RELATÓRIO - EP1

MAC 0323

ALGORITMOS E ESTRUTURA DE DADOS II

ODAIR G. DE OLIVEIRA
13671581

ÍNDICE

1. 1/2 SOBRE O EXERCÍCIO

O relatório feito aqui é sobre o Exercício Programa 1 (EP1) da disciplina MAC0323 - Algoritmos e Estruturas de Dados II executado por Odair Gonçalves de Oliveira, N° USP: 13671581, durante o 3° semestre do Bacharelado em Ciência da Computação no Instituto de Matemática e Estatística (IME) da Universidade de São Paulo (USP).

O Exercício foi produzir um programa que simula o gerenciamento de um aeroporto. O aeroporto simulado possui 3 pistas sendo 1 delas usada exclusivamente para decolagem exceto em casos de emergência.

A cada instante de tempo o aeroporto recebe uma determinada quantidade de aviões, com determinados atributos. A tarefa do programa é gerenciar uma fila com esses aviões para que todos possam pousar ou decolar, sem que os aviões esperem por mais tempo do que podem esperar ou fiquem sem combustível.

A cada pouso ou decolagem que ocorre na pista, ela fica interditada durante 2 unidades de tempo, ou seja, só é possível pousar aviões de 3 em 3 unidades de tempo.

Os aviões podem ser classificados em normais e especiais, sendo que aviões especiais têm permissão para pousar ou decolar imediatamente, isto é, esperar até 2 unidades de tempo (Tal valor é justificado pelo tempo de espera caso a pista esteja interditada).

1. 2/2 SOBRE O EXERCÍCIO

Além disso, os aviões devem ser atendidos por ordem de chegada, ou seja, quem chega primeiro tem preferência. Entretanto, caso o combustível do avião que acabou de chegar chegue a zero ou ele tenha que ficar esperando por mais de 10% do seu tempo de voo se for colocado em último da fila, ele ganha o direito de ultrapassar os aviões que chegaram antes se tal rearranjo for possível.

Além disso, é possível, caso seja necessário, mandar aviões para outros aeroportos, quando não é possível gerenciar todos os pedidos que estão sendo recebidos para evitar que aviões caiam, ou atrasar as decolagens. Ambas alternativas devem ser evitadas ao máximo.

Dadas tais regras, a seguir é detalhado a implementação do programa e a estratégia escolhida para gerenciar o aeroporto de forma a minimizar o envio de aviões para outros aeroportos, os atrasos e as quedas, além de respeitar as prioridades estabelecidas.

2. 1/5 ESTRUTURAS UTILIZADAS

Para a implementação, foram criadas as seguintes estruturas:

>>> PISTA

```
class Pista {
  public:
    int id;
    int quantidade;
    int tempo_interditada;
    Fila * fila;
};
```

A class <Pista> serve para guardar as informações sobre as pistas, sendo estas informações:

- **id** - Um número inteiro de 0 a 3 que identifica a pista;
- **quantidade** - Um número inteiro que indica a quantidade de aviões na fila da pista;
- **tempo_interditada** - Um número inteiro que indica por quanto tempo, após um pouso ou decolagem, a pista ficará interditada (de 0 a 2);
- **fila** - Um vetor do tipo Fila (Descrito a seguir) que guarda os aviões que estão esperando para pousar/decolar na pista;

2_ ESTRUTURAS UTILIZADAS

>>> AVIOES

```
class Avioes {
  public:
    char id[5];
    char id_voo[3];
    int time_comb;
    int time_voo;
    int type;
    int inst;
};
```

A class <Avioes> serve para guardar as informações sobre os aviões, sendo estas informações:

