Fase 1 do trabalho de:

**Estruturas de Dados Avançada**

**Docente:** Luís Ferreira



João Antunes N.º 23478

Índice

[Índice 2](#_Toc1897701694)

[Índice de Imagens 3](#_Toc614977698)

[Índice de Tabelas 5](#_Toc848294772)

[Tabela 1. Process Plan para um problema de escalonamento com dimensoa 8x7 5](#_Toc109260760)

[Introdução 7](#_Toc1826409398)

[Propósitos e Objetivos 8](#_Toc759292290)

[Fase 1 10](#_Toc1590749387)

[Estrutura de Dados 13](#_Toc749341220)

[Testes Realizados 15](#_Toc980432298)

[Conclusão 17](#_Toc842767634)

# Índice de Imagens

Figura 1. Estrutura de Dados usada para as Máquinas-----------------------------------------12

Figura 2. Estrutura de Dados usada para as Operações----------------------------------------13

Figura 3. Estrutura de Dados usada para os Jobs------------------------------------------------13

Figura 4. Erro com a Criação de Operações-------------------------------------------------------14

Figura 5. Erro Resolvido com a Criação de Operações-----------------------------------------15

Figura 6. Erro nos Cálculos----------------------------------------------------------------------------15

Figura 7. Erro dos Cálculos Corrigido---------------------------------------------------------------15

# Índice de Tabelas

# Tabela 1. Process Plan para um problema de escalonamento com dimensoa 8x7 ------10

# Introdução

Nesta Cadeira de Estruturas de Dados Avançadas foi nos pedidos a tarefa *Flexible Job Shop Problem*, usando Estruturas de Dados e Técnicas aprendidas nas aulas. Este projeto tem como Objetivo mostrar os nosso Conhecimentos na Linguagem C que foram adquiridos ao longo do Ano. O Objetivo deste Relatório é explicar as Estruturas de Dados usadas bem como certas Técnicas Usadas para solucionar vários problemas.

# Propósitos e Objetivos

Com este trabalho prático pretende-se sedimentar os conhecimentos relativos a definição e manipulação de estruturas de dados dinâmicas na linguagem de programação C. A essência deste trabalho reside no desenvolvimento de uma solução digital para o problema de escalonamento denominado Flexible Job Shop Problem (FJSSP). A solução a implementar deverá permitir gerar uma proposta de escalonamento para a produção de um produto envolvendo várias operações e a utilização de várias máquinas, minimizando o tempo as unidades de tempo necessário na sua produção (makespan). Um FJSSP pode ser formulado da seguinte forma:

1. Existe um conjunto finito de n jobs que têm de ser processados por um conjunto finito de m máquinas;

2. O conjunto de m máquinas é identificado por: M = {M1 , M2 , ..., Mn};

3. Um job é constituído por uma sequência de ni operações como: (Oi, 1 , Oi, 2 , ...,Oi, ni).

4. Cada operação deve ser executada para completar o job. A execução de cada operação j de um job i (Oi, j) requer uma máquina de um conjunto de máquinas Mi, j . O tempo de uma operação Oi, jrealizada na máquina em Mi, j é pi, j, k . As seguintes suposições são consideradas num problema FJSSP:

a. Todas as máquinas estão disponíveis no instante t = 0. 1

b. Todos os jobs estão disponíveis no tempo t = 0.

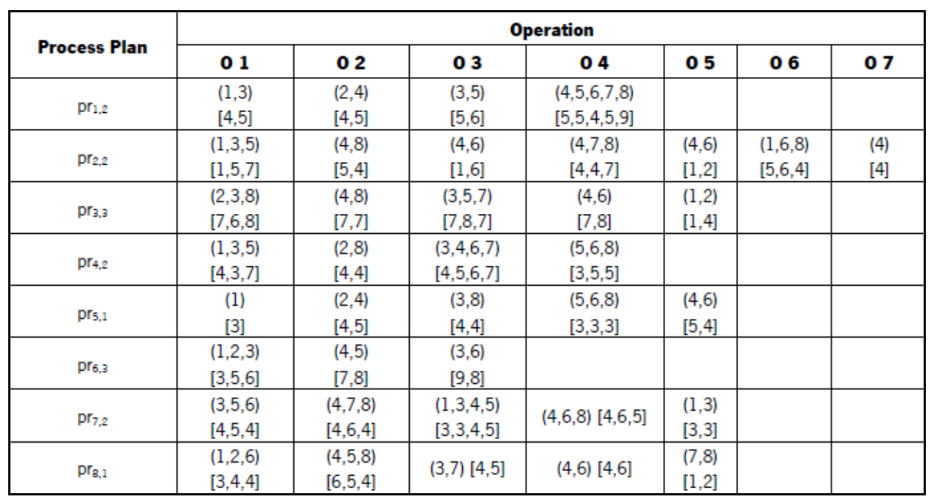
c. Cada operação pode ser realizada por apenas uma máquina de cada vez.

d. Não há restrições de precedência entre as operações de diferentes jobs; portanto os jobs são independentes.

e. Uma operação, uma vez iniciada, não pode ser interrompida.

f. O tempo de transporte de jobs entre as máquinas e tempo para configurar a máquina para realizar uma determinada operação estão incluídos no tempo de processamento.

