DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MEEC)

Algoritmos e Estruturas de Dados
Aula prática #03

Conteúdo

1 Objectivos

Nesta aula abordam-se temas relacionados com análise de complexidade algorítmica, notação assimptótica e resolução de recorrências. Serão resolvidos alguns problemas para cada um daqueles tópicos.

No final da aula os alunos deverão:

- compreender a metodologia para determinação da complexidade algorítmica com base na contabilização de instruções básicas, para efeitos de determinação da complexidade temporal ou de requisitos de memória;
- entender e saber usar as propriedades da notação assimptótica;
- produzir recorrências para determinação de complexidade algorítmica em programas que façam uso de recursividade;
- saber resolver recorrências pelo método telescópico ou através do Master Theorem.

2 Plano de aula

Para atingir os objectivos anteriormente listados propõe-se o seguinte plano de aula.

1. Considere o código abaixo apresentado e indique, como função de $N = r - l + 1$, em que `l` é o índice do primeiro elemento da tabela `tab1` e `r` o índice do último elemento, como evolui o tempo total de computação com base na determinação do número de instruções básicas.

```
int * funcao_1(int * tab1, int l, int r)
{
  int i, j;
  for (i = l+1; i <= r; i++)
   for (j = i; j > l; j--) {
    int x;
    if (tab1[j] < tab1[j-1]) {
     x = tab1[j];
     tab1[j] = tab1[j-1];
     tab1[j-1] = x;
    }
   }
  return tab1;
}
```

2. Considere a função abaixo e seja $N = r - l + 1$, com `l` o índice do primeiro elemento da tabela `tab2` e `r` o índice do último elemento.

```
float funcao_2(float *tab2, int l, int r)
{
  int i, j, k, contador=0;
  float v, out=0;
  v = tab2[r]; i = l; j = r;
  while (i <= r && test(tab2[i], v, i)) {
   out = out + tab2[i];
   contador += 1;
   i++;
  }
  printf("Elementos que passam o teste:%d\n", contador);
  printf("A sua soma é %g\n", out);
  for (j = l; j <= r; j++)
   for (k = r; k > l+1; k -=(k-l)/2)
    tab2[j] += tab2[k];
  return out;
}
```

   a) Assumindo que a função função `test` tem complexidade $\mathcal{O}(1)$, determine a complexidade temporal de `funcao_2`.
   b) Assumindo que a função função `test` tem complexidade $\mathcal{O}(i)$, em que `i` é o terceiro argumento, determine a complexidade temporal de `funcao_2`.

3. Considere a definição abaixo

$$g(N) \in \mathcal{O}(f(n)) \text{ sse } \exists c_0 \in \mathbb{R}^+, N_0 \in \mathbb{N}, \text{ tais que } g(N) < c_0 f(N), \forall N > N_0, \quad (1)$$

   e prove os seguintes resultados.

   a) $N^2 + 10N \in \mathcal{O}(N^2)$
   b) $N \lg N \in \mathcal{O}(N^2)$

4. Considerando o seguinte teorema

$$g(N) \in \mathcal{O}(f(N)) \text{ sse } \lim_{N \to \infty} \frac{g(N)}{f(N)} < \infty, \quad (2)$$

   indique quais das seguintes afirmações são verdadeiras.

   a) $\lg N \in \mathcal{O}(\sqrt{N})$
   b) $\sum_{i=0}^{N} a^i \in \mathcal{O}(a^N), a > 1$
   c) $N^2 \lg N \in \mathcal{O}(N \lg^2 N)$

5. Considere o seguinte código

```
int funcao_3(int N)
{
  if (N < 1)
   return 0;
  else return N + funcao_3(N/2);
}
```

Escreva uma recorrência que descreva a complexidade temporal da função. Resolva-a pelo método telescópico.

6. Considere o código abaixo

```
void funcao_4 (Line* mat, int N) {
  if (N == 0) return;
  mat->storage = (int*) malloc (N * sizeof(int));
  if (mat->storage == NULL) {
   fprintf(stderr,"ERRO: Memória insuficiente\n");
   exit(2);
  }
  mat->next = (Line*) malloc(1*sizeof(Line));
  if (mat->next == NULL) {
   fprintf(stderr,"ERRO: Memória insuficiente\n");
   exit(3);
  }
  funcao_4(mat->next, N-1);
}
```

Determine uma recorrência que descreva a complexidade de memória da função.

7. Resolva as seguintes recorrências pelo Master Theorem.

   a) $C_N = 3C_{N/2} + N^{\lg_3 2}$
   b) $C_N = 5C_{N/3} + N^2 \lg N$
   c) $C_N = 4C_{N/2} + N^2 \lg^3 N + N^{3/2}$