- **id[5]**- Uma string de 5 caracteres no formato AB123 que identifica o avião com um código único.
- **id_voo[3]** - Uma string de 3 caracteres no formato ABC que identifica o voo;
- **time_comb** - Um número inteiro entre 0 e C que guarda quantas unidades de tempo o combustível aguenta;
- **time_voo** - Um número inteiro entre 0 e V que informa quantas unidades de tempo o voo ira durar;
  - Caso seja um pouso, time_voo é salvo como 0, caso seja uma decolagem, o time_comb é dado como 0;
- **type** - Um inteiro 0 ou 1 que indica o tipo do voo. Se 0 é normal, se 1 é especial;
- **inst** - Um inteiro que informa o instante em que o avião entrou em contato com o aeroporto;

2_ ESTRUTURAS UTILIZADAS

3/5

>>> FILA

```
class Fila {
  public:
    Avioes aviao;
    Fila * prox;
}
```

A class <Fila> serve para guardar os <Avioes> e cada pista possui uma. Seus atributos são:

- **aviao** - Um objeto do tipo <Avioes>. É a chave da nossa fila de prioridades.
- **prox** - Um ponteiro do tipo Fila * que aponta para a próxima posição da fila.

2_ 4/5 ESTRUTURAS UTILIZADAS

>>> FILAP

```
class FilaP {
  public:
    Pista pista;
    FilaP * prox;
};
```

A class <FilaP> serve para guardar as pistas em uma fila. É semelhante a <Fila> de Aviões, entretanto, guarda pistas ao invés de <Avioes>. Seus atributos são:

- **pista** - Um objeto do tipo <Pista>, já descrito anteriormente;
- **prox** - Um ponteiro do tipo <FilaP> que aponta para a próxima posição da fila;

2. ESTRUTURAS UTILIZADAS

5/5

>>> HISTORICO

```
class Historico {
  public:
    Avioes aviao;
    Historico * dir;
    Historico * esq;
};
```

A class <Historico> serve somente para guardar uma lista dos aviões que entraram em contato e verificar se não há códigos repetidos na simulação. A estrutura utilizada para tal feito é uma Árvore Binária de Busca. Assim temos como atributos:

- **aviao** - Um objeto do tipo <Avioes>. É o conteúdo de cada célula;
- **dir** - Um ponteiro do tipo * Historico. Dada uma célula, o ponteiro a direita aponta para uma subárvore onde todos os aviões são "maiores" considerando uma ordenação por ordem alfabética;
- **esq** - Um ponteiro do tipo * Historico. Assim como o 'dir', esse também aponta para uma subárvore mas com aviões "menores" considerando a ordem alfabética;

Destaco novamente que essa estrutura só fora implementada no intuito de evitar a repetição de códigos de aviões e não tem papel fundamental na estratégia de gerenciamento das filas.

3. 1/4 ENTRADAS & SAÍDAS

Como parâmetros de **entrada** temos:

- **T -** Unidades de tempo da simulação;
- **K -** Número máximo de aviões que chegam no aeroporto por unidade de tempo;
- **PP -** Probabilidade de ser um pouso;
- **PD -** Probabilidade de ser uma decolagem (1 - pp);
- **PE -** Probabilidade de ser de emergência;
- **C -** Tempo máximo de combustível de um avião que deseja pousar;
- **V -** Tempo máximo de voo de uma decolagem;

A cada iteração, são gerados de 0 a K aviões. Tal valor é determinado utilizando a função rand() e a srand() do C++. Os valores T e K devem ser inteiros maiores que 0 para que a simulação ocorra normalmente.

Já as probabilidades (pp, pd, pe) são todos valores de 0 a 100 que indicam em porcentagem (%) a chance de um avião ser de um determinado tipo. Assim, exemplificando, caso a entrada seja pp = 80, entende-se que o usuário deseja realizar uma simulação na qual há 80% de chance de um avião ser de pouso. Nota-se que a probabilidade pd não é uma entrada necessária dado que pd = 100 - pd.

Já os valores C e V indicam, respectivamente, os valores máximos de combustível e tempo de voo para cada avião.

