Trabalho 2 - Escalonamento de transações

Alunos:

Lucas Block Villatore GRR20171677

Leonardo Bueno Nogueira Kruger GRR20180130

Algoritmos e convenções utilizadas no teste de seriabilidade.

Manipulação do grafo: **Matriz de adjacencia.**

Verificação de ciclo: **Busca de profundidade.**

Funções importantes

leituraArquivo: Faz a leitura do arquivo de entrada salvando as entradas e separando em escalonamentos (linhas para a saida).

```
TransactionT **leituraArquivo(int *limiar, EscalonadorT ***escalonador, int
*numero_escalonacoes)
```

aloca_grafo: Aloca memoria e produz a matriz de adjacencia para manipulação do grafo, ocorre um for entre as operações de transação de modo a testar todas as possibilidades dois a dois de modo a verificar se ocorre uma operação de escrita e alguma outra operação no mesmo atributo adicionando uma aresta entre os vertices, caso não esteja ocorrendo uma escrita, atributos das operações serem diferentes ou operação de commit é ignorado e continua para proxima iteração até acabar.

```
int **aloca_grafo(int numero_vertices, int *tarefa_escalonada, int
numero_transacoes, TransactionT **transacoes)
```

tem_ciclo: Dentro desta função ocorre a busca de profundidade dentro do grafo enviado, são visitados todos os vertices e caso um vertice seja visitado mais de uma vez retorna o valor 1 que significa que existe ciclo no grafo e 0 caso não exista.

```
int tem_ciclo(int **grafo, int vertice, int tamanho_grafo, int **nos_visitados)
```

```
int visao_equivalente(int *escalonacao, int inicio, int fim, TransactionT
**transacoes, int quantidade_transacoes)
```

visao_equivalente: Nessa função ocorre a permutação das transações presentes no escalonamento (ex:[1,2],[2,1]) e chama a função testa_visao_equivalente.

```
int testa_visao_equivalente(TransactionT **transacoes, int tamanho_transacoes,
int *escalonacao, int tamanho_escalonacao)
```

testa_visao_equivalente: A partir das operações em memoria e do escalonamento dado (ex: [1,2],[2,1]) monta agendas e realiza os testes para verificar equivalencia, a regra nº 2 verifica se é leitura em seguinda permuta com as outras operações verificando se o identificador é diferente, se é uma escrita e se é no mesmo atributo para verificar a partir do tempo das duas se a ordem da agenda original foi violada. A regra Nº 3 para cada ultima escrita numa Tansação(i) é verificada se permaceu como ultima operação da mesma transação na agenda S'.

Como executar:

```
$ make
$ ./escalona < entradas/input1.in > output.out
$ diff output.out entradas/input1.out
```