Universidade Federal do Espirito Santo

Departamento de Engenharia

Engenharia de Computação

Relatório

Aluno(a): Arthur Coelho Estevão e Milena da Silva Mantovanelli

Professora: Veruska Carretta Zamborlini

Vitória

2022

1 Introdução

O Problema do Empacotamento, em que buscamos particionar um conjunto de itens com pesos de forma que a soma dos pesos dos itens em cada parte seja menor ou igual a um número inteiro dado, minimizando o tamanho do número de partes, é um problema extensamente estudado na computação. Trata-se de um problema NP-difícil, ou seja, é improvável que exista algoritmo polinomial para o mesmo e por isso, são usados algoritmos heurísticos e de aproximação para gerar uma solução próxima da ótima. Este trabalho, visa projetar e implementar heurísticas que executam de maneira rápida e produzam soluções próximas do ótimo. Sendo assim, foram implementadas quatro heurísticas: worst-fit, best-fit, worst-fit decrescente e best-fit decrescente.

2 Teoria

2.1 Algoritmos Heurísticos

Os algoritmos heurísticos são algoritmos que não garantem encontrar a solução ótima de um problema, mas são capazes de retornar uma solução de qualidade em um tempo adequado para as necessidades da aplicação. O objetivo de uma heurística é tentar encontrar uma solução “boa” de maneira simples e rápida.

2.1.1 Worst-fit

A heurística worst-fit, consiste em escolher sempre a maior área livre possível, de forma que a sobra seja grande o suficiente para ser usada em outras alocações.

Best-fit A heurística best-fit, consiste em escolher sempre a menor área possível que possa receber a alocação, minimizam o desperdício de memória

2.1.2 Worst-fit Decrescente

A heurística worst-fit decrescente, funciona da mesma maneira, requer que os itens do vetor de entrada sejam ordenados antes que eles sejam alocados.

2.1.3 Best-fit Decrescente

A heurística best-fit decrescente, funciona da mesma maneira, porém ela requer que os itens do vetor de entrada sejam ordenados antes que eles sejam alocados.

2.2 Algoritmo de ordenação utilizado

Para a ordenação do vetor de entrada das heurísticas, utilizamos da função qsort da biblioteca stdlib.h, pois devido analises anteriores podemos constatar que se apresenta como um dos melhores métodos de ordenação entre os existentes.

2.2.1 Função Qsort

A função qsort() nada mais é do que a implementação do QuickSort contida na biblioteca padrão da linguagem C. Ela permite ordenar qualquer tipo de vetor (int, float, string, struct, etc). Para tanto, basta definir a maneira como os elementos do vetor devem ser comparados.

2.3 Lista Duplamente Encadeada

É uma estrutura de dados que consiste em um conjunto de nós sequencialmente ligados. Cada nó contem a referência para o nó anterior e para o nó posterior na sequência de nós. As re-

ferências anteriores e posteriores dos nós inicial e final, respectivamente, apontam para algum tipo de terminador, tipicamente um nó sentinela ou nulo, para facilitar o percorrimento da lista.

3 Implementação

3.1 Código

Para melhorar o desempenho do worst-fit, a inserção de novos discos na lista de discos sempre é feita no início figura 1.

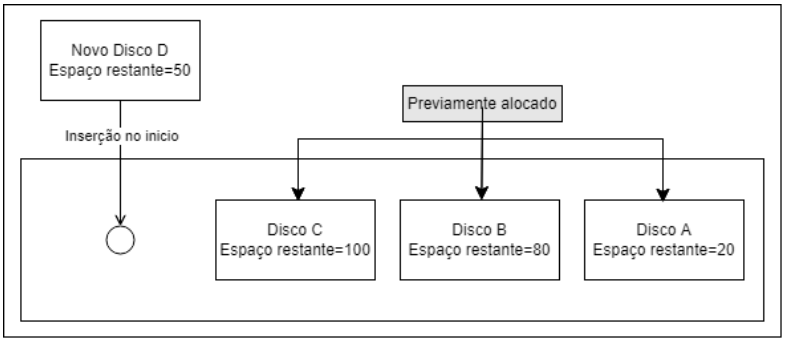

Figura 1: Diagrama de inserção de discos

Pois, como todo novo disco é o que possui maior espaço livre é nele que o arquivo é alocado. Depois, a lista de discos é ordenada figura 2 de forma decrescente para que o disco com maior espaço livre seja o primeiro da lista. Portanto, o arquivo sempre é inserido no primeiro disco da lista, não a vendo a necessidade de percorrer a lista de disco para inserir um arquivo.