3_ 2/4 ENTRADAS & SAÍDAS

Como **saída** temos:
Para cada iteração - As saídas estão separadas em diferentes segmentos, sendo esses:

- **Instante Atual:**
  - Informando o instante atual e o número de aviões que entraram em contato com o aeroporto naquele instante;
  - Caso algum desses aviões não seja inserido em alguma das filas das pistas, é printado o código do avião que não foi inserido e um alerta de que esse mesmo avião foi inserido na fila de emergências;

- **Status das Filas:**
  - Para cada uma das filas das 3 pistas é impresso:
    - A quantidade de aviões presente na fila;
    - O tempo de interdição da pista;
    - Informações sobre os aviões, sendo essas:
      - O código de 5 caracteres do avião;
      - O combustível atual (Comb Atual) considerando o valor de entrada e tempo já passado caso seja pouso ou o tempo que ele ainda aguenta esperar dado que cada avião pode esperar 10% do seu tempo de voo (Voo Atual) caso seja uma decolagem;
      - O instante em que entrou em contato com o aeroporto;
      - E, por fim, o tipo - Se é normal ou especial;

3 3/4 ENTRADAS & SAÍDAS

- **Pousos e Decolagens:**
  - Indica que aviões estão pousando ou decolando em cada uma das pistas;
    - Caso nenhum avião esteja pousando ou decolando em nenhuma das pistas, nada é impresso;
    - Informa dados do avião que está pousando ou decolando, como o código, tempo de espera total para decolagem ou o tempo de espera total para pouso e combustível restante no pouso
  - Indica a quantidade de aviões que já decolaram e pousaram;

- **Médias:**
  - MCPE - Média de Combustível dos Aviões Esperando;
  - MCRP - Média de Combustível Restante dos Aviões que já Pousaram;
  - MED - Média do tempo de Espera para Decolagem;
  - MTPE - Média do Tempo de Espera para Pouso;

- **Emergências:**
  - Lista com todos os aviões que estão na lista de emergência dispondo as mesmas informações sobre cada um dos aviões que são expostas na parte dos Status das Filas;
  - Caso o avião esteja em situação de crítica (detalhes a frente) - É impresso se o avião foi inserido em alguma das filas, se ele foi mandado para outro aeroporto, se ele caiu ou se não foi possível retirá-lo da fila naquela iteração;

3_ 4/4 ENTRADAS & SAÍDAS

Ao final da simulação:

- Número total de aviões que entraram em contato com o aeroporto;
- Os valores finais de todas as médias;
- O número de quedas;
- O número de decolagens atrasadas;
- O número de pousos;
- O número de decolagens;
- O número de desvios para outros aeroportos;

4. FUNÇÕES

Para a implementação, foram criadas as seguintes funções:

```
Pista::Pista(int id_pista, int quant, int time, Fila * f);
```

Essa função cria um objeto do tipo <Pista> com os parâmetros dados.

```
Avioes Pista::remove_na_pos(int pos);
```

Essa função remove na fila da Pista o avião que está ocupando a posição de número pos.

```
void Pista::pousa_decola_primeiro_aviao(int * mcrp, int *
mtep, int * med, int * quant_dec, int * quant_pou, int t);
```

Essa função remove da fila da Pista o primeiro avião, printa suas informações de pouso ou decolagem e atualiza as médias e quantidades relacionadas ao pouso e ou a decolagem. Recebe como parâmetros ponteiros para todas as médias e quantidades calculadas e o instante atual t para o calculo das médias.