Um job é um processo de produção de uma instância de um produto específico que é definido por um process plan. Uma operação é uma tarefa individual que é alocada a uma máquina e está associada a um job específico. Uma máquina é um recurso capaz de executar operações, e por fim um process plan é uma lista ordenada de operações necessárias para concluir um job.

Tabela 1. Process Plan para um problema de escalonamento com dimensão 8x7 e 8 máquinas

A Tabela 1 incorpora os process plan com dimensão 8x7 para a produção de um produto, envolvendo a realização de 8 jobs (com um máximo de 7 operações) distribuídos por 8 máquinas. Cada linha da Tabela 1 apresenta a descrição da sequência das operações necessárias para cada tipo de job (um job representa a produção de um produto, por exemplo pr1,2). No caso do tipo de job pr1,2 (primeira linha do process plan), este requer a execução de 4 operações numa predeterminada ordem, i.e. 01, 02, 03 e 04. Para cada operação, o process plan indica quais são as máquinas onde a mesma pode ser realizada, bem como a respetiva quantidade de unidades de tempo necessária para a sua realização. A título de exemplo, a primeira operação (01) pode ser realizada na máquina 1 com uma duração de 4 unidades de tempo ou na máquina 3 com uma duração de 5 unidades de tempo. Cada job de um process plan é composto por n operações que podem ser encadeadas com outras operações de outros jobs, mas dentro do mesmo job necessitam ser executadas pela sua ordem, isto é, num job que tenha três operações, a operação 3 não pode ser iniciada sem que a operação 2 esteja finalizada, e esta por sua vez também não pode ser iniciada sem que a operação 1 esteja finalizada. O cálculo da distribuição das operações pelas máquinas terá de se basear na capacidade das máquinas poderem executar essa operação, e na ocupação destas.

## Fase 1

1. Definição de uma estrutura de dados dinâmica para a representação de um job com um conjunto finito de n operações;

2. Armazenamento/leitura de ficheiro de texto com representação de um job;

3. Inserção de uma nova operação;

4. Remoção de uma determinada operação;

5. Alteração de uma determinada operação;

6. Determinação da quantidade mínima de unidades de tempo necessárias para completar o job e listagem das respetivas operações;

7. Determinação da quantidade máxima de unidades de tempo necessárias para completar o job e listagem das respetivas operações;

8. Determinação da quantidade média de unidades de tempo necessárias para completar uma operação, considerando todas as alternativas possíveis;

# Fase 2

1. Definição de uma estrutura de dados dinâmica para representação de um conjunto finito de m jobs associando a cada job um determinando conjunto finito de operações;

2. Armazenamento/leitura do ficheiro de um process plan (representar para efeito de teste o process plan da Tabela 1);

3. Inserção de um novo job;

4. Remoção de um job;

5. Inserção de uma nova operação num job;

6. Remoção de uma determinada operação de um job;

7. Edição das operações associadas a um job;

8. Geração de uma solução para o FJSSP, apresentando a distribuição das operações pelas várias máquinas, e determinando o menor makespan (unidades de tempo necessárias para realização de todos os jobs);

# Estrutura de Dados

Antes de começar a fazer o trabalho, tive de pensar como iria planear as minhas Estrutura de Dados, para que tornasse mais fácil para mim, como programador, perceber as tais.

Comecei por realizar que iria precisar de pelo menos três Structs, fazendo então uma para as Máquinas, outra para as Operações e uma última para Jobs, que apesar de não ser usada muito na primeira fase ela está lá.

A Primeira fez que as escrevi, ainda estava a começar a perceber Listas, então escrevi-as sem um apontador a apontar para o conteúdo seguinte, ao ponto que quando comecei a programar o trabalho não conseguia pensar como é que o iria fazer, então decidi fazer umas pesquisar e ver o que o Professor fez nas aulas e consegui perceber então como é que as Listas funcionavam.

Acabei por ter na mesma três Structs como tinha pensado no Inicio, criei então uma para as Máquinas, contendo o Id da Maquina, o Tempo que a Maquina leva a fazer certa Operação, e então um apontador a apontar para o “Next”, fazendo assim uma lista.

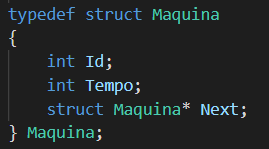


Figura 1. Estrutura de Dados usada para as Máquinas

As Operações e os Jobs, são muito parecidas, sendo a única diferença o facto de que os Jobs criam uma lista de Operações, enquanto as Operações criam uma lista de Maquinas, sendo que os dois têm o Id de cada um e um apontador a apontar para o seguinte da Lista.

