Algoritmos e Estruturas de Dados I

Prof. Jurair Rosa

PROVA - VALOR: 10 PONTOS

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

CEFET/RJ
Uned Petrópolis

Turma: 3^o período Data: 14/03/2022

Instruções:

- A prova deve ser feita integralmente sem consulta.
- Essa prova tem duração de 100 minutos.
- É permitido responder às questões a lápis, desde que fique legível.
- A compreensão do enunciado faz parte da avaliação. Portanto, dúvidas relativas às questões **NÃO** serão respondidas.
- Ao término anexe a(s) folha(s) com as respostas no MS Teams.

Questão 1) Considere o algoritmo que implementa o seguinte processo: uma coleção desordenada de elementos é dividida em duas metades e cada metade é utilizada como argumento para a reaplicação recursiva do procedimento. Os resultados das duas reaplicações são, então, combinados pela intercalação dos elementos de ambas, resultando em uma coleção ordenada. Qual é a complexidade desse algoritmo? (2,0 ptos)

a) $O(n^2)$.
b) $O(n^{2n})$.
c) $O(2^n)$.
d) $O(\log\, n \times \log\, n)$.
e) $O(n \times \log\, n)$.

Questão 2) Um programador propôs um algoritmo não-recursivo para o percurso em pré-ordem de uma árvore binária com as seguintes características: (2,0 ptos)

- Cada nó da árvore binária é representado por um registro com três campos: `chave`, que armazena seu identificador; `esq` e `dir`, ponteiros para os filhos esquerdo e direito, respectivamente;
- O algoritmo deve ser invocado inicialmente tomando o ponteiro para o nó raiz da árvore binária como argumento;
- O algoritmo utiliza `push()` e `pop()` como funções auxiliares de empilhamento e desempilhamento de ponteiros para nós de árvore binária, respectivamente.

A seguir, está apresentado o algoritmo proposto, em que λ representa o ponteiro nulo.

```
1  Procedimento preordem (ptraiz : PtrNoArvBin)
2    Var ptr : PtrNoArvBin;
3    ptr := ptraiz;
4    Enquanto (ptr ≠ λ) Faca
5      escreva(ptr↑.chave);
6      Se (ptr↑.dir ≠ λ) Entao
7        push(ptr↑.dir);
8      Se (ptr↑.esq ≠ λ) Entao
9        push(ptr↑.esq);
10     ptr := pop();
11   Fim_Enquanto
12 Fim_Procedimento
```

Com base nessas informações e supondo que a raiz de uma árvore binária com n nós seja passada ao procedimento `preordem()`, julgue os itens seguintes:

I. O algoritmo visita cada nó da árvore binária exatamente uma vez ao longo do percurso.

II. O algoritmo só funcionará corretamente se o procedimento `pop()` for projetado de forma a retornar λ caso a pilha esteja vazia.

III. Empilhar e desempilhar ponteiros para nós da árvore são operações que podem ser implementadas com custo constante.

IV. A complexidade do pior caso para o procedimento `preordem()` é $O(n)$.

Assinale a opção correta:

a) Apenas um item está certo.
b) Todos os itens estão certos.
c) Apenas os itens I e IV estão certos.
d) Apenas os itens I, II e III estão certos.
e) Apenas os itens II, III e IV estão certos.

Questão 3) A sequência de Fibonacci é uma sequência de números inteiros que começa em 0, a que se segue 1, e na qual cada elemento subsequente é a soma dos dois elementos anteriores. A função `fib` a seguir calcula o n-ésimo elemento da sequência de Fibonacci: (2,0 ptos)

```
unsigned int fib(unsigned int n)
{
  if (n <= 1)
    return n;
  return fib(n-2) + fib(n-1);
}
```

Considerando que o programa acima não reutilize resultados previamente computados, quantas chamadas são feitas à função para computar `fib(5)`? Justifique sua resposta apresentação a pilha de recursão.

a) 11.
b) 12.
c) 15.
d) 24.
e) 25.

Questão 4) Uma pilha é uma estrutura de dados que armazena uma coleção de itens de dados relacionados e que garante o seguinte funcionamento: o último elemento a ser inserido é o primeiro elemento a ser removido. É comum na literatura utilizar os nomes *push* e *pop* para as operações de inserção e remoção de um elemento em uma pilha, respectivamente. O seguinte trecho de código em linguagem C define uma estrutura de dados pilha utilizando um vetor de inteiros, bem como algumas funções para sua manipulação. (2,0 ptos)

```
#include <stdlib.h>
#include <stdio.h>

typedef struct {
  int elementos[100];
  int topo;
}pilha;

pilha * cria_pilha() {
  pilha * p = malloc(sizeof(pilha));
  p->topo = -1;
  return pilha;
}

void push(pilha *p, int elemento) {
  if (p->topo >= 99)
    return;
  p->elementos[++p->topo] = elemento;
}

int pop(pilha *p) {
  int a = p->elementos[p->topo];
  p->topo--;
  return a;
}
```

```
26
27 int main() {
28   pilha * p = cria_pilha();
29   push(p, 2);
30   push(p, 3);
31   push(p, 4);
32   pop(p);
33   push(p, 2);
34   int a = pop(p) + pop(p);
35   push(p, a);
36   a += pop(p);
37   printf("%d", a);
38   return 0;
39 }
```

Avalie as afirmações a seguir:

I. A complexidade computacional de ambas as funções `push` e `pop` é $O(1)$.

II. O valor exibido pelo programa seria o mesmo caso a instrução da linha 36 fosse substituída por `a += a;`.

III. Em relação ao vazamento de memória (*memory leak*), é opcional chamar a função `free(p)`, pois o vetor usado pela pilha é alocado estaticamente.

É correto o que se afirma em:

a) I, apenas.
b) I e III, apenas.
c) II e III, apenas.
d) I e II, apenas.
e) I, II e III.

Questão 5) O algoritmo de Kruskal utiliza Estruturas de Dados para Conjuntos Disjuntos para criar uma árvore geradora mínima, tendo como entrada um conjunto de vértices e arestas (cada uma contendo os vértices 1 e 2, e seu respectivo peso). Utilizando os dados a seguir, apresente o passo-a-passo deste algoritmo e, por fim, defina a árvore geradora mínima e seu peso total: (2,0 ptos)

Vértices = {0, 1, 2, 3, 4, 5}

Arestas = {(0,1,1), (0,2,5), (1,2,2), (1,3,5), (1,4,2), (2,4,2), (3,4,1), (3,5,2), (4,5,4)}