4. FUNÇÕES


```
Avioes::Avioes(char * id_aviao, char * info_voo, int comb,
int voo, int tipo, int t);
```

Essa função cria um objeto do tipo <Avioes> com os parâmetros dados.

```
int Avioes::simula_na_pos(int pos, int t, int
time_interditada)
```

Essa função pega um avião e simula colocar ele numa posição de número 'pos' considerando o instante atual e o tempo de interdição da pista e os atributos de tempo de voo ou quantidade de combustível ou o tipo do avião em que está sendo executado o teste. A função devolve:

- 0 - Se não for possível colocar na posição 'pos';
- 1 - Caso seja possível;

```
Fila * insere2(Fila * f, Avioes A)
```

Essa função insere o avião A ao final da fila f e devolve um ponteiro para o início da nova fila atualizada. É utilizada somente para a fila de emergências.

```
Fila * remove2(Fila * f, Avioes A)
```

Essa função busca o avião A na fila f e remove-o da mesma, retornando um ponteiro para o início da nova fila atualizada. É utilizada somente para a fila de emergências.

4. FUNÇÕES


```
void calcula_media_mcpe(FilaP * p, int t, int * mcpe);
```

Essa função recebe uma <FilaP> p com todas as pistas e o instante atual t e calcula a média atual de combustível restante (MCPE) dos aviões que irão pousar e que estão nas filas.

```
int conta_posicoes(Fila * fila);
```

Essa função recebe uma <Fila> fila e devolve quantos elementos estão na fila. Tal valor é utilizado para determinar a posição dos aviões na fila em outras funções.

```
Avioes gera_aviao(int C, int V, int pp, int pe, int t);
```

Dados os parâmetros de entrada, essa função gera um avião com dados aleatórios usando as funções rand() e srand(). No caso das probabilidades, é gerado um valor de 0 a 100 'aleatoriamente' e caso o valor seja menor que pp, o avião será de pouso, caso contrário de decolagem. Ideia análoga para decidir se será um avião normal ou especial com pe.

```
Fila * insere_em_pos(Fila * fila, int pos, Avioes A, int
quantidade)
```

Insere o avião A na posição 'pos' da <Fila> fila e retorna o ponteiro para o início da fila atualizada. O valor 'quantidade' indica a quantidade de aviões na fila e é utilizado para verificar onde inserir o avião A.

4. FUNÇÕES

```
FilaP * insere(FilaP * fila, Pista px)
```

Insere a <Pista> px ao final da <FilaP> fila.

```
FilaP * insere_pista(FilaP * fila, Avioes A, Pista p1,
Pista p2, Pista p3, int pp)
```

Essa função insere as pistas p1, p2 e p3 na <FilaP> fila de pistas ordenando-as de acordo com atributos do avião e da porcentagem de aviões de pouso.
Caso o avião seja normal e de pouso, apenas as pistas p1 e p2 são inseridas e ordenadas por menor quantidade. Caso seja pouso e especial, o mesmo ocorre, porém a p3 é adicionada ao final.
Caso o avião seja de decolagem e a probabilidade PP >= 60%, a p3 é colocada em primeiro lugar e as outras duas ordenadas por quantidade. Caso contrário, todas são inseridas em ordenação de quantidade de aviões.

```
Historico * insere_em_hist(Historico * hist, Avioes A, int
* v)
```

Essa função serve puramente para verificar se o código do avião A é repetido ou não. Caso o código do avião A já esteja no <Historico> hist, o valor de v é 0 e A não é inserido em hist, caso contrário v é 1 e o avião A é inserido em hist. A estrutura utilizada é uma Árvore Binária de Busca onde os menores elementos ao nó ficam à esquerda e os maiores à direita.

4. FUNÇÕES

5/5

`int simula_empurrar(Pista pista, int pos, int t, int time_interditada)`

Essa função recebe uma <Pista> pista, uma posição, o instante atual e o tempo de interdição da pista. E então, verifica se o avião que está na posição 'pos' da fila da pista pode ser empurrado para a posição 'pos + 1', isto é, se algum avião pode passar na frente dele ou não. Se der, a função devolve 1, se não der a função devolve 0 ou -1. O caso -1 é quando não há avião na 'pos' da fila, ou seja, tem menos aviões do que 'pos' na fila.

