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

1 Busca binária

Considere o seguinte problema: dado um vetor $v[0\,..\,n-1]$ de inteiros em ordem crescente (ou seja, $v[0] \leq v[1] \leq \ldots \leq v[n-1]$), devolver o índice de um elemento e em v, ou devolver -1 caso e não esteja contido em v. O algoritmo que chamaremos de *buscabin* recebe um inteiro e e um vetor crescente $v[i\,..\,f]$, e devolve um índice m em $i\,..\,d$ tal que $v[m] = e$, ou se tal m não existe, o algoritmo devolve -1.

```
    int buscabin(int e, int i, int f, int v[]) {
(1)    if (i > f) {
(2)       return -1;
       }
       else {
(3)       int m = (i + f)/2;
(4)       if (v[m] == e) {
(5)          return m;
(6)       } else if (v[m] < e) {
(7)          return buscabin(e, m+1, f, v);
          } else {
(8)          return buscabin(e, i, m-1, v);
          }
       }
    }
```

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

Prova. Por indução em $n = f - i + 1$. Suponha que $n = 0$. Neste caso vale que $i > f$ e, portanto, o algoritmo devolve -1. Como o intervalo $v[i\,..\,f]$ é vazio, o algoritmo devolve a resposta correta. Agora suponha que $n > 0$. Neste caso $i \leq f$ e, portanto, o algoritmo compara e com $v[m]$. Temos três casos a analisar. Se $v[m] = e$, então o algoritmo devolve m, que é uma resposta correta, pois m é um índice de um elemento e em v. Se $v[m] < e$, então certamente o vetor $v[i\,..\,m-1]$ não contém e, pois v está em ordem crescente. Neste caso o algoritmo devolve o resultado da chamada recursiva da linha (7), que por hipótese de indução devolve o índice de e no vetor $v[m+1\,..\,f]$ ou -1 caso e não esteja no respectivo vetor. Logo, quando $v[m] < e$ o algoritmo devolve a resposta correta. Se $v[m] > e$, então certamente o vetor $v[m+1\,..\,f]$ não contém e, pois v está em ordem crescente. Neste caso, o algoritmo devolve o resultado da chamada recursiva da linha (8), que por hipótese de indução devolve o índice de e no vetor

$v[i \,..\, m-1]$ ou -1 caso e não esteja no respectivo vetor. Logo, quando $v[m] > e$ o algoritmo devolve a resposta correta.

Como em todos os possíveis casos o algoritmo devolve a resposta correta, segue que o algoritmo `buscabin` está correto. ∎

Lema 1.2. *O algoritmo* `buscabin` *consome tempo* $O(\lg n)$.

Prova. O pior caso para este algoritmo ocorre quando o elemento e não está contido em v. Considere a seguinte recorrência:

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

Claramente $T(n)$ é um limitante superior para o consumo de tempo do algoritmo `buscabin`. Mostraremos, por indução em n, que se n é potência de dois então $T(n) = \lg n + 1$.

Suponha que $n = 1$. Neste caso $T(n) = 1 = \lg n + 1$. Agora suponha que $n > 1$. Neste caso $T(n) = T(\frac{n}{2}) + 1$. Por hipótese de indução vale que $T(\frac{n}{2}) = \lg \frac{n}{2} + 1$. Portanto, $T(n) = \lg \frac{n}{2} + 1 + 1 = \lg \frac{n}{2} + \lg 2 + 1 = \lg n + 1$.

Com isso, concluímos que $T(n) = \Theta(\lg n)$ e, portanto, o algoritmo `buscabin` consome tempo $O(\lg n)$. ∎

Eis uma versão iterativa do algoritmo de busca binária:

```
    int buscabin_it(int e, int n, int v[]) {
       int i, m, f;
       i = 0; f = n-1;
(1)    while (i <= f) {
          m = (i + f)/2;
          if (v[m] == e) {
             return m;
          } else if (v[m] < e) {
             i = m + 1;
          } else {
             f = m - 1;
          }
       }
       return -1;
    }
```

Exercício opcional: prove que o algoritmo `buscabin_it` está correto utilizando o seguinte invariante: *imediatamente antes de qualquer execução da linha (1), vale que* $v[i-1] < e < v[f+1]$. Outro exercício opcional: qual é o consumo de tempo do algoritmo `buscabin_it`?