MAC0122 - Relatório EP4

Felipe Fernandes de Lima, Nº: 9345321

Para o EP4, havia a possibilidade de se fazer tanto um algoritmo online (considerando entradas em tempo real) como um offline (recebendo todas as entradas previamente). A implementação que utilizei foi de um algoritmo online.

• TRATAMENTO DO TEMPO

O funcionamento do algoritmo ocorre em rodadas, onde cada rodada corresponde a aos acontecimentos de um instante específico de tempo e é representada por uma repetição dentro de um laço do tipo *while*. Cada entrada é dividida em três etapas: entrada, saída e atualizações. O tempo é controlado por uma variável de crescimento constante (*tempo_atual* que aumenta 1 a cada rodada).

Entretanto, como foi possível perceber pela sugestão de teste contida no enunciado, os tempos de entrada (instantes em que cada entrada chega ao aeroporto) não necessariamente seguirão uma ordem linear (uma entrada pode acontecer em 2, e a próxima, só em 5). Para lidar com isso, a cada rodada é feita uma comparação entre os tempos relativos à última entrada e ao tempo atual:

• Se o tempo atual for maior, é aguardada uma nova entrada do usuário, da qual inicialmente só será lido o tempo, que se tornará o novo valor de tempo de entrada;

•Se ambos forem iguais, o resto da entrada é lido e analisado;

• Se o tempo da última entrada for maior, a leitura do resto das informações é adiada até que os tempos sejam iguais e, enquanto isso, as rodadas não terão a etapa de entrada.

• A LISTA LIGADA

Para armazenar os dados dos aviões foi utilizada uma única lista ligada simples, cujas células (chamadas *aviões*) são compostas pelos seguintes atributos:

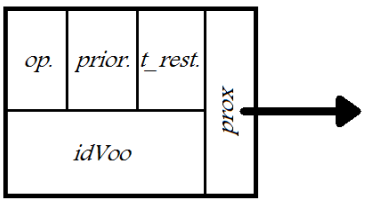
• *idVoo* (identificação do avião);

• *operacao* (pouso ‘P’ ou decolagem ‘D’);

• *prioridade* (0 ou 1);

• *t_restante* (tempo restante);

• *prox* (ponteiro para próxima célula).



Célula do tipo avião

Os aviões já serão inseridos na lista conforme as regras de prioridade e ordenação escolhidas. Portanto, os que receberão autorização para pousar ou decolar a cada rodada em que o aeroporto estiver aberto serão sempre os primeiros da lista. O resultado é que a maior parte das decisões tomadas pelo algoritmo ocorre no processo de inserção em vez de durante a busca.

• ORDENAÇÃO DA FILA DE AVIÕES

Para este EP4, uma das partes mais centrais do algoritmo é a disposição dos aviões na fila de espera, pois ela quem define quais poderão pousar/decolar primeiro e, assim, decide a ordem de entrada e saída no aeroporto.

A ordenação dos aviões na lista (fila de espera) ocorre durante o processo de inserção (na função *insereNaFila*) e é definida por quatro critérios:

- **Prioridade:** como definido no próprio enunciado, os aviões atendidos pelo aeroporto já possuirão algum nível de prioridade inicial (1 para prioritários, 0 para não-prioritários). Esse será o primeiro critério de ordenação. Durante toda a execução do programa os aviões de prioridade 1 estarão nas primeiras posições da lista, à frente dos de prioridade 0.

- **Tempo Restante:** os aviões com a mesma prioridade serão dispostos conforme o tempo que podem esperar. Os que possuírem o atributo *tempo_restante* menor serão postos antes daqueles cujo mesmo atributo seja maior.

- **Pouso / Decolagem:** caso haja dois aviões na fila com mesma prioridade e que podem esperar o mesmo tempo, se um estiver aguardando autorização para pouso e o outro para decolagem, será dada prioridade ao que quer pousar. Esse critério foi definido por uma inferência pessoal. Em uma situação real, seria mais perigoso deixar aviões esperando para descer, pois se forem deixados em longa espera podem cair, do que esperando para levantar voo, pois, uma vez estando em terra, não correm riscos tão grandes.