`int simula_colocar_na_pista(Pista pista, Avioes aviao, int t)`

Essa função recebe uma <Pista> pista, o instante atual e um avião e simula colocar o avião na pista, testando inserir o mesmo sempre na última posição da fila, e vai diminuindo se fosse necessário, e utilizando as funções de 'simula_na_pos' e 'simula_empurrar' para verificar se é possível inserir o avião na fila da pista. E assim devolve:

- POS > 0 - posição em que é possível inserir o avião na fila da pista;
- 0 - Caso não seja possível inserir o avião na fila.

5. 1/6 ESTRATÉGIA

Vamos a descrição da estratégia utilizada. Mas antes, algumas definições:

- Aviões com combustível menor ou igual a 3 são considerados emergências - Pois, considerando que as pistas podem ficar interditadas por até 2 unidades de tempo, ainda é plausível colocar o avião na primeira posição para esperar essas 2 unidades de tempo sem que o mesmo caia.
- O 'pouso' ou 'decolagem' imediata é considerado como imediato se ocorrer em até 2 unidades de tempo, pela mesma razão acima. Pois, caso tenhamos um avião especial que deseja pousar imediatamente, por exemplo, mas terá que aguardar 2 unidades de tempo pela interdição da pista, é melhor que ele aguarde do que ir para outro aeroporto.
- Não são gerados aviões com 0 de combustível e 0 de tempo de voo ao mesmo tempo. Se um dos valores for 0, o outro automaticamente é diferente de zero.
    - Tal dinâmica é justificada pela implementação da estratégia pois a diferenciação de aviões de pouso e decolagem é feito pelo valor do atributo de 'time_comb' e 'time_voo'.

5. 2/6 ESTRATÉGIA

A estratégia funciona da seguinte forma:
Existem 4 filas no total, 1 fila para cada pista e 1 fila de emergências - para os aviões que não puderam ser inseridos de primeira em umas das filas das pistas. A cada novo avião que entra em contato, são organizados testes de inserção do mesmo em cada uma das pistas. A organização desses testes é baseada no seguinte:

- Se o avião for de decolagem:
  - Se a probabilidade de ser de decolagem for < 40%, coloca-se a P3 em primeiro lugar e ordena por quantidade de aviões as outras duas para executar os testes;
    - A decisão de testar primeiro com a P3 é justificada pelo fato da mesma só receber pousos em casos de exceção como aviões especiais ou em casos de emergência.
  - Caso contrário, ordena-se as pistas por quantidade de aviões;
    - Tal decisão foi tomada para evitar, no caso de uma probabilidade mais alta dos aviões serem de decolagem, para exemplificar, 90% de PD e 10% de PP, os aviões serem inseridos em sua maioria na pista 3 enquanto as outras ficam completamente vazias.
    - Note que não importa o avião ser normal ou especial nesse caso;
- Se o avião for de pouso:
  - Se for normal - Ordena as pistas P1 e P2 por quantidade de aviões;
  - Se for especial - Ordena também as pistas P1 e P2 e deixa a P3 por último;

5. 3/6 ESTRATÉGIA

Após a ordenação das pistas para executar os testes de inserção na pista, começa-se os testes.

O avião é testado em cada pista da seguinte forma:

- Simula colocar ele na última posição da fila e vai diminuindo essa posição até encontrar uma posição aceitável para ele.
  - Note que aviões especiais aceitam somente a posição 1 como aceitável;
- Ao achar essa posição aceitável, ocorre uma outra simulação para verificar se os aviões que já estão na fila em uma posição posterior a posição aceitável desse novo avião, se houverem aviões após essa posição, podem ser empurrados para trás.
  - Caso possam, insere-se o avião na fila e vamos para o próximo avião a ser testado;
  - Caso não dê certo, testamos com a próxima pista;

Tal estratégia mantém um respeito pela questão da prioridade de quem chegou antes, pois a tentativa de inserção do novo avião sempre começa pelo final da fila. Se não deu certo, significa que o avião iria cair ou atrasar sua decolagem caso ficasse naquela posição, logo, ele pode passar na frente de algum outro avião caso este aguente ser empurrado para trás na fila.
No caso de o avião não ser inserido em nenhuma das pistas em que os testes aconteceram, ele é inserido na fila de emergências.

