Lista de Exercícios II – ED I
Data de entrega: 27/09/2022

As listas de exercícios são para serem resolvidas em duplas. Cuidado com as cópias de trabalhos, pois cópias surtirão na divisão de uma nota pela quantidade de trabalhos iguais. As respostas aos problemas aqui tratados envolvem programação. Sendo assim, cada exercício é tratado como um programa diferente. Somente os arquivos-fonte devem ser enviados pelo SIGAA. Para o envio é necessário se compactar TODOS os arquivos fontes construídos e então enviar este arquivo compactado pelo SIGAA.

Exercício 1: implemente o problema da Torre de Hanói. A Torre de Hanói é composta de 3 hastes A, B e C, onde em uma haste encontram-se discos dispostos como uma pirâmide, todos de tamanhos diferentes, onde um disco menor está sempre sobre um disco maior. O problema é passar todos os discos da torre A para a torre C, utilizando a torre B como auxiliar, um disco por vez, onde nenhum disco maior pode ficar em cima de um disco menor. A seguir encontra-se a Figura 1 que ilustra o problema:

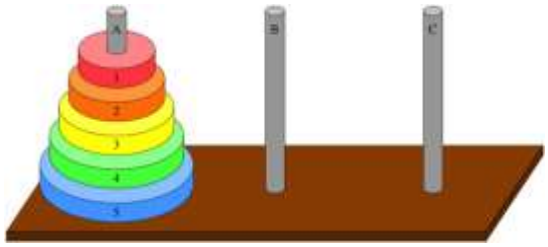

Figura 1: Ilustração da Torre de Hanói

Exercício 2: implemente um algoritmo que implemente um deque e controle a inserção e remoção de valores numéricos inteiros em um vetor de 10 valores. Um menu deve ser construído com as opções de inserção e remoção que o usuário pode escolher para uma das 2 extremidades do vetor. Após a execução de uma opção do menu o programa deve retornar para o menu principal para ser escolhida a próxima função a ser executada. Segue as funcionalidades do programa:

- Inserção no topo: esta inserção, muito parecida com a pilha, deve ser realizada na frente dos valores já existentes no vetor, caso ainda se tenha espaço;
- Inserção no final: assim como a fila, a inserção deve ser realizada na última posição válida do vetor, caso ainda se tenha espaço;
- Remoção no topo: esta opção remove o valor que está na cabeça do vetor, caso ainda se tenha valores a serem removidos;
- Remoção no final: esta opção remove o valor que está no final do vetor, caso ainda se tenham valores a serem removidos;

- Verificar valor inicial: o programa retorna ao usuário o valor que se encontra no início do vetor;
- Verificar valor final: o programa retorna ao usuário o valor que se encontra no final do vetor;
- Fim: o programa encerra sua execução.

Exercício 3: para um dado número inteiro $n > 1$, o menor inteiro $d > 1$ que divide n é chamado de fator primo. É possível determinar a *fatoração prima* de n achando-se o fator primo d e substituindo n pelo quociente n/d, repetindo essa operação até que n seja igual a 1. Utilizando uma Pilha ou Fila, implemente um programa que compute a fatoração prima de um número imprimindo os seus fatores em ordem crescente. Por exemplo, para $n = 3960$, deverá ser impresso $(11 * 5 * 3 * 3 * 2 * 2 * 2)$.

Exercício 4: utilizando a estratégia de algoritmos gulosos ou de força bruta implemente os problemas a seguir na busca de uma solução possível, não necessariamente a melhor. Não necessariamente precisa-se utilizar a mesma estratégia de algoritmos (guloso/força bruta) em todos os problemas:

- Problema do Caixeiro Viajante
    - Este problema se baseia na realização de um percurso saindo de uma cidade, passando por todas as outras, e retornando na cidade inicial, com o menor caminho a ser percorrido;
    - Implemente para 100 cidade;
    - Cada cidade terá sua coordenada (x, y) que deve ser inserida de forma automática e aleatória com limite de 500 (quinhentos) por coordenada, evitando, assim, cidades com as MESMAS coordenadas (x, y);
    - Deve-se tratar o problema como um grafo completo onde todas as cidades estão conectadas com todas as cidades;
    - A distância entre 2 cidades diferentes deve ser calculada, ou seja, euclidiana, $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.
- Problema da Mochila
    - O usuário possui uma mochila que comporta uma certa quantidade de peso. Existem diversos itens para serem escolhidos a serem incluídos na mochila onde cada item possui seu peso e seu valor. O objetivo é se obter a mochila com o maior valor possível de itens;
    - O peso da mochila o usuário pode estipular, assim como a quantidade de itens;
    - O peso e o valor de cada item devem ser construídos de forma automática e aleatória com um limite de 30 para cada característica do tem (peso ou valor), as quais não necessariamente devem ser iguais.