Cursos de Verão 2010 - IME/USP
Tópicos de Programação
Mergesort
Prof. Alexandre da Silva Freire

1 Mergesort

Considere o seguinte problema: dados dois vetores crescentes $v[p\,..\,q-1]$ e $v[q\,..\,r-1]$, rearranjar $v[p\,..\,r-1]$ em ordem crescente (diremos que $v[p\,..\,r-1]$ é uma *intercalação* dos vetores $v[p\,..\,q-1]$ e $v[q\,..\,r-1]$). Eis um algoritmo para resolver o problema:

```
    /*
     * A função recebe vetores crescentes v[p..q-1] e
     * v[q..r-1] e rearranja v[p..r-1] em ordem crescente.
     */
    void merge(int v[], int p, int q, int r) {
       int i, j, k, *t;
       t = malloc ((r-p) * sizeof (int));
       i = p; j = q; k = 0;
(3)    while(i < q || j < r) {
          if((i<q && j<r && v[i]<=v[j]) || (i<q && j==r)) {
             t[k++] = v[i++];
          } else {
             t[k++] = v[j++];
          }
       }
(7)    for(i = p; i < r; i++) {
          v[i] = t[i-p];
       }
       free(t);
    }
```

Vejamos os invariantes do algoritmo `merge`:

(i0) *imediatamente antes de qualquer execução da linha (3), vale que o vetor* $t[0\,..\,k-1]$ *contém uma intercalação dos vetores* $v[p\,..\,i-1]$ *e* $v[q\,..\,j-1]$.

(i1) *imediatamente antes de qualquer execução da linha (3), vale que* $t[k-1] \leq v[i]$ *e* $t[k-1] \leq v[j]$.

(i2) *imediatamente antes de qualquer execução da linha (7), vale que o vetor* $v[p\,..\,i-1]$ *contém uma cópia do vetor* $t[0\,..\,i-p-1]$.

Lema 1.1. *O algoritmo* `merge` *está correto.*

Exercício opcional: provar o lema 1.1 utilizando os invariantes (i0), (i1) e (i2).

O consumo de tempo do algoritmo `merge` é $\Theta(r-p)$. Utilizaremos o algoritmo `merge` para desenvolver um algoritmo de ordenação que chamaremos de `mergesort`. A função `mergesort` ordena o vetor $v[p\,..\,r-1]$. Se quisermos ordenar um vetor $v[0\,..\,n-1]$, basta chamar `mergesort` $(v, 0, n)$. Eis o algoritmo:

```
    void mergesort(int v[], int p, int r) {
       int q = (p + r)/2;
       if (p < r-1) {
(3)       mergesort(v, p, q);
(4)       mergesort(v, q, r);
(5)       merge(v, p, q, r);
       }
   }
```

Lema 1.2. *O algoritmo* `mergesort` *está correto.*

Prova. Provaremos por indução em $n = r - p$. Suponha que $n = 1$. Neste caso temos que $r - p = 1 \Rightarrow p = r - 1$. Logo, temos que $v[p\,..\,r-1]$ tem apenas um elemento (logo, o vetor $v[p\,..\,r-1]$ está ordenado). Além disso, a condição $(p < r - 1)$ não é satisfeita e, portanto, o algoritmo não altera o vetor $v[p\,..\,r-1]$. Agora suponha que $n > 1$. Por hipótese de indução, vale que *após a linha (4) os vetores* $v[p\,..\,q-1]$ *e* $v[q\,..\,r-1]$ *estão em ordem crescente*. Pelo lema 1.1, logo após a linha (5) o vetor $v[p\,..\,r-1]$ contém uma intercalação dos vetores $v[p\,..\,q-1]$ e $v[q\,..\,r-1]$. Portanto, o algoritmo `mergesort` está correto. ∎

Seja $T(n)$ o consumo de tempo do algoritmo `mergesort`, sendo que $n = r - p$. Definimos $T(n)$ através da seguinte recorrência:

$$T(n) = \begin{cases} 1, & \text{se } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n, & \text{se } n > 1. \end{cases}$$

Lema 1.3. *Para todo* n *da forma* 2^k*, temos que* $T(n) = n \lg n + n$.

Prova. Provaremos por indução em n. Suponha que $n = 1$. Neste caso temos que $T(n) = 1 = n \lg n + n$, pois por definição $\lg 1 = 0$. Agora suponha que $n > 1$. Neste caso temos que $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n = 2T(\frac{n}{2}) + n$, sendo que a última igualdade vale pelo fato de n ser potência de 2 (repare que só o fato de n ser par não seria suficiente para completar a indução – pense no assunto). Por hipótese de indução vale que $T(\frac{n}{2}) = \frac{n}{2} \lg \frac{n}{2} + \frac{n}{2}$.

Logo, temos que $T(n) = 2(\frac{n}{2}\lg\frac{n}{2}+\frac{n}{2})+n = n\lg\frac{n}{2}+2n = n\lg\frac{n}{2}+n+n\lg 2 = n(\lg\frac{n}{2}+\lg 2)+n = n\lg n + n$. ■

Corolário 1.1. $T(n) = \Theta(n\lg n)$.

Prova. Note que para todo número inteiro n existe um inteiro k tal que $2^k \leq n \leq 2^{k+1}$. Portanto, podemos aplicar o lema 1.3 para concluir que $n\lg n + n \leq T(n) \leq 2n\lg 2n + 2n \Rightarrow T(n) = \Theta(n\lg n)$. ■