5. ESTRATÉGIA

4/6

O que foi descrito até então é a primeira parte da estratégia. Note que os aviões que são inseridos na fila de alguma das pistas de primeira, isto é, sem passar pela fila de emergências, não correm, a princípio, o risco de atraso ou possível queda pois os possíveis decolagens atrasadas e quedas já foram previstos na simulação e o avião foi inserido na melhor posição possível, respeitando a prioridade de quem veio antes e a questão de evitar atrasos e quedas. O único caso de possível atraso é caso em que o avião seja removido da fila, entrando na fila de emergências para dar lugar a um avião que é emergencial no instante atual.

Antes da segunda parte da estratégia, ocorrem os pousos e decolagens caso as pistas não estejam interditadas e consequente remoção destes aviões da fila.

A segunda parte da estratégia refere-se à fila de emergências. Caso o avião não consiga ser inserido em nenhuma das filas das pistas, ele é inserido na fila de emergências, que não conta com nenhuma ordenação pois, a cada instante novo, todos os aviões que estão na fila de emergências passam por novos testes para verificar se podem ou precisam serem inseridos nas filas das pistas no instante da iteração.

5. 5/6 ESTRATÉGIA

Caso a fila de emergências esteja vazia, o instante atual acabou e seguimos para o próximo instante. Caso esteja com algum avião, fazemos o seguinte:

Se o combustível atual do avião que está sendo verificado, o tempo de voo que marca quanto tempo ele pode esperar para decolar sem se atrasar for menor ou igual a 3 ou o avião for do tipo especial iremos tentar reinserir o mesmo nas filas das pistas ou mandá-lo para pousar em outro aeroporto, caso seja um pouso. Caso contrário, vamos para o próximo avião da fila de emergências, caso não tenha mais nenhum, iremos para o próximo instante.

Para verificar se o avião pode ser reinserido na primeira posição, é feito o seguinte: Primeiro, para cada pista, verificamos se é possível empurrar o primeiro elemento da sua fila. Isso só é possível se o avião que está em primeiro for normal e não for cair ou se atrasar sendo empurrado, pois, se for especial, se for cair ou se atrasar, ele é uma emergência e não faz sentido retirar uma emergência da fila para substituir por outra no mesmo instante. Se der para empurrar o primeiro, colocamos a pista numa fila de pistas para realizar outros testes.

5. ESTRATÉGIA 6/6

Note que caso um avião seja retirado da fila das pistas, isso indica que o mesmo ainda não é uma emergência, porém, se ele foi retirado, isso indica que ele iria aguardar o tempo máximo possível para pousar ou decolar. Ou seja, a chance de estarmos ‘atrasando’ esse avião ainda mais é baixa, já que quando ele se tornar uma emergência, iremos tentar inseri-lo em todas as outras pistas.

Note também que, caso tenhamos um avião que está na iminência de pousar ou decolar, mas aguenta ser empurrado, ele será empurrado para dar lugar a uma emergência. Como existem vários aviões assim na fila das pistas, já que ela não é ordenada por 'time_comb' ou 'time_voo', a chance de colocar um avião de emergência na fila de alguma pista é alta.

Além disso, após passar por todos os aviões da fila de emergência, retirando aviões e/ou adicionando novos, note que a fila de emergência tem seu tamanho diminuído ou mantém-se igual. O número de elementos na fila de emergência só pode crescer no começo da iteração quando chegam novos aviões.

Isso conclui a estratégia que é baseada em respeitar a prioridade de quem chegou primeiro, exceto quando envolve a presença de emergências e rearranjar as filas para evitar consequentes desvios, quedas e atrasos.

