Relatório Projeto EDA

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

2021/2022

Índice

[Introdução 1](#_Toc99735979)

[Propósitos e Objetivos 2](#_Toc99735980)

[Estruturas de dados 5](#_Toc99735981)

[Testes Realizados 8](#_Toc99735982)

[Conclusão 10](#_Toc99735983)

Índice de imagens

[Figura 1-Struct Machine 5](#_Toc99714673)

[Figura 2-Struct Operations 6](#_Toc99714675)

[Figura 3-Struct Jobs 7](#_Toc99714677)

[Figura 4-Teste realizado #1 9](file:///P:\Escola\SegundoSemestre\EDA\Aulas\TrabalhoPraticoUm\Documentação\RelatórioProjeto%20EDA.docx#_Toc99714679)

[Figura 5-Teste realizado #2 9](file:///P:\Escola\SegundoSemestre\EDA\Aulas\TrabalhoPraticoUm\Documentação\RelatórioProjeto%20EDA.docx#_Toc99714680)

[Figura 6-Teste realizado #3 10](file:///P:\Escola\SegundoSemestre\EDA\Aulas\TrabalhoPraticoUm\Documentação\RelatórioProjeto%20EDA.docx#_Toc99714681)

Introdução

Este trabalho foi proposto pelo professor da unidade curricular Estrutura de dados Avançadas, Luis Ferreira integrada o segundo semestre do 1º ano de licenciatura de Desenvolvimento de Jogos Digitais, que visa o reforço e a aplicação dos conhecimentos adquiridos ao logo do semestre.

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. 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, j realizada na máquina em Mi, j é pi, j, k. As seguintes suposições são consideradas num problema FJSSP:
   1. Todas as máquinas estão disponíveis no instante t = 0.
   2. Todos os jobs estão disponíveis no tempo t = 0.
   3. Cada operação pode ser realizada por apenas uma máquina de cada vez.
   4. Não há restrições de precedência entre as operações de diferentes jobs; portanto os jobs são independentes.
   5. Uma operação, uma vez iniciada, não pode ser interrompida.
   6. 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.

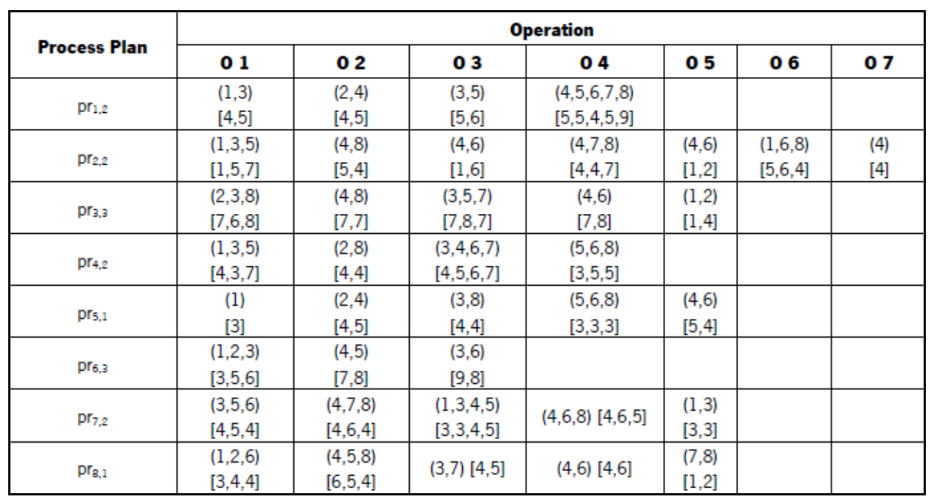
A Tabela 1 incorpora os *process plan* com dimensão 8x7 para a produção de um produto, envolvendo a realização de 8 jobs (com um máximo de 7 operações) distribuídos por 8 máquinas. Cada linha da Tabela 1 apresenta a descrição da sequência das operações necessárias para cada tipo de job (um job representa 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.

Tabela - Process plan para um problema de escalonamento com dimensão 8x7 e 8 máquinas

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;

Estruturas de dados

Criei uma estrutura de dados para as máquinas, onde é guardado o id, o valor de tempo e a proxima máquina.

Assim conseguindo guardar todas as maquinas de uma certa operação.

Texto

Descrição gerada automaticamente

Figura 1-Struct Machine

Criei uma estrutura operation que guarda o num, um apontador de uma estrutura machine e um apontador para a proxima operation.

Assim conseguindo guardar todas as operações de um job com as suas respetivas máquinas.

Texto

Descrição gerada automaticamente

Figura 3-Struct Operations

Criei um struct job que guarda o id, um apontador para uma struct operation e um para o proximo job.

Assim consegui guardar todos os dados necessarios em uma struct.

Texto

Descrição gerada automaticamente

Figura 3-Struct Jobs

Testes Realizados

Na Figura 4 observei o tempo mínimo que o job 2 pode demorar a completar.  
Na Figura 5 observei o tempo máximo que o job 2 pode demorar a completar.

Texto

Descrição gerada automaticamente  
  
  
Texto

Descrição gerada automaticamente

Figura 5-Teste realizado #2

Figura 4-Teste realizado #1

No 3º teste consegui observar o tempo médio que a operação 3 do job 2 pode demorar a completar.

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

Figura 6-Teste realizado #3

Conclusão

Com esta primeira fase foi possível aprofundar mais conhecimentos acerca de estruturas de dados avançadas e também sobre listas ligadas. Aprendi também que fundamentalmente que ter um bom código e bem documentado é melhor que ter um código que funciona.