UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

RODRIGO DE CASTRO MICHELASSI
NUSP: 13672703

Exercício-Programa 1: Gerenciamento de Aeroporto

São Paulo
2023

Sumário

1 Introdução

Nesse primeiro EP, havia o objetivo de criar um algoritmo, o mais eficiente possível para gerenciar os pousos e decolagens em um aeroporto, conforme determinadas medidas e situações.

É possível descrever aqui três situações possíveis, que foram administradas de diferentes formas. São elas:

- Avião usual chegando ao aeroporto e solicitando pouso
- Avião usual saindo do aeroporto e solicitando decolagem
- Avião de emergência chegando ao aeroporto e solicitando pouso imediato

O grande objetivo do EP é administrar esse aeroporto da melhor maneira possível, para que nenhum dos aviões caia, atrase e que todos os aviões de emergência pousem nesse aeroporto, ao invés de um outro aeroporto próximo, e, caso não seja possível, minimizar o máximo possível esses contra tempos.

2 Estrutura de Dados

Para resolver o problema que nos foi dado, é necessário utilizar uma estrutura de dados. Caso contrário, o código, que já é grande por si só, ficaria diversas vezes maior e desorganizado. Com base nisso, fazendo um levantamento de possíveis lógicas, as duas conclusões finais da estrutura de dados que mais se adaptavam ao problema seriam uma Fila de Listas Ligadas ou uma Fila de Prioridades.

Por motivos de adaptação do código e maior facilidade de implementação, foi escolhido usar uma Fila de Prioridades. A ideia aqui é que, a cada rodada o programa recebe K, K $\in \mathbb{N}$ novos aviões, com base na entrada do usuário. A princípio, nota-se que esse valor não pode ser tão alto, pois como nosso aeroporto possui apenas 3 pistas, se chegassem muitos aviões em determinado momento, o aeroporto não suporta tal número.

A fila de prioridades utilizada foi modificada para se adequar melhor ao problema dado. Como analisamos cada avião da fila, um a um, utilizamos o atributo "prioridade"para sabermos a situação do avião na fila, com prioridade 2, caso ele esteja sem combustível ou o voo tenha que sair imediatamente, prioridade 3 caso seja um avião de emergência e prioridade 0 caso contrário. Além disso, a remoção da fila não funciona de uma forma comum, retirando exatamente o avião que queremos, a depender da sua posição da fila, e não apenas o item de maior prioridade.

Por meio desses argumentos, chegamos a uma conclusão que a execução, embora feita por meio da entrada do usuário via teclado, deve ser feita com bom senso, para que o programa funcione corretamente para uma quantidade de novos aviões na proporção correta.

3 Algoritmo

Com base no que foi visto até o momento, o algoritmo apresentado segue os seguintes passos:

- Primeiramente, verificamos todos os aviões da lista que devem decolar **imediatamente**, visto que, caso isso não ocorra, ocorre um prejuízo para a empresa aerea. Essa verificação deve ser feita primeiro pois, apesar de existir casos de emergência, os aviões que devem decolar dependem obrigatoriamente do aeroporto que estamos gerenciando, e caso as pistas se ocupem, precisamos aguardar mais duas rodadas para que sejam usadas novamente.
- Em segundo lugar, tratamos dos aviões que estão chegando no aeroporto. É nossa obrigação pousar, também de forma imediata, os aviões sem combustível o suficiente para se manter no ar, e os aviões de emergência, os quais são sempre pousados primeiro. Obviamente, pela entrada tentamos controlar para que não haja uma grande necessidade de enviar inúmeros aviões para outros aeroportos.
- Por fim, tratamos dos aviões restantes para decolagem e pouso. Nesse algoritmo, foi escolhido deixar a decolagem apenas para o seu horário previsto, com tolerância sobre o seu tempo, e, caso hajam pistas disponíveis, pousamos no aeroporto os aviões independente do tempo que eles podem ficar no ar.

Vale lembrar que foi definido para a nossa fila 3 níveis de prioridade, dentre os quais no nível 1 se encontram os aviões que podem pousar ou decolar daqui algumas iterações, no nível 2 se encontram os aviões que precisam pousar ou decolar imediatamente e no nível 3 se encontram os aviões de emergência.

Usando dessa lógica conseguimos um resultado satisfatório ao analisar as saídas finais.

Vale lembrar que, por usar um número grande de objetos e realocação dinâmica da memória em diversos momentos, o algoritmo não funciona para grandes números de iterações, por questões de memória. Foi possível testar até a 250° iteração, ou seja, por 250 rodadas. Além disso, o programa apenas roda com pelo menos 3 aviões novos a cada rodada.

