurro”	409610	230570	43,7%
lz78.cpp	Código deste trabalho	5599	3984	28,8%
novo_testamento.txt	Novo Testamento da Bíblia	942896	512806	45,6%
os_lusiadas.txt	Livro “Os Lusíadas”	344538	194970	43,4%
package_lock.json	JSON gerado pelo node.js	1220117	520306	57,4%
romeo_and_juliet.txt	Peça “Romeu e Julieta” (em inglês)	169255	100311	40,7%
sherlock_holmes.txt	Livro “As Aventuras de Sherlock Holmes” (em inglês)	607425	339714	44,1%
tp1.txt	Especificação deste trabalho	6550	5730	12,5%