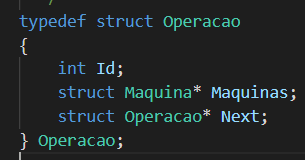


Figura 2. Estrutura de Dados usada para as Operações

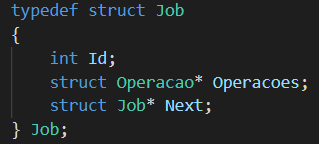
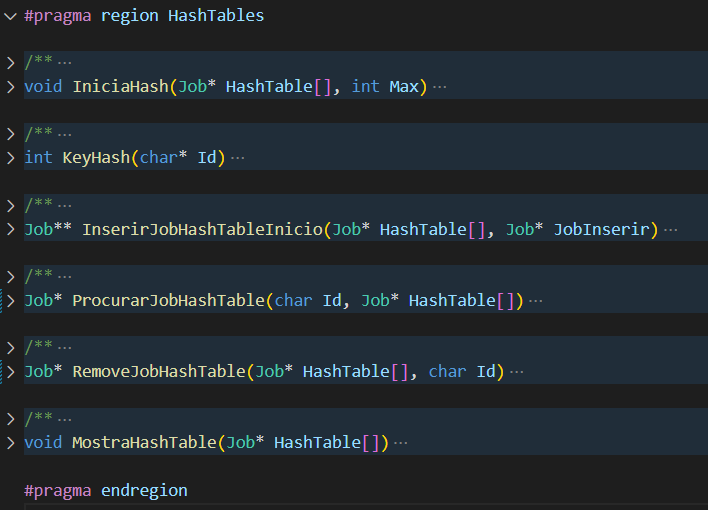


Figura 3. Estrutura de Dados usada para os Jobs

Para a segunda fase deste trabalho realizei que era preciso fazer uma estrutura de dados dinâmica para os jobs, resultando depois em remover, adicionar e alterar um job.

Essa estrutura de dados dinâmica que decidi usar foi a Hash Table, substituindo quase todas as funções que já tinha feito para o trabalho que envolviam Jobs.

Figura 4. Região da Hash Table no projeto

Tendo dificuldades mais no facto de que a Struct do Job teve de mudar, uma mudança simples, mas que mudou uma boa parte do trabalho:

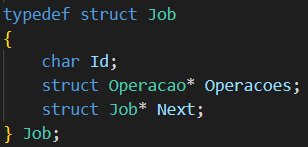


Figura 5. Estrutura do Job atualizada para a Fase 2

Essa pequena mudança foi o facto de que o Id passou de int para char, isso fez com que eu mudasse todas as funções que precisavam do Id do Job, a não ser isso o problema maior foi compreender como funcionavam as HashTables, mas acho que agora consigo trabalhar nelas.

# Testes Realizados

Ao fazer alguns testes fui encontrando erros, um deles tinha a ver com a criação de uma Operação, em que quando lhe juntava as Maquinas desejadas e pedia para mostrar as Listas, o erro mostrava isto:

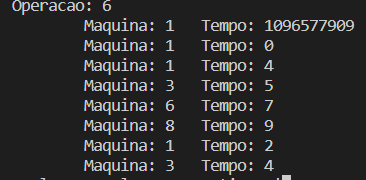


Figura 6. Erro com a Criação de Operações

Este erro foi facilmente resolvido quando percebi que a lista de Maquinas que estava a fornecer à Função já tinha algumas Maquinas á escritas, sendo resolvido com um “headListaMaquinas = NULL” no início da Função.

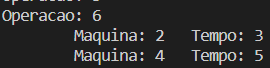


Figura 7. Erro Resolvido com a Criação de Operações

Outro erro ocorreu quando tentei ver se os Cálculos estavam bem, isto é, se o tempo máximo, mínimo e médio estavam bem, descobri logo que não já que o que recebi no terminal foi:



Figura 8. Erro nos Cálculos

O Erro foi um erro difícil de encontrar, mas que fazia completamente sentido pois em pensava que “Maquina->Next”, fazia com que a Máquina se passa a ser o Next, mas não, após a descoberta de tal erro andei a mudar tudo para “Maquina = Maquina->Next”, corrigindo assim o erro.

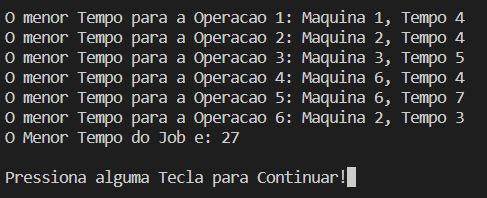
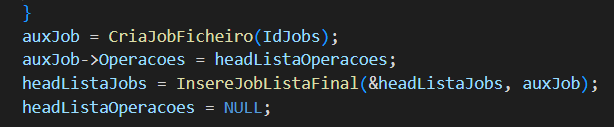


Figura 9. Erro dos Cálculos Corrigido

Na Segunda Fase ocorreram uns poucos erros sendo que os maiores foram notados na parte dos ficheiros, a não ser isso não houve muitos outros. O problema dos Ficheiros Designava-se no facto:





De que eu tinha um fscanf depois do while, isso causava o facto de ele criar Jobs se eles estarem no Ficheiro, ao reparar nisso mudei e ficou bem.

# Conclusão

Com a Primeira Fase deste Trabalho consegui melhorar o meu conhecimento sobre Estrutura de Dados, Listas e em si a Linguagem de Programação C. Aprendi também que ter um código bom e melhor organizado é melhor do que ter um código que funciona.