Faculdade de Computação (FACOM) - UFMS
Algoritmos e Programação 2
Prof. Dr. Anderson Bessa da Costa

Lista 7 - Noções de Eficiência

1. Como um exemplo de análise de tempo de execução, nós trabalharemos com o programa `three_sum` (disponibilizado no AVA como `three_sum.cpp`) mostrado aqui, que conta o número de triplas em um arquivo de N inteiros que somam 0. Essa computação pode parecer trivial para você, mas é profundamente relacionada com numerosas tarefas de computação fundamentais. Estude o código `three_sum.cpp` apresentado a seguir.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  /* Prototipos */
5  void leia_ints (char nome_do_arquivo[], int a[], int &n);
6  int conte (int a[], int n);
7
8  /* Main */
9  int main() {
10   char filename[30];
11   int a[1000000], n;
12
13   // Leia o nome do arquivo
14   printf("Entre com o nome do arquivo:\n");
15   scanf("%29s", filename);
16
17   // Carregue o arquivo no vetor a
18   leia_ints(filename, a, n);
19
20   // Conte e imprima quantas vezes 3 elementos somam 0
21   printf("%d\n", conte(a, n));
22 }
23
24 // Funcao que abre arquivo e carrega todos os ints para o vetor a
25 void leia_ints (char nome_do_arquivo[], int a[], int &n) {
26   FILE *f;
27   int nread;
28
29   f = fopen(nome_do_arquivo, "r");
30   if (f == NULL) {
31     perror("Erro ao abrir o arquivo");
32   }
33   else {
34     n = 0;
35     while(1) {
36       nread = fscanf(f, "%d", &a[n]);
37       if (nread == EOF)
38         break;
39
40       n = n + 1;
41     }
42     n = n - 1;
43   }
```

```
44 }
45
46 // Funcao que conta e retorna quantas vezes 3 elementos somam 0
47 int conte (int a[], int n) {
48   int i, j, k, c = 0;
49
50   for (i = 0; i < n; i++) {
51     for (j = i + 1; j < n; j++) {
52       for (k = j + 1; k < n; k++) {
53         if ( (a[i] + a[j] + a[k]) == 0)
54           c = c + 1;
55       }
56     }
57   }
58
59   return c;
60 }
```

2. Como um primeiro experimento, execute o código `three_sum` no seu computador para os conjuntos de arquivos de entrada fornecidos: `1Kints.txt`, `2Kints.txt`, `4Kints.txt`, e `8Kints.txt`. Esses arquivos de entrada estão disponibilizados no AVA.

   Quantas triplas existem no arquivo `1Kints.txt`? E no arquivo `2Kints.txt`? E no restante de arquivos de entradas `4Kints.txt` e `8Kints.txt`?

3. O arquivo de entrada `1Mints.txt` contém 1 milhão de números inteiros. O algoritmo `three_sum` consegue nos dizer quantas triplas existem que somam 0, porém ele consegue fazê-lo em uma quantidade razoável de tempo? (Por enquanto, apenas reflita sobre essa questão. Você conseguirá respondê-la ao final desta lista de exercícios.)

4. Frequentemente você se perguntará a seguinte questão: *"Quanto tempo meu programa levará?"*. Como você verá, responder essa questão para esse programa é relativamente fácil. Nós apresentaremos uma predição precisa para medir o tempo que o programa leva para a execução.

   Mensurar de forma confiável o tempo de execução exato de um dado programa pode ser difícil. Felizmente, em geral estamos satisfeitos com estimativas. Nós queremos ser capazes de distinguir programas que terminam em alguns segundos ou alguns minutos daqueles que podem requerer dias, meses ou mais. Ademais, estamos interessados em saber quando um programa é duas vezes mais rápido que outro programa para a mesma tarefa.

   Estudo o código `stopwatch.cpp` fornecido a seguir.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  int conte (int a[], int n);
6
7  /* Main */
8  int main() {
9    char filename[30];
10   int a[1000000], n, i;
11   double time1, timedif;
12
13   srand(time(NULL));
```

```
14
15    // Leia um valor n
16    printf("n:\n");
17    scanf("%d", &n);
18
19    // Preencha o vetor a com n numeros inteiros aleatorios
20    for (i = 0; i < n; i++)
21      // numeros entre -10000 e 9999
22      a[i] = (rand() % 20000) - 10000;
23
24    time1 = (double) clock();              /* get initial time */
25    time1 = time1 / CLOCKS_PER_SEC;        /*    in seconds    */
26
27    // Conte e imprima quantas vezes 3 elementos somam 0
28    printf("%d\n", conte(a, n));
29
30    // Calcule e imprima o tempo gasto
31    timedif = ( ((double) clock()) / CLOCKS_PER_SEC) - time1;
32    printf("The elapsed time is %lf seconds\n", timedif);
33  }
34
35  // Funcao que conta e retorna quantas vezes 3 elementos somam 0
36  int conte (int a[], int n) {
37    int i, j, k, c = 0;
38
39    for (i = 0; i < n; i++) {
40      for (j = i + 1; j < n; j++) {
41        for (k = j + 1; k < n; k++) {
42          if ( (a[i] + a[j] + a[k]) == 0)
43            c = c + 1;
44        }
45      }
46    }
47
48    return c;
49  }
```

