Universidade Federal da Bahia (UFBA)
Instituto de Matemática e Estatística (IME)
Departamento de Ciência da Computação (DCC)
MATA 40 - Estrutura de Dados e Algoritmos I - Prof. George Lima

Lista de Exercícios – 01

Apresentando justificativas convincentes, responda as questões abaixo.

1. Algoritmos A e B possuem tempos de execução com complexidade $\Theta(n \log n)$ e $\Theta(n^{3/2})$, respectivamente. Qual deles é mais eficiente em termos assintóticos?

2. Considere que $f(n) = O(s(n))$ e $g(n) = O(r(n))$. Assinale Falso ou Verdade para as seguintes afirmações, demonstrando suas respostas:

   (a) $f(n) - g(n) = O(s(n) - r(n))$.
   (b) $f(n) + g(n) = O(s(n) + r(n))$.
   (c) $f(n) \times g(n) = O(s(n) \times r(n))$.
   (d) $f(n) \div g(n) = O(s(n) \div r(n))$.

3. É correto ou incorreto afirmar que:

   (a) $(\log n)^a = O(n^b)$, para qualquer b e qualquer a positivo?
   (b) $\log_{10} n = O(\log_2 n)$?
   (c) $2^{2n} = O(2^n)$?
   (d) $f(n) = \Theta(f(n/2))$?
   (e) $\log n! = O(n \log n)$?
   (f) $\sum_1^n (1/2)^i = \Theta(1)$?
   (g) Se $T(\frac{n}{100}) = c\, n \log_2 n + n$, para algum $c > 0$, então $T(n) = O(n \log n)$?

4. Qual a complexidade de tempo de execução dos algoritmos abaixo, em notação Θ?

   (a)
```
// Recebe um vetor de inteiros com b-a+1 elementos
// Devolve o indice de um elemento particular apos processar o vetor
int part(int *p, int a, int b) {

  int v = p[a], l = a; r = b, w;

  while (l < r) {
    while(p[l] <= v && l <= b) l++;
    while(p[r] > v  && r >= a) r--;
    if (l < r) {
      w = p[l];
      p[l] = p[r];
      p[r] = w;
    }
  }
  p[a] = p[r];
  p[r] = v;
  return(r);
}
```

```c
// Recebe um vetor de inteiros p com n elementos
// Chama a funcao da questao anterior
int f(int *p, int n) {

  int a = (n+1)/2, b = (n+a)/2, i = 0;
  for (; a <= b; a++)
    i += part(p,a,b);
  return (i);
}
```

(15) Desenvolva as funções listadas a seguir para operar sobre conjuntos. Para tanto, considere o tipo `set_t` como apresentado abaixo.

```c
typedef struct set {
  int num_elem; // numeros de elementos no conjunto
  int *v; // ponteiro para a sequencia de elementos que forma o conjunto
} set_t;
```

Para cada uma destas funções, forneça a complexidade de tempo de execução para o código que você construiu. Use alocação e liberação de memória de forma dinâmica.

(a) `set_t * new_set(int v[])`. Cria um novo conjunto a partir de um vetor de inteiros.

(b) `set_t * union_set(set_t *a, set_t *b)`. Retorna o conjunto formado pela união de dois conjuntos.

(c) `set_t * inter_set(set_t *a, set_t *b)`. Retorna o conjunto formado pela intersecção de dois conjuntos.

(d) `set_t * diff_set(set_t *a, set_t *b)`. Retorna o conjunto correspondente à diferença entre o primeiro conjunto e o segundo.

(e) `boolean is_set(int *v[])`. Verifica se um dado vetor pode ser considerado conjunto.

(f) `boolean eq_set(set_t *a, set *b)`. Verifica se dois conjuntos são iguais.

(g) `boolean is_element_set(set_t *a, int k)`. Checa se um dado valor é elemento do conjunto.

(h) `set_t * add_element_set(set_t *a, int k)`. Adiciona um elemento ao conjunto.

(i) `set_t * del_element_set(set_t *a, int k)`. Remove um elemento do conjunto, caso este exista.

6. Reflita sobre a questão anterior caso deseje-se que conjuntos sejam genéricos, com elementos que podem ser de tipos pre-definidos na linguagem C (considere `int`, `float`, `double`, e `char`). Por exemplo, $A = \{'a', 12.3, 45\}$ seria um conjunto genérico. Como você construiria tais conjuntos em C? Como seriam definidos os elementos e estruturadas as funções da questão anterior?

7. Em C não existe o tipo 'cadeia de caracteres'. Este é tratado como ponteiro para o tipo `char`. Construa o tipo `string` de forma a deixar as operações sobre cadeias de caracteres transparentes e mais legível. Declarações do tipo `string_t s;` devem ser permitidas. A variável `s`, declarada `string_t`, contém como atributos seu tamanho e o ponteiro para a cadeia de caracteres. As seguintes funções para lidar com variáveis do tipo `string_t` devem ser construídas.

(a) `string_t new_str_from_char(char *s)`

(b) `string_t new_str_from_str(string_t s)`

(c) `string_t concat_str(string_t s, string_t u)`

(d) `string_t invert_str(string_t s, string_t u)`

8. Apresente um código em C para resolver o seguinte problema.

(a) Deseja-se construir uma matriz para armazenar valores, mas o número de linhas l é desconhecido. O número de elementos em cada linha i, c_i, só é fornecido quando os elementos desta linha são inseridos. Enfim, apenas durante a execução tem-se conhecimento dos valores l e c_i. Além disso, cada linha pode conter valores inteiros `int`, reais `float`, reais com dupla precisão `double` ou caracteres `char`. Estruture a matriz para que seja possível ter esta versatilidade. Deve-se ter um campo indicando o tipo de elemento em cada linha e outro para armazenar c_i. Use `typedef` e `enum` para deixar seu código mais legível.

(b) Construa uma função que adiciona uma nova linha à matriz. Além da referência à matriz, a função deve receber o indicador de tipo do dado a ser armazenado e um ponteiro para o vetor que contém estes dados.

(c) Construa uma função que recebe uma referência para a matriz e lê, do teclado, cada uma de suas linhas, alocando a memória necessária para conter seus elementos.

(d) Elabore uma função que remove uma linha da matriz, liberando a memória correspondente e outra que remove todo o conteúdo da matriz, também devolvendo toda a memória.