Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Estrutura de Dados
Gabarito AP1 - 2019/1

1) Leia atentamente o texto a seguir, e depois resolva as questões abaixo.

Sabemos que um problema computacional pode admitir vários algoritmos para resolvê-lo, que podem ter complexidades de pior caso diferentes. Dentre esses algoritmos, aqueles que possuem as menores complexidades de pior caso podem ou não ser ótimos, pois o que caracteriza um algoritmo ótimo não é o fato de ele ser ``melhor'' do que os outros, mas é algo relacionado à complexidade intrínseca do problema que ele resolve, isto é, ao limite inferior do problema.

(a) (1,0) Dê a definição formal de algoritmo ótimo.

R: Seja P um problema. Um limite inferior para P é uma função L, tal que a complexidade de pior caso de qualquer algoritmo que resolva P seja $\Omega(L)$. Neste sentido, um algoritmo A, com complexidade O(L) que resolva P é então considerado um algoritmo ótimo para o problema P.

(b) (1,0) Responda: Um problema computacional pode admitir mais de um algoritmo ótimo para resolvê-lo? Justifique.

R: Sim, é possível admitir que haja mais de um algoritmo ótimo capaz de resolver o mesmo problema. Isto acontece pois todo algoritmo que apresentar complexidade de pior caso igual ao limite inferior de resolução deste problema, será considerado um algoritmo ótimo.

2) Considere um volumoso cadastro de CPF's, no qual as seguintes operações são executadas:

1. **Busca de um CPF no cadastro.**
2. **Inserção de um novo CPF no cadastro.**
3. **Remoção de um CPF do cadastro.**

Suponha ainda que, para cada CPF, existe um contador que é incrementado de uma unidade a cada vez que este é buscado no cadastro. Isto forma uma estatística dos acessos. Temos a seguir várias estruturas de dados que poderiam ser utilizadas para implementar o cadastro de CPF's:
(a) Lista linear não ordenada;
(b) Lista linear ordenada por CPF;
(c) Lista linear ordenada (decrescentemente) por contador;
(d) Lista simplesmente encadeada não ordenada;
(e) Lista simplesmente encadeada ordenada pelo CPF;
(f) Lista simplesmente encadeada ordenada (decrescentemente) pelo contador.

Resolva as questões a seguir, justificando:
(a) (1,5) Quais as melhores estruturas em relação à operação 1?
R: Dentre as estruturas mencionadas, a melhor para implementação de uma operação de busca, neste cenário, é a lista linear ordenada pelo CPF. O fato desta lista estar ordenada pelo mesmo elemento que se faz a busca (CPF) torna a operação mais simples. Neste caso podemos, por exemplo, utilizar a busca binária, que parte do princípio que a lista está ordenada e possui complexidade de pior caso igual a O(log N). Por outro lado, as demais estruturas apresentam complexidade de pior caso, para busca, igual a O(N).

(b) (1,5) Quais as melhores estruturas em relação à operação 2?
R: Neste caso, as seguintes estruturas podem ser utilizadas:

> **- Lista linear não ordenada:** utilizando esta estrutura, o custo da operação pode ser constante - O(1). Basta inserirmos o novo elemento no final do vetor.
> **- Lista simplesmente encadeada não ordenada:** esta estrutura também permite que o custo da inserção seja constante em um cenário onde tenhamos um ponteiro **para o último elemento da lista**, portanto, não precisando percorrer toda.

Seguindo este mesmo princípio, as **listas ordenadas pelo contador (linear e simplesmente encadeada)** também nos permite inserir os novos elementos sempre no fim da lista, uma vez que o contador do novo elemento será 1 (o menor de toda a lista) no momento da inserção. Para inserirmos na lista simplesmente encadeada, basta utilizarmos o ponteiro para o último elemento da lista. As demais estruturas podem precisar de uma reordenação após a inserção, com custo O(n log n). "

(c) (1,5) Quais as melhores estruturas em relação à operação 3?
R: Se considerarmos apenas a operação de remoção, as estruturas (a), (d), (e) e (f) possuem complexidade de remoção O(1). No caso da lista linear não ordenada, basta substituirmos o elemento removido pelo valor da última posição do vetor e decrementarmos de 1 o tamanho do vetor. Para as estruturas encadeadas, é necessário apenas reajustarmos os ponteiros, fazendo com que o campo *prox* do elemento anterior ao removido aponte para o seu próximo.

3) Dado um vetor V com n > 1 elementos distintos, a mediana de V é um elemento V[i] (para algum índice i entre 1 e n) com a seguinte propriedade: existem int(n/2) elementos em V que são menores do que V[i], onde int(x) é a parte inteira do número x. Como exemplo, considere o vetor V = (2 19 26 3 4 1 5). Temos n=7 e int(n/2) =3. Portanto, a mediana de V é o elemento V[5] = 4.

(a) (1,5) Escreva um algoritmo que monta um outro vetor M, contendo n elementos, com a seguinte propriedade: para cada índice i entre 1 e n, o elemento M[i] armazena quantos elementos de V são menores do que V[i]. Como exemplo, para o vetor V acima, temos que M = (1 5 6 2 3 0 4).

R: Seja M, um vetor com N posições, temos:

```
para i de 1 até N, faça:
    menores := 0
    para j de 1 até N, faça:
        se V[j] < V[i] então:
            menores = menores + 1
    M[i] = menores
```

(b) (1,0) Suponha que o vetor M já esteja calculado. Escreva um comando do tipo “enquanto” que obtém a mediana de V a partir de M. Se possível, faça o comando se encerrar tão logo a mediana seja encontrada.

R: Uma possível solução é apresentada abaixo:

```
menores := int(N/2)
i := 1

enquanto i < N faça:
    se M[i] = menores então:
        indice := i
        i := n
    senão
        i = i + 1

imprimir ("A mediana é ", V[indice])
```

4) (1,0) Considere uma fila F contendo as posições de 1 a 5. A variável f marca a posição de início da fila ("frente"), e a variável r marca a posição de fim da fila ("retaguarda"). No início, a fila F encontra-se vazia, e as variáveis f e r valem zero.

Usamos a notação R para denotar a operação de remoção de um elemento da fila F, e a notação I(X) para denotar a operação de inserção de um elemento X na fila F.

Considere a seguinte sequência de operações em F:

I(A), I(B), I(C), R, I(D), R, I(E), I(G), I(H), R, R, R, R, I(J)

Desenhe como fica a fila F após a sequência de operações acima, e forneça os valores finais das variáveis f e r. Use um traço (-) para denotar as posições vazias. Como um exemplo de configuração, poderíamos ter: F = (- - C D -), com f=3 e r=4.
R: Após a execução da sequência de operações apresentadas, acima a fila F ficará da seguinte forma: (- H J - -), sendo f=2 e r=3. O passo a passo das operações, que levam a fila a este estado, é apresentado a seguir:

Operação	**Fila**					***f***	***r***
	1	**2**	**3**	**4**	**5**		
I(A)	A	-	-	-	-	1	1
I(B)	A	B	-	-	-	1	2
I(C)	A	B	C	-	-	1	3
R	-	B	C	-	-	2	3
I(D)	-	B	C	D	-	2	4
R	-	-	C	D	-	3	4
I(E)	-	-	C	D	E	3	5
I(G)	G	-	C	D	E	3	1
I(H)	G	H	C	D	E	3	2
R	G	H	-	D	E	4	2
R	G	H	-	-	E	5	2
R	G	H	-	-	-	1	2
R	-	H	-	-	-	2	2
I(J)	-	H	J	-	-	2	3