5. Utilizando do código `stop_watch` apresentado na questão 4 (e disponível no AVA como `stopwatch.cpp`), qual o tempo necessário para executar para os tamanhos de entrada:

   (a) 1000

   (b) 2000

   (c) 4000

   (d) 8000

6. Compare os tempos obtidos com algum colega de sala. Os tempos são similares?

7. O programa `doubling_test` a seguir (disponibilizado no AVA como `doubling_test.cpp`) é mais sofisticado que `stopwatch`, produzindo dados experimentais para o `three_sum`. Quando você executa `doubling_test`, você observará que as primeiras linhas são impressas rapidamente, entretanto desacelerará consideravelmente. A cada linha impressa, você se perguntará quanto tempo levará até imprimir a próxima linha. Obviamente, uma vez

que você execute em máquinas diferentes, o tempo de execução que se obterá provavelmente será diferente. Execute `doubling_test`, fornecendo 10000 (dez mil) como valor de N. Anote os tamanhos e tempos obtidos.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  int conte (int a[], int n);
6
7  /* Main */
8  int main() {
9    char filename[30];
10   int a[1000000], n, i, k;
11   double time1, timedif;
12
13   // Leia um valor n
14   printf("n: ");
15   scanf("%d", &n);
16
17   srand(time(NULL));
18
19   for (k = 250; k < n; k += k) {
20
21     // Preencha o vetor a com n numeros inteiros aleatorios
22     for (i = 0; i < k; i++)
23         a[i] = (rand() % 20000) - 10000;
24
25     time1 = (double) clock();              /* get initial time */
26     time1 = time1 / CLOCKS_PER_SEC;        /*    in seconds    */
27
28     conte(a, k);
29
30     timedif = ( ((double) clock()) / CLOCKS_PER_SEC) - time1;
31     printf("%7d %5.1lf\n", k, timedif);
32   }
33 }
34
35 // Funcao que conta e retorna quantas vezes 3 elementos somam 0
36 int conte (int a[], int n) {
37   int i, j, k, c = 0;
38
39   for (i = 0; i < n; i++) {
40     for (j = i + 1; j < n; j++) {
41       for (k = j + 1; k < n; k++) {
42         if ( (a[i] + a[j] + a[k]) == 0)
43           c = c + 1;
44       }
45     }
46   }
47
48   return c;
49 }
```

8. Na análise da complexidade de um algoritmo, de maneira geral estamos interessados em responder a seguinte pergunta *"Quanto tempo levará o meu programa para executar, como*

uma função do tamanho da entrada?". Essa é uma pergunta difícil de responder. Para propósitos desta aula prática, irei facilitar e estarei lhe fornecendo a equação do tempo de execução:

$$T(N) = aN^3 \tag{1}$$

onde N é o tamanho da entrada, e a é uma constante.

Considere o tempo que você obteve anteriormente para uma entrada de tamanho 8000 (oito mil). Calcule o valor do a. Reescreva a função substituindo o a com o valor estimado. **Atenção:** o valor de a é específico para a sua máquina.

9. Fazendo uso da equação obtida anteriormente, qual o tempo estimado para se encontrar triplas que somam 0 dado um conjunto de entrada 1 milhão de elementos inteiros?

10. Por fim, volte e responda a questão 3.