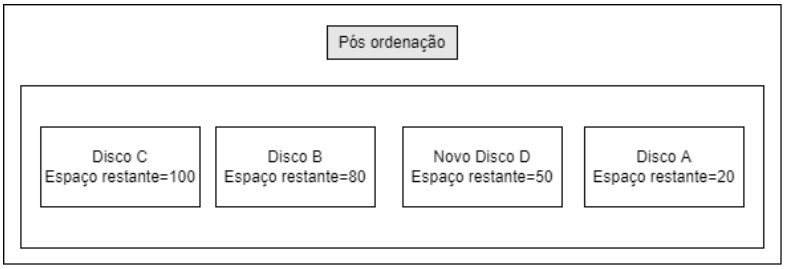

Figura 2: Discos pós-ordenação por qsort

Outra melhoria feita no código, foi a retirada de discos que não possuem espaço disponível figura 3, não sendo necessário percorrer discos vazios principalmente, isso é mais interessante para a heurística best-fit, que insere o arquivo no disco que possui um espaço livre maior e mais próximo do tamanho do arquivo, necessitando percorrer a lista toda vez que for alocar um arquivo em um disco.

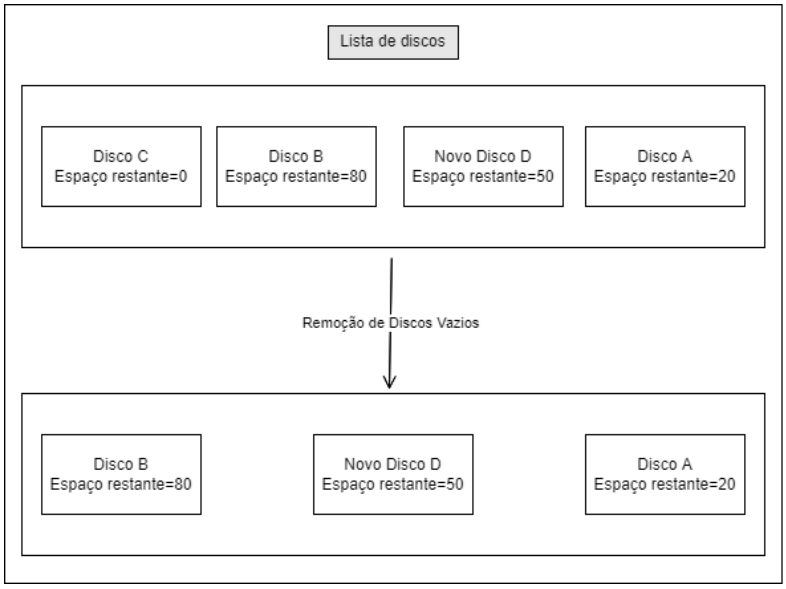

Figura 3: Discos pós-remoção de discos vazios

3.2 Makefile

Foi feito um Makefile pra facilitar a compilação e execução do programa. Como ele é possível:

- Compilar: make
- Executar: make run arg=diretorio-do-arquivo
- Compilar e executar: make compile-run arg=diretorio-do-arquivo
- Compilar, executar e gerar um arquivo .txt com os todos os resultados: make compile-run-all
- Compilar e executar com Valgrind: make valgrind arg=diretorio-do-arquivo
- Remove os arquivos não necessários pra gerar um arquivo executável: make clear

Github: `www.encurtador.com.br/sxLS2`

4 Conclusão

Concluímos que a heurística worst-fit é executada com o menor tempo gasto em comparação com o best-fit. Mas este último, têm um melhor aproveitamento dos discos, estando os itens do vetor ordenados ou não.

5 Referências

https://www.ic.unicamp.br/ rafael/alunos/exemplos/PIBIC/TiagoSouza.pdf https://www.dcce.ibilce.unesp.br/ saraujo/disciplinas/Metaheuristicas.pdf