



Relatório

Estruturas de Dados Avançadas

Aluno: André Freitas nº21112

Professor/es: João Carlos

Luís Gonzaga

Licenciatura em Engenharia de Sistemas Informáticos

Barcelos, junho, 2022

Resumo:

Trabalho individual que visa a criação de um projeto em C de forma a resolver um problema denominado por “Flexible Job Shop Problem”.

Índice

1. Introdução.....	1
1.1. Propósito	1
1.2. Objetivo	1
2. Estrutura de Dados	2
3. Testes realizados.....	3
3.1. Inserção de jobs	3
3.2. Remoção de jobs.....	4
3.3. Inserir uma operação num Job	4
3.4. Remover uma operação de um determinado Job	5
3.5. Alteração de uma operação.....	5
3.6. Problema de escalonamento	6
4. Conclusão	8
5. Bibliografia	8

1. Introdução

1.1. Propósito

Este trabalho prático de realização individual da Unidade Curricular (UC) Estruturas de Dados Avançadas (EDA) visa o reforço e a aplicação dos conhecimentos adquiridos ao longo do semestre.

1.2. Objetivo

O objetivo deste trabalho prático 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

2. Estrutura de Dados

Para esta fase do trabalho foram criadas três estruturas. Estas serão: “Maquina”, “Job” e “Cel”.

Na estrutura “Maquina” iremos alocar as máquinas, as operações e os tempos de cada máquina. Além disso teremos dois apontadores (“next” e “prev”) que irão permitir percorrer a lista para a direita e para a esquerda.

```
typedef struct Maquina{  
    int maquina;  
    int op;  
    int tempo;  
    struct Maquina* next, *prev;  
}Maquina;
```

Na estrutura “Job” iremos armazenar os jobs. Tal como a estrutura anterior irá ter apontadores “next” e “prev”, a diferença é que irá ter dois apontadores para a estrutura “Maquina” denominados “first” e “last”.

```
typedef struct Job{  
    int job;  
    struct Job *next, *prev;  
    struct Maquina *first, *last;  
}Job;
```

Por último na estrutura “Cel” iremos criar uma matriz encarregue de guardar as células da tabela do planeamento. Essas células são constituídas pelo “idjob”, “idop” e “inicio”.

```
typedef struct Cel{  
    int idjob;  
    int idop;  
    int inicio;  
}Cel;
```

3. Testes realizados

3.1. Inserção de jobs

Para esta tarefa o programa ira adicionar na lista os jobs que o utilizador deseja.

Além disso irá também listar toda a informação, bem como registá-la num ficheiro de texto que neste caso se chama "dados.txt". Além disso também foi feita uma função encarregue ler qualquer ficheiro de texto solicitando esta informação.

Por exemplo neste caso o utilizador inserir 4 operações.

```
Insira um Job: 1, 4, 5, 6
Insira um Job: 2, 3, 1, 7
Insira um Job: 3, 1, 2, 9
Insira um Job: 4, 1, 3, 11
```

Depois de aceder o menu escolhemos a opção relativa a esta tarefa e como podem ver todas as inserções forma listadas corretamente de forma ordenada. Além disso é criado o ficheiro de texto aonde também fica registado todas as inserções.

```
1, 1, 1, 6
1, 1, 1, 6
2, 3, 4, 5
2, 1, 2, 3
3, 4, 5, 6
3, 1, 3, 4
4, 2, 3, 8
4, 1, 4, 7
```

```
4, 1, 3, 11
3, 1, 2, 9
2, 3, 1, 7
1, 4, 5, 6
1, 1, 1, 6
1, 1, 1, 6
2, 3, 4, 5
2, 1, 2, 3
3, 4, 5, 6
3, 1, 3, 4
4, 2, 3, 8
4, 1, 4, 7
```

3.2. Remoção de jobs

Para remover os jobs o programa pede a identificação do job a remover e remove todos os processos com esse job associado. Como demonstrado em baixo.

