Projeto 1 - FP

Nome: Gabriel Jorge Menezes
RA: 11201921315
Link do video

O objetivo do projeto é implementar um pequeno motor de busca com buscas exatas, ou seja achar documentos que contêm os tokens “X Y Z..” nesta exata sequência.
Nesta primeira parte o indexador foi implementado, basicamente recebemos um diretório com documentos no formato de texto (txt), cada um desses documentos receberá um ID interno e será indexado, onde resultado é acumalado em três grandes estruturas: um mapa do ID do documento para seu path e um mapa de token para sua Postings List. Onde Postings List contém duas listas, primeira lista refere a todos os IDs de documentos que contêm tal token e uma lista de listas em que cada valor da lista interna significa em qual posição este token foi visto no documento, por exemplo para um token t: [0, 1, 2], [[10, 20], [1, 2], [3, 4, 5]]. Significa que o documento 0 tem o token t nas posições 10, 20, o documento 1 tem o token t nas posições 1, 2 e assim por diante. Todo este processo é feito criando um estado inicial do indexador, toda e qualquer modificação que altere este estado recebe o estado atual e retorna um novo estado com as modificações necessárias. Ao finalizar a indexação o último estado do indexador é serializado em bytes para o disco para que possa ser reutilizado para busca (que será feito na parte 2 do projeto) e algumas estatísticas serão imprimidas no console.
Para rodar “stack exec indexer-exe <path do diretório que contém os documentos> <nome do arquivo que conterá o último estado do indexador serializado>”, por exemplo: “stack exec indexer-exe ./data ./db” (o projeto já vem com uma pasta “data” que contém alguns livros em formato .txt, retirados do Projeto Gutenberg, porém qualquer arquivo de texto funciona normalmente).
Algumas dificuldades foram encontradas durante o desenvolvimento, primeira delas foi a forma com que “stack” gerencia suas dependências, originalmente gostaria de ter usado segmentação de unicode usando o pacote text-icu, porém existe um bug neste pacote onde inputs muito grandes corrompem o segmentador causando um segmentação incorreta. Este bug parece ter sido corrigido na versão 0.8.0, mas a versão mais atual 0.8.0.5 ainda apresenta este bug, sendo assim tentei fixar a versão 0.8.0 porém sem sucesso, logo usei a função “words” que não é a ideal. Além disso, tive muitos problemas com HLS, onde qualquer adição de arquivo, dependência faz com que a mesma pare de funcionar e seja necessário reiniciar meu editor, junto a isso encontrei um bug, em que na minha função principal “indexFolder” a mesma verifica se o diretório informado existe, o ideal seria retornar “IO (Maybe IndexState)”, porém ao alterar a função para esta assinatura HLS para de funcionar completamente e assim não tenho mais feedback sobre o código ou autocompleção, tendo isto em vista prefiro ter uma LSP que funciona, para resolver o problema a função deve retornar “IO IndexState” e caso o diretório não exista terminamos o programa com falha.
Agora para alguns highlights a função em si que processa os tokens (processTokens) é bem concisa e simples de se entender, quando comparado com outras versões que já implementei em linguagens imperativas é realmente bem mais simples. Junto a isso as funções que geram as estatísticas finais se tornam bem simples de se implementar ao encadear funções que operam sobre listas.