Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
AP1 - Primeiro Semestre de 2015

Nome -
Assinatura -

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

1. (Valor 2,0) Explicar, com precisão, os seguintes conceitos:

   (1a) Complexidade de melhor caso.

   Resposta: Seja A um algoritmo e $E = \{E_1, E_2, ..., E_n\}$ o conjunto de todas as entradas possíveis de A. Dada uma entrada E_i, para $1 \leq i \leq n$, seja t_i o número de passos efetuados por A. Podemos definir: *complexidade de melhor caso* = $min_{E_i \in E}\{t_i\}$.

   (1b) Árvore binária completa.

   Resposta: Primeiro, vamos definir árvore binária e árvore cheia. Uma árvore é binária se cada nó possui no máximo 2 filhos. Uma árvore é cheia se todas as suas subárvores vazias se localizam no último nível. Uma árvore é binária completa se ela é binária e cheia até o penúltimo nível.

2. (Valor 2,0) Assinale V ou F, justificando:

   (2a) Se T é uma árvore binária cheia com n nós e k níveis, então $n = 2^k - 1$.

   Resposta: Verdadeiro. Seja T' uma árvore binária não-vazia com k níveis. O primeiro nível de T' tem apenas 1 nó. O segundo nível tem 2 nós. O terceiro nível tem 4 nós. Assim sucessivamente até o k-ésimo nível que tem no máximo 2^{k-1} nós.

   Uma árvore binária cheia T tem exatamente 2^{k-1} nós em cada nível. Portanto, o número de nós em T é a soma do número de nós em todos os níveis: $2^0 + 2^1 + 2^2 + ... + 2^{k-1} = 2^k - 1$.

   (2b) A complexidade de pior caso da busca binária aplicada a uma lista sequencial ordenada com n elementos é $\Theta(n)$.

   Resposta: Falso. A complexidade de pior caso da busca binária é $\Theta(log_2 n)$.

3. (Valor 3,0) Dada uma lista simplesmente encadeada L com n nós, descreva um algoritmo para inverter a direção do encadeamento de L. Isto é, o algoritmo deve transformar L em uma outra lista, contendo exatamente os mesmos nós do que L, porém na ordem invertida. Pede-se:

(3a) Descrever a estratégia geral do algoritmo, em palavras.

Resposta: O algoritmo terá 3 ponteiros apontando para elementos consecutivos de L. Sejam *pont1*, *pont2* e *pont3* esses ponteiros. O algoritmo começa com *pont1* apontando para o primeiro elemento de L e seu campo *prox* apontando para λ. Percorremos L fazendo com que o campo *prox* de cada nó aponte para o seu antecessor, para isso manipulamos os 3 ponteiros fazendo com que *pont1* aponte para *pont2*, *pont2* aponte para *pont3* e *pont3* aponte para *pont1*.

(3b) Descrever uma implementação do algoritmo, supondo que a lista L está armazenada com a utilização de ponteiros.

Resposta:

Algoritmo:

$pont1 := ptlista \uparrow .prox$
$pont2 := pont1 \uparrow .prox$
$pont1 \uparrow .prox := \lambda$
se $pont2 \neq \lambda$ então
    $pont3 := pont2 \uparrow .prox$
    enquanto $pont3 \neq \lambda$ faça
        $pont2 \uparrow .prox := pont1$
        $pont1 := pont2$
        $pont2 := pont3$
        $pont3 := pont2 \uparrow .prox$
    $pont2 \uparrow .prox := pont1$
    $ptlista \uparrow .prox := pont2$

(3c) Determinar e justificar a complexidade do algoritmo.

Resposta: O algoritmo percorre a lista L apenas uma vez executando um número constante de passos para cada elemento, portanto sua complexidade é $\Theta(n)$.

4. (Valor 3,0) Os processos que estão esperando para serem executados pelo processador ficam armazenados numa estrutura de dados que utiliza a seguinte política: o processo mais antigo (ou seja, aquele que está esperando há mais tempo) tem sempre prioridade de execução quando o processador estiver liberado.

(4a) Responda, justificando: qual estrutura de dados vista na disciplina é a mais adequada para armazenar os processos em espera?

Resposta: A estrutura de dados mais adequada seria uma fila, pois nesta todas as inserções ocorrem numa extremidade e as remoções em outra, o que garantirá que os processos sejam executados por ordem de chegada na fila.

(4b) Suponha que a estrutura de dados em questão armazena apenas as identificações dos processos em espera (estas identificações são números inteiros). Suponha também que a estrutura de dados seja um vetor com n posições. Descreva o algoritmo de inserção de um novo processo nesta estrutura.

Resposta: Seja V o vetor com n posições que armazena a fila. Sejam f e r os valores da frente e da retaguarda da fila. Seja id a identificação do processo em espera.

Algoritmo:

prov:=r mod n + 1
se $prov \neq f$ então
    $r := prov$
    $V[r] := id$
    se $f = 0$ então
        $f := 1$
    senão
        overflow

(4c) Supondo as mesmas condições do item anterior, descreva agora o algoritmo para remoção do processo mais antigo presente na estrutura.

Resposta:

Algoritmo:

se $f \neq 0$ então
    $removido = V[f]$
    se $f = r$ então
        $f := r := 0$
    senão
        f:=f mod n + 1

senão

underflow