6. 1/2 TESTES & RESULTADOS

Para a execução dos testes basta dar os parâmetros de entrada da seguinte forma:

- >> T - Número inteiro maior que 0;
- >> K - Número inteiro maior que 0;
- >> pp - Número inteiro entre 0 e 100 indicando a probabilidade em % de ser um pouso;
- >> pd - Não é dado com entrada, mas é determinado como 100 - pp;
- >> pe - Número inteiro entre 0 e 100 indicando a probabilidade em % de ser especial;
- >> C - Número inteiro maior que 0 e de preferência maior que 2, considerando que a interdição da pista é de 2 unidades de tempo;
- >> V - Número inteiro maior que 0 e de preferência maior que 20, considerando que a interdição da pista é de 2 unidades de tempo e o ‘time_voo’ é 10% do tempo de voo;

Para executar os testes instante a instante, basta ir pressionando ‘enter’ para ir de um instante para outro, enquanto verifica as informações. Para realizar todas as iterações de uma vez só, basta comentar a linha 773 (scanf("%c", &s);).

Para verificar a estratégia de uma forma simples, sugere-se dar a entrada com valores baixos de C e V ou com um valor alto de PE.

6. 2/2 TESTES & RESULTADOS

Após a execução dos testes que realizei, a conclusão foi de que o número máximo de pousos e decolagens sempre ficam em torno de T. O número máximo de aviões nas filas, após vários instantes, aproxima-se de C/3 ou V/3 (dependendo da probabilidade PP dada como entrada). Quando chega-se próximo a esses valores, mais e mais aviões são colocados na fila de emergências e os atrasos começam a ocorrer com mais frequência, pois ocorrem mais casos de 4 emergências ao mesmo tempo.

Ao realizar os testes com uma probabilidade alta de aviões especiais é possível verificar melhor o funcionamento da estratégia. Nesse tipo de teste, mais aviões são mandados para outros aeroportos e mais aviões sofrem atrasos na decolagem pois em quase toda iteração ocorrem 4 emergências.

Para testes com uma alta porcentagem de aviões de pouso, antes de implementada a estratégia de ordenar as pistas por quantidade de aviões caso PP <= 60%, todos os aviões se acumulavam na P3 e deixavam a P1 e P2 livres nos instantes iniciais, o que era um problema, já que muitos aviões que chegavam bem antes estavam sendo atendidos com uma espera bem maior daqueles que chegavam e entravam nas pistas vazias. Após a mudança, os aviões começaram a se distribuir de maneira mais uniforme nos primeiros instantes.

CONCLUSÃO

Por fim, os testes ocorreram como esperado, o rearranjo das pistas com a estratégia da fila de emergências funcionou bem até onde observei, entretanto, sendo honesto, ocorreram alguns casos em quantidade bem baixa de quedas de aviões, que a princípio não deveriam ocorrer, considerando a implementação feita, já que caso os aviões não entrassem na fila de emergências e não conseguissem serem inseridos nas filas das pistas em um momento crítico, eles iriam ser mandados diretamente a outros aeroportos ainda com combustível.

Porém, não tive tempo de explorar a fundo os motivos dessas quedas ocorreram, mas elas estavam provavelmente relacionadas ao tempo de interdição da pista não estar sendo considerado corretamente na inserção dos aviões saindo da fila de emergências em algum caso que não eu não fui capaz de prever.

Fora isso, outro problema a relatar, foi um bug que não consegui resolver relacionado a produção do código de 3 letras que identifica o voo. Ao gerar esse código, que fica salvo em um parâmetro ‘id_voo’ em cada avião, tudo ocorreu bem e corretamente, mas ao imprimir tal código com o “cout” e também o “printf”, erros começaram a surgir. O ‘id’ do avião, código de 5 caracteres gerado anteriormente começou a apresentar erros e o código ‘id_voo’ estava sendo concatenado com o ‘id’ do avião. Por tal fato, decidi não printar o código em nenhum momento.

Enfim, é isso, obrigado pela correção! :)

FIM