Projecto 1

Teoria da Informação

Instruções para submissão

- A submissão deve consistir de um ficheiro zip,

  `<número-de-aluno>-<nome-completo>.zip`

  que contém os ficheiros `compress.py` e `decompress.py`, e possivelmente um ficheiro `readme.txt` (facultativo). E mais nenhum outro ficheiro.

- A especificação do funcionamento dos ficheiros python está na descrição do projecto.

- Serão feitos uma série de testes automáticos aos ficheiros. Qualquer execução dos scripts python só pode ler e escrever os ficheiros passados como argumento. Não poderá fazer nenhum outro acesso a nenhum outro recurso do sistema (e.g. acesso à rede).

- Se qualquer outro ficheiro for lido ou escrito, o aluno poderá reprovar. Se o programa aceder à internet, o aluno poderá reprovar. Se o programa contiver código copiado, o aluno poderá ser disciplinado, ou expulso da universidade. Se o programa contiver código malicioso, o aluno poderá ser expulso da universidade, ou mesmo processado em tribunal.

- Ou seja, vou executar os scripts no meu computador privado (com algum sandboxing básico, mas sei lá, estou longe de ser um perito em segurança informática) e ficarei muito, muito, muito chateado se houver qualquer tentativa desse género. E respeito o vosso know-how nesta matéria para saber que se um aluno do vosso curso me quisesse muito lixar a vida, provavelmente conseguia `:)`

- De resto, os algoritmos devem ser razoávelmente eficientes. Como limite superior, a compressão de um ficheiro de 500Kb deve demorar menos que 10 minutos[1] e utilizar menos que 300Mb de RAM. O programa será abortado caso contrário. É de notar que este limite seria já completamente irrazoável para uma ferramenta prática. Mas a versão que eu ensinei do algoritmo Lempel–Ziv é muito simples e pouco optimizada, e estamos a utilizar Python.

  Se o algoritmo demorar mais tempo que o razoável, será considerado mal implementado.

  O importante aqui é assegurar que quando estão a ler bits do input, durante a compressão, para um novo bloco, não tenham de percorrer uma

[1] No CPython. Menos de 2 minutos no PyPy.

lista do princípio ao fim. Usem um diccionário ou alguma outra estrutura de dados que vos permita saber rapidamente se a sequência que estão a acumular já apareceu antes. (Essencialmente, queremos um algoritmo com complexidade n ou $n \log n$ e não n^2.)

- Peço alguma compreensão se eu demorar a responder perguntas sobre os projectos, porque vou ter 30 projectos para avaliar.