- **Ordem de chegada:** por fim, se para um conjunto de mais de um avião todos os outros critérios forem iguais (mesma prioridade, mesmo tempo restante e mesma operação), a ordem deles na fila de espera será definida pela sua ordem de chegada e, assim, estarão na frente aqueles que forem inseridos primeiro na lista.

•CASOS DE URGÊNCIA

Na implementação feita para este EP foi considerado importante minimizar o uso da pista auxiliar. A participação dela ficou restrita a situações de urgência, que serão as únicas exceções para as regras de ordenação escolhidas. Quando algum avião com prioridade 0 estiver muito próximo de esgotar seu tempo de espera, ele será retirado da fila, receberá prioridade 1 e será reinserido conforme as regras citadas acima.

O tempo restante considerado limite para um avião permanecer com prioridade 0 (tempo restante de urgência) será inicialmente de 2. Esse valor foi escolhido ponderando-se entre as percepções de que tempos de urgência maiores aumentam a segurança para os aviões, pois fazem com que mais aviões com tempo restante curto decolem mais cedo e evita que sejam surpreendidos por fechamentos repentinos do aeroporto, mas também de que tempos maiores ignoram a função da prioridade inicial com que os aviões são inseridos no programa, pois misturam muito rápido aviões prioritários com inicialmente não-prioritários.

Entretanto, 2 é apenas um valor inicial, que pode ser alterado caso seja verificado em alguma etapa que há muitos aviões – digamos n – com tempos restantes iguais. Nesse caso, alterando-se o valor do tempo restante de urgência para $\frac{n}{2}$ é possível fazer com que eles, caso tenham que esperar muito, consigam ser autorizados ainda em tempo suficiente para que todos decolem (seriam dois a dois por rodada, utilizando-se ambas as pistas), contanto que o aeroporto não feche repentinamente, e não haja (ou surjam) outros aviões em mais urgência.

Com essa medida, é possível reduzir em grande quantidade a chance de perda de aviões, sem um risco expressivo de invalidar as prioridades iniciais, pois isso só aconteceria se um número muito grande de aviões estivesse com o mesmo tempo limite de espera, o que não é muito provável.

Para ilustrar isso melhor, podemos considerar o exemplo de 8 aviões da lista que em determinada rodada de tempo estão com o mesmo tempo restante. Para esta situação, já difícil de ocorrer, o tempo restante a partir do qual aviões ganhariam prioridade seria de 4, que ainda não atrapalha tanto a função do atributo prioridade. Além disso, como essa verificação é refeita por rodada, assim que os aviões de tempo repetido começarem a ser autorizados, o tempo restante de urgência vai ser revisto, podendo retornar a valores que interfiram menos na prioridade inicial.

A verificação de urgência e a definição do número máximo para que um avião não-prioritário continue nessa condição são feitos respectivamente pelas funções *buscaUrgencia* e *defineUrgencia*.

• ESTRUTURA DE UMA RODADA

Como já foi dito, cada rodada de execução do programa (que equivale a uma unidade de tempo real para o aeroporto) será dividia em três etapas, que serão a base da operação do aeroporto.

A primeira delas será a fase de **entrada**, na qual será recebida uma requisição do usuário, que pode ser tanto para abertura ou fechamento do aeroporto, como para pouso ou decolagem de um avião. Para entradas que se refiram ao estado do aeroporto será feita a alteração solicitada na variável *aberto* (ela valerá 0 para a situação *fechado*, 1 para *aberto* e -1 para o *fim da operação do aeroporto*). Já no caso de serem requisições de aviões, as informações destes serão lidas e eles serão inseridos de forma ordenada na lista. A entrada ainda pode não ocorrer para determinadas rodadas, quando for verificado que a última requisição feita se refere a um instante posterior ao atual.

