ALGORITMO DO BANQUEIRO

Foi um problema descrito e solucionado por Dijkstra, é um algoritmo que trata a questão de resolução de impasses, o problema proposto por Dijkstra é que um banqueiro de uma pequena cidade pode negociar com um grupo de cliente para os quais libera linhas de crédito, e o algoritmo irá verificar se a liberação de uma requisição pode gerar dois estados, seguro que é quando se tem recursos suficientes para fazer a liberação de todas as solicitações podem ser atendidas com os recursos disponíveis, caso contrário será gerado um estado inseguro.

O algoritmo é aplicado para recursos simples e recursos múltiplos. Para recursos simples pode ser visto como uma tabela, na qual duas colunas representam a quantidade de recursos solicitados por processo e a quantidade máxima que o sistema dispões para o processo a quantidade de recursos que tem alocadas para os mesmos e em outra coluna a quantidade máxima de recursos que o ele pode precisar, se o sistema consegue escalonar os recursos de uma forma que todos os processos sejam finalizados, isso é dado como um estado seguro, caso contrário em um estado inseguro o sistema não pode garantir que todos os processos serão ser executados.

Exemplos:

a) Estado seguro

	Possui	Máximo
A	3	9
B	2	4
C	2	7
Recursos Disponíveis: 2		

Recursos totais: 10

b) Estado inseguro

	Possui	Máximo
A	4	9
B	4	4
C	2	7
Recursos Disponíveis: 0		

Recursos totais: 10

Algoritmo do banqueiro para recursos múltiplos é a forma mais comum de se apresentar o algoritmo, pois é mais próximo a realidade dos sistemas operacionais modernos, nos quais se necessita orquestras vários processos e vários recursos. A forma de se usar é a mesma, contudo para o algoritmo com recursos compartilhados são utilizadas 3 matrizes PxR (processos por recursos), que são as matrizes de recursos necessários, recursos alocados e recursos disponíveis.

Exemplo:

Recursos alocados:

	A	**B**	**C**	**D**
P1	1	2	2	1
P2	1	0	3	3
P3	1	1	1	0

Recursos disponíveis:

A	**B**	**C**	**D**
3	1	1	2

Recursos Máximos:

	A	B	C	D
P1	3	3	2	2
P2	1	2	3	4
P3	1	1	5	0

Matriz de necessidade:

	A	B	C	D
P1	2	1	1	1
P2	0	2	0	1
P3	0	0	4	0

Pelas matrizes pode se observar que o recurso C está em estado inseguro pois a necessidade que vai ser exigida é maior do que a quantidade disponível, ou seja, não há uma forma de escalonar os recursos para resolver o problema. Dessa forma pode se concluir que o algoritmo do banqueiro é uma ótima opção para se prever os deadlocks que podem ocorrer entre os processos.

Exemplo da execução do algoritmo:

Valores de entrada:

```
int processos[] = {0, 1, 2, 3, 4};

int disponivel[] = {3, 3, 2};

int maximo[][R] = {{7, 5, 3},
                   {3, 2, 2},
                   {9, 0, 2},
                   {2, 2, 2},
                   {4, 3, 3}};

int alocado[][R] = {{0, 1, 0},
                    {2, 0, 0},
                    {3, 0, 2},
                    {2, 1, 1},
                    {0, 0, 2}};
```

Saída:

```
System is in safe state.
Safe sequence is: 1 3 4 0 2
```

Essa é a ordem que os processos serão executado para que não haja o deadlock no sistema.