2, 3, 4, 5	Qual Job deseja remover: 2
2, 1, 2, 3	3, 4, 5, 6
3, 4, 5, 6	3, 1, 3, 4
3, 1, 3, 4	4, 2, 3, 8
4, 2, 3, 8	4, 1, 4, 7
4, 1, 4, 7	1, 3, 2, 5
1, 3, 2, 5	3, 3, 2, 7
2, 4, 5, 1	4, 3, 9, 10
3, 3, 2, 7	
4, 3, 9, 10	

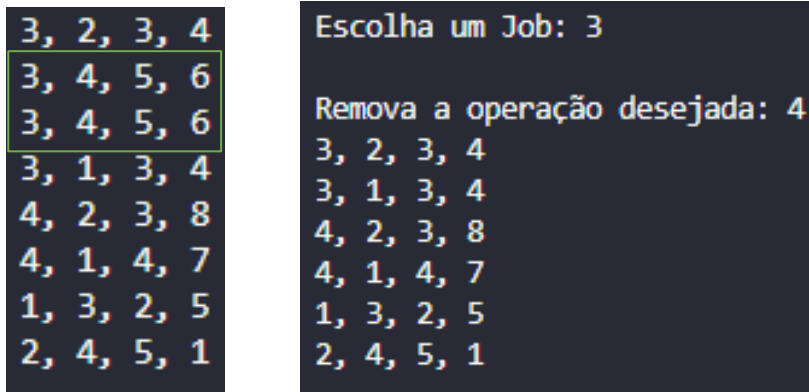
3.3. Inserir uma operação num Job

Em semelhança esta função também insere uma operação á escolha do utilizador a única diferença é que o programa primeiro verifica se o Job na qual esta operação vai ser inserida existe.

<pre>if (inicio->job != jobid) { printf("Job não foi encontrado"); exit(0); }</pre>	<pre>3, 4, 5, 6 3, 4, 5, 6 3, 1, 3, 4 4, 2, 3, 8 4, 1, 4, 7 1, 3, 2, 5 2, 4, 5, 1</pre>
<pre>Em qual Job deseja inserir a operação: 3 !Escreva todos os campos! Insira a operação: 2,3,4 3, 2, 3, 4 3, 4, 5, 6 3, 4, 5, 6 3, 1, 3, 4 4, 2, 3, 8 4, 1, 4, 7 1, 3, 2, 5 2, 4, 5, 1</pre>	

3.4. Remover uma operação de um determinado Job

À semelhança da remoção de um Job esta tarefa passa por requisitar o utilizador no sentido, de saber qual Job este deseja retirar todas as operações associadas ao Id da operação escolhida pelo mesmo.



```

3, 2, 3, 4
3, 4, 5, 6
3, 4, 5, 6
3, 1, 3, 4
4, 2, 3, 8
4, 1, 4, 7
1, 3, 2, 5
2, 4, 5, 1

Escolha um Job: 3

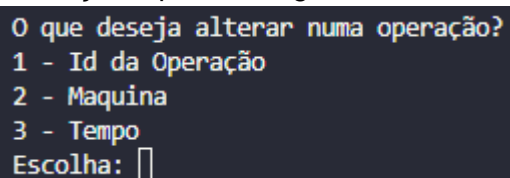
Remova a operação desejada: 4
3, 2, 3, 4
3, 1, 3, 4
4, 2, 3, 8
4, 1, 4, 7
1, 3, 2, 5
2, 4, 5, 1

```

3.5. Alteração de uma operação

Provavelmente uma das tarefas mais extensas pois, passa por avaliar três campos de uma operação. Sendo eles o próprio Id da operação, uma máquina associada a ela e o tempo de uma máquina ligada a essa mesma operação.

Começa-se por interrogar o utilizador sobre o que este quer mudar.

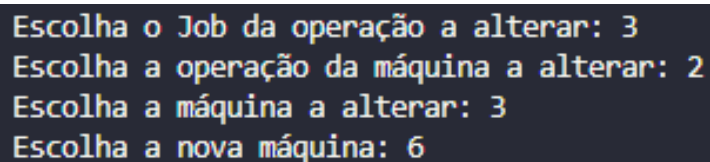


```

0 que deseja alterar numa operação?
1 - Id da Operação
2 - Máquina
3 - Tempo
Escolha: 

```

Depois para cada operação é pedido ao utilizador a identificação de cada campo. Como pode-se verificar no exemplo abaixo.



```

Escolha o Job da operação a alterar: 3
Escolha a operação da máquina a alterar: 2
Escolha a máquina a alterar: 3
Escolha a nova máquina: 6

```

Isto aplica-se pra todas operações sendo que no tempo não é pedido o tempo antigo, simplesmente o utilizador muda para um tempo novo.