4 Testes Recomendados

Como já visto nesse documento, os nossos casos de testes procuram ser o mais objetivo possível e não ultrapassar limites do bom senso, tendo em vista que o nosso aeroporto possui apenas 3 pistas e, se houvesse a chegada de diversos aviões, o algoritmo não suportaria, causando um enorme fluxo de aviões para aeroportos vizinhos e atrasos nos voos. Seguem agora alguns exemplos de entradas e saídas que podem ser testadas. O programa funciona para, no mínimo, 3 aviões por rodada.

OBS: o modelo probabilístico utilizado não utiliza uma semente fixa, ou seja, não teremos sempre as mesmas respostas ao utilizar as mesmas entradas, o que leva a uma dinamização do algoritmo, todavia, ao realizar inúmeros casos de testes, as respostas em média permanecem semelhantes sempre.

Para a compilação do nosso programa, foi utilizado no terminal:

g++ -o ep1 ep1.cpp filaPrioridades.cpp aviao.cpp funcoes.cpp

Entrada 1:
15 20 13 14 15 16

Saída 1:
Quantidade de Aviões gerados: 315
Quantidade de Aviões de emergência gerados: 7
Quantidade de decolagens realizadas: 11
Quantidade de decolagens realizadas sem atraso: 9
Quantidade de pousos totais: 18
Quantidade de pousos em outro aeroporto: 14
Quantidade de emergências atendidas: 7

Podemos ver, que para esses dados de entrada o número de aviões novos por rodada (no máximo 20) sobrecarregou um pouco o nosso sistema, fazendo com que, em média, 14 a cada 18 pousos fossem no aeroporto vizinho, devido a superlotação do nosso aeroporto.

Todavia, o sistema se mostrou eficiente em evitar atrasos de decolagens e ao atender as emergências, mesmo que em outro aeroporto.

Entrada 2:

27 4 50 15 23 14

Saída 2:

Quantidade de Aviões gerados: 135

Quantidade de Aviões de emergência gerados: 10

Quantidade de decolagens realizadas: 13

Quantidade de decolagens realizadas sem atraso: 13

Quantidade de pousos totais: 31

Quantidade de pousos em outro aeroporto: 18

Quantidade de emergências atendidas: 9

Nesse novo caso, com 4 novos aviões por rodada, ainda mais do que o que é possível acumular na pista, temos uma média de 18 a cada 31 pousos em outro aeroporto, o que diminuiu a sobrecarga do nosso aeroporto, e nenhum voo atrasado. Vale lembrar que a sobrecarga persiste pois, a cada uso da pista, ela fica inoperante pelos próximos 2 turnos seguintes, todavia as emergências foram bem atendidas.

Entrada 3:

20 14 40 32 12 13

Saída 3:

Quantidade de Aviões gerados: 300

Quantidade de Aviões de emergência gerados: 37

Quantidade de decolagens realizadas: 3

Quantidade de decolagens realizadas sem atraso: 3

Quantidade de pousos totais: 66

Quantidade de pousos em outro aeroporto: 48

Quantidade de emergências atendidas: 24

Agora, usamos uma entrada que nos traz até 14 novos aviões por rodada no nosso aeroporto. Como consequência direta disso, com a disponibilidade de apenas 3 pistas operantes no nosso aeroporto, que, após a utilização ficam por 2 unidades de tempo paradas, tivemos um grande número de aviões tendo que pousar em um aeroporto vizinho.

Entrada 4:

38 3 40 20 14 10

Saída 4:

Quantidade de Aviões gerados: 152

Quantidade de Aviões de emergência gerados: 14

Quantidade de decolagens realizadas: 23

Quantidade de decolagens realizadas sem atraso: 21

Quantidade de pousos totais: 39

Quantidade de pousos em outro aeroporto: 24

Quantidade de emergências atendidas: 12

Agora com o número mínimo de aviões possíveis a cada rodada, vemos um desempenho muito melhor do algoritmo, que consegue evitar bem os atrasos e pousar a maior parte das emergências (vale lembrar que podemos ter mais emergências geradas do que atendidas pois algumas são geradas na última iteração, sem tempo para serem atendidas).

5 Considerações Finais

Realizando esse projeto foi possível revisar, estudar e modificar a estrutura de dados da Fila de Prioridades, o que é sempre um ótimo exercício para estudar o funcionamento de uma estrutura e entender seu código de forma mais aprofundada. Podemos ver que o algoritmo implementado é eficiente, em vista que não leva muito tempo para uma pequena quantidade de objetos, embora tenha um crescimento de duração muito rápido a medida que aumentamos o número de iterações.

Por fim, pode-se considerar que, apesar do algoritmo ser um pouco demorado para um número grande de iterações, devido às complexidades dos algoritmos da fila de prioridades e ao grande número de laços comparativos e números gerados aleatoriamente, a cada iteração, o algoritmo funciona de forma satisfatória para uma entrada que se aproxima da realidade e atinge às demandas do nosso aeroporto.