Simulado da 2ª Avaliação

Algoritmos e Estrutura de Dados I (AE22CP)

Prof. Jefferson T. Oliva

1. Implemente uma função recursiva que receba, pelo menos (ou seja, se você preferir, pode colocar mais parâmetros na função, desde que explique para quê servem), um caractere e uma string. A função deverá retornar a quantidade de vezes que o caractere aparece na string. Por exemplo, para o char 'a' e a string "abaacdba", a função deverá retornar 4.

2. Dado o seguinte arranjo de caracteres que foi submetido a um algoritmo ordenação: {O, R, D, E, N, A, D, O}. Em algum momento, o arranjo encontra-se na seguinte forma: {D, E, O, R, A, D, N, O}.

   Responda:

   a) - Cite **um** dos algoritmos de ordenação que não foi aplicado. Justifique a sua resposta com a aplicação de um teste de mesa demonstrando o funcionamento do algoritmo escolhido.

   b) - Qual algoritmo de ordenação foi aplicado? Justifique a sua resposta.

   c) - Em um teste de mesa, a partir do arranjo em seu estado inicial (ordem decrescente), aplique o algoritmo de ordenação, passo-a-passo, até chegar ao estado atual {D, E, O, R, A, D, N, O}

3. Dado o algoritmo de inserção no final de listas encadeadas abaixo:

```
Cell* inserir_ultimo(int x, Cell *cel){
     if (cel == NULL)
         return cel = criar_celula(x);
     else if (cel->prox == NULL)
         return cel->prox = criar_celula(x);
     else
         return inserir_ultimo(x, cel->prox);
   }


void inserir_lista(int x, Lista *l){
     if (l == NULL)
         l = criar_lista();
     l->head = inserir_ultimo(x, l->head);
}
```

   Implemente uma versão alternativa à função "*inserir_ultimo*" de modo que, em vez de utilizar recursão, utilize iteração explícita para a inserção de um novo elemento ao final da lista encadeada.

 Continua na próxima página...

4. Implemente uma função que receba três filas estáticas (f1, f2, e f3), sendo uma com elementos (f1) e as outras duas, vazias. A função deverá remover os elementos de f1 e enfileirá-las nas outras duas filas, onde cada uma deve conter a metade dos elementos de f1. Observe que na estrutura da Fila, apresentada no anexo, não há campo para representar o tamanho. Por fim, os elementos da fila podem ser acessados apenas pela função desenfileirar. Exemplo:

   - Antes:
     - f1 = {1, 2, 3, 4, 5}
     - f2 = { }
     - f3 = { }
   - Depois:
     - f1 = { }
     - f2 = {1, 2}
     - f3 = {3, 4, 5}

5. Implemente uma função que receba duas listas encadeadas. A função deverá retornar uma lista encadeada resultante da intercalação das duas listas. Para isso, podem ser utilizadas as estruturas e as funções definidas no anexo. Caso necessário, implemente funções auxiliares para resolver este exercício. Em seguida, faça a análise de complexidade da sua função. Exemplo de antes e depois da aplicação da função:

   - Antes:
     - l1 = {1, 2, 3}
     - l2 = {4, 5, 6}
     - l3 = { }
   - Depois:
     - l1 = { }
     - l2 = { }
     - l3 = {1, 4, 2, 5, 3, 6}

 Continua na próxima página...

6. Considere:

   - Os seguintes algoritmos de pesquisa: busca sequencial; busca sequencial indexada com tabela de índices de tamanho 5; e busca binária.
   - O seguinte arranjo ordenado com 25 elementos: {1, 4, 9, 11, 14, 19, 22, 23, 27, 34, 38, 42, 55, 56, 60, 62, 67, 74, 78, 84, 88, 89, 90, 91, 95}

   Responda:

   a) - Qual dos algoritmos listados acima é o mais rápido para procurar a chave 14? Quantas comparações são necessárias para esse algoritmo encontrar a chave 14? Mostre como você contou a quantidade de comparações. (10 pontos)

   b) - Qual dos algoritmos listados acima é o mais lento para procurar a chave 14? Quantas comparações são necessárias para esse algoritmo encontrar a chave 14? Mostre como você contou a quantidade de comparações. (10 pontos)

Anexo

Estruturas de dados e protótipos de função para resolução do Exercício 5:

```
#define TMAX 100
typedef struct Cell{
    int item;
    struct Cell *prox;
}Cell;
   typedef struct{
    Cell *cabeca;
}Lista;
   typedef struct{
    Cell *cabeca;
}Pilha;
   typedef struct{
    int item[TMAX];
    int ini, fim;
}Fila;
```

```
Cell* criar_celula(int chave);
Lista* criar_lista();
Pilha* criar_pilha();
Fila* criar_fila();
int pilha_vazia(Pilha *p);
int fila_vazia(Fila *f);
void empilhar(Pilha *p, int chave);
int desempilhar(Pilha *p);
void enfileirar(Fila *f, int chave);
int desenfileirar(Fila *f);
void liberar_lista(Lista *l);
void liberar_pilha(Pilha *p);
void liberar_fila(Fila *f);
```

 Final do qeustionário.