3.6. Problema de escalonamento

Esta tarefa é sem dúvida a carne e osso desta fase. É pedido para criar uma proposta de escalonamento para o problema FJSSP apresentando a distribuição das operações pelas várias máquinas. Além disso exportar tudo para um ficheiro de interpretação intuitiva, como por exemplo representação gráfica HTML. Esta foi a maneira proposta de resolução.

		Tempo							
		t ₁	t ₂	t ₃	t ₄				t _T
Máquinas	m ₁	O _{1J1}	O _{1J1}	O _{1J1}					
	m ₂								
	m ₃	O _{1J2}	O _{1J2}		O _{2J1}	O _{2J1}	O _{2J1}		
	m ₄								
				O _{2J2}	O _{2J2}	O _{2J2}	O _{2J2}		
								O _{3J2}	O _{3J2}
	m _n							O _{3J1}	O _{3J1}

Para atender a esta resolução foi necessário criar uma matriz composta por linhas de máquinas e colunas de tempo.

```
Cel plano[M][T];
```

De seguida foi necessário criar o plano daí verificarmos se existe já linhas e colunas para depois poder introduzi-las.

```
void IniciaPlano(Cel p[][T], int codJob, int codOper) {
    for (int l = 0; l < M; l++){
        for (int col = 0; col < T; col++) {
            p[l][col].idjob = codJob;
            p[l][col].idop = codOper;
        }
    }
}
```

Criada a estrutura da tabela é preciso inserir as células armazenadas na matriz.

```
for (; col < totTempo; col++) {
    p[mId][col].idjob = c->idjob;
    p[mId][col].idop = c->idop;
    p[mId][col].inicio = c->inicio;
}
```

Com isto estamos prontos a criar uma linha na tabela do planeamento.

```
Ocupa(Cel p[][T], int mId, int tempo, int totTempo, int codJ, int codO)
```

Mostrando na prática para se perceber melhor. Irá ser pedido ao utilizador para escolher uma máquina, o início do processo dessa respetiva máquina, tal como o final do mesmo. Para além disso obviamente é necessário identificar um Job e uma operação.

```
Escolha uma máquina: 4
Escolha o início do processo: 11
Escolha o final do processo: 14
Escolha um Job: 3
Escolha uma operação: 1

Plano: Máquina: 4, Início: 11, Final: 14, Célula: Job - 3 || Operação - 1
```

Todas estas inserções são guardadas num ficheiro csv, que irá ser posteriormente usado para produzir uma tabela html.

```
plano.csv
1 M1, Job 1 - Operação 1, 1, 5
2 M2, Job 1 - Operação 2, 5, 8
3 M3, Job 2 - Operação 1, 8, 11
4 M4, Job 3 - Operação 1, 11, 14
5
```

Através do ficheiro “plano.html”, usando uma template da Google assistida por dois ficheiros JavaScript, é possível produzir um front end básico para a proposta de escalonamento armazenada no ficheiro “plano.csv”.

Planeamento

M1	Job 1 - Operação 1
M2	Job 1 - Operação 2
M3	Job 2 - Operação 1
M4	Job 3 - Operação 1

4. Conclusão

O grau de dificuldade desta fase para a primeira é muito maior, mas é algo que eu estava à espera. Se calhar poderia ter saído melhor o trabalho, mas devido ao aperto de horários isso não é possível. Mesmo assim estou orgulhoso do meu esforço.

5. Bibliografia

Repositório do professor Luís:

- [GereMaquinas](#)
- [Planeamento](#)
- [PlaneamentoChart](#)

Eventuais dúvidas que apareceram ao longo do projeto foram resolvidas junto do professor ou com que recurso dos sites [Stack Overflow](#) e [W3Schools](#).