A segunda etapa de cada rodada será a de **saída**, na qual será dada a resposta correspondente à entrada obtida ou às que foram acumuladas na lista (fila de espera de aviões). Inicialmente é verificado se o tipo da entrada atual foi A, F ou Z. Para esses casos a saída será a mensagem correspondente: respectivamente "aeroporto aberto" ou "aeroporto fechado". Caso contrário, se o tipo da entrada for P, D ou nula ('\0' – quando não há entrada), é verificado se o aeroporto está aberto e se a fila não está vazia. Em caso afirmativo para ambas as verificações, será dada autorização para algum avião ou para dois, em caso de urgência. Em caso negativo para alguma, não haverá saída para aquela rodada.

A terceira e última etapa de uma rodada é a de **atualizações**, cuja função é simplesmente realizar as ações finais para que se possa passar à rodada seguinte. Na etapa de atualizações o tempo restante de cada avião é decrementado em 1, os aviões que atingirem tempo restante 0 são retirados da lista e o tempo atual é incrementado em 1.

• LIMITAÇÕES

O algoritmo implementado foi pensado para funcionar bem nos casos que pareceram ser os mais comuns ou prováveis para a operação de um aeroporto, mas ele pode falhar ou não ser muito eficiente em várias situações.

Como a cada rodada em que o aeroporto está aberto pelo menos um avião é liberado, os principais motivos que levariam um avião a esperar muito tempo na fila seriam quando o aeroporto permanecesse muito tempo fechado, ou quando ele fosse um avião desfavorecido pelas regras de ordenação da fila de espera.

Por exemplo, um avião que fosse não-prioritário e tivesse tempo restante alto poderia passar muito tempo na fila se aparecessem sempre novos aviões prioritários ou com tempos restantes menores que o dele. Somente quando seu tempo restante se tornasse mais baixo, ele poderia chegar às primeiras posições da fila de não-prioritários ou ainda ser considerado caso de urgência e virar prioritário (mas ainda poderia competir com outros prioritários que estivessem em urgência).

Apesar das precauções que foram tomadas ao se definir os critérios de ordenação e urgência ainda existem casos que poderiam levar à perda de aviões. Como foi feita a opção por um algoritmo online, ele sempre corre o risco de ser surpreendido por situações como o fechamento do aeroporto em momentos inoportunos e/ou de durações prolongadas, ou número muito grande de aviões com tempo restante igual ou próximo. Nesses casos se torna mais difícil evitar a perda de algum avião.

• COMPLEXIDADE

Considerando n aviões na fila, e analisando cada uma das funções do programa é possível ver que a maior parte delas tem complexidade linear, o que se deve em grande parte à eficiência das listas ligadas para esse tipo de problema.

Função	Complexidade	Casos ruins (ou pior caso)
criaAviao	O(1)	
insereNaFila	O(n)	Inserir avião não prioritário no final da lista
removeDaFila	O(n)	Remover avião do final da lista
		Receber aviões com nomes muito parecidos
atualizaTempoRestante	O(n)	
buscaUrgencia	$O(n^2)$	Muitos aviões não prioritários na lista
		Muitos aviões em urgência
defineUrgencia	O(n)	
removePerdidos	O(n)	Muitos aviões perdidos na mesma rodada

No geral, o que mais afeta a velocidade do programa é a quantidade de aviões armazenados. Como a maioria das funções precisa percorrer boa parte ou completamente a lista de aviões, quanto mais aviões estiverem no programa, maior será o tempo para que ela seja percorrida.

As únicas exceções a essa tendência são as funções *criaAviao* e *buscaUrgencia*. A primeira tem complexidade constante, pois, para qualquer avião que se queira inserir no programa, o processo de criação de uma célula é sempre a mesmo. A segunda tem complexidade quadrática devido à necessidade de, além de se percorrer a lista inteira buscando casos de urgência, remover e reinserir aviões quando um caso é encontrado. Portanto, o pior caso seria quando todos ou praticamente todos estão em urgência ao mesmo tempo, pois, assim, seria necessário remover e reinserir (operações de O(n)) cada avião.

É possível concluir então, que a situação que mais exige em complexidade do algoritmo é quando se tem um grande número de aviões e boa parte deles possui o mesmo tempo restante. Como foi visto na seção anterior, essa mesma situação também é a que mais aumenta o risco de perda de aviões durante a execução.