PRÁTICA 2 – Trabalho 2 Exercício 2: Análise por operações primitivas

Nome: Davi Gabriel Domingues

Número USP: 15447497

O trabalho consiste na implementação em C++ de uma função que, dado um vetor **v** com n inteiros e um inteiro x (chave de busca), retorne o índice da segunda ocorrência de x em **v**; sendo que caso não o encontre no vetor **v**, a função deve retornar o valor -1.

A busca é interrompida assim que o índice da segunda ocorrência de x em **v** é encontrado ou quando se chega no último elemento do vetor em questão.

Para esse propósito, foi – se desenvolvido o programa a seguir:

a) O código para essa situação é o seguinte:

```
1
2   //letra a)
3   #include <iostream>
4   using namespace std;
5
6   int count = 0;
7
8   int buscaVetor(int x, int v[], int n){
9       for (int i = 0; i < n; i++){
10          if (x == v[i])
11              count++;
12
13          if (count == 2)
14              return i;
15      }
16          return -1;
17  }
```

```
18
19  int main(){
20      int n, encontrado;
21      float x;
22
23      cout<<"Informe o tamanho do vetor de interesse: ";
24      cin>>n;
25
26      int *v = new int[n];
27
28      for (int i = 0; i < n; i++){
29          cout<<"Valor "<<i + 1<<": ";
30          cin>>v[i];
31      }
32
33      cout<<"Informe a chave de busca numerica para o vetor: ";
34      cin>>x;
35
36      encontrado = buscaVetor(x, v, n);
```

```
37
38  if (encontrado > 0)
39      cout<<"O numero foi encontrado pela segunda vez no vetor na posicao "<<encontrado;
40
41  else{
42      if (count == 1)
43          cout<<"O numero foi encontrado apenas uma vez na estrutura do vetor";
44      else
45          cout<<"O numero foi encontrado nenhuma vez na estrutura do vetor";
46  }
47
48  return 0;
49 }
```

b) Tem-se os cálculos e afirmações a seguir:

Para o caso geral, verifica-se o seguinte:

O algoritmo de "a)" é este:

```
1  buscaVetor(int x, int v[], int n){
2        int count = 0;
3        for(int i=0; i<n; i++){
4            if(x == v[i])
5                count++;
6            if(count == 2)
7                return i;
8        }
9        return -1;
10 }
```

Vê-se que na:

- linha 2, há uma operação (count = 0);
- linha 3, há três tipos de operação, as quais:
  - i = 0 ocorre uma vez
  - i < n ocorre m vezes ($\sum_{0}^{m-1} 1 = m-1+1 = m$)
  - i++ ocorre m - 1 vezes (porque $\sum_{0}^{m-2} 1 = m-2+1 = m-1$)
- linha 4, há duas operações, "==" e "v[i]", que ocorrem, cada, m vezes.
- linha 5, há uma operação (count++) a qual ocorrerá duas vezes.
- linha 6, há uma operação (count == 2) que ocorrerá m vezes.
- linha 7, há uma operação única (return i).

Assim:

$f(m) = 1 + 1 + m + m - 1 + 2m + 2 + m + 1$

$\Rightarrow \boxed{f(m) = 5m + 4}$, $2 \leq m \leq n$

c) A função expressa em “a)” modificada que imprime o número de operações primitivas executadas em suas operações primitivas é esta:

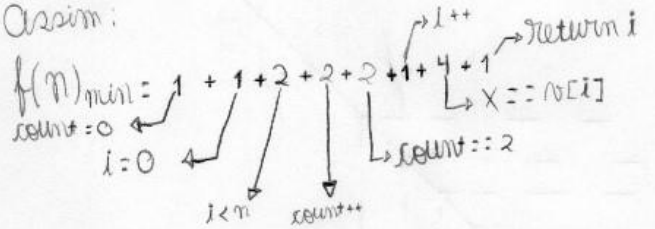
```
58  int buscaVetor(int x, int v[], int n){
59      int count = 0;
60      operacoes += 2; // count = 0 e i = 0.
61      for (int i = 0; i < n; i++){
62          operacoes++; //durante os casos em que i < n.
63          operacoes += 2; // == e v[i], ambos de x == v[i].
64          if (x == v[i]){
65              count++;
66              operacoes ++; //relacionado ao count.
67          }
68
69          operacoes++; //relacionado ao == de count == 2
70          if (count == 2){
71              operacoes++; //relacionado ao return i.
72              return i;
73          }
74
75          if (i < n) //relacionado ao i++.
76              operacoes++;
77      }
78      operacoes += 2; //relacionado ao return -1 e ao caso em que i == n.
79      return -1;
```

d) A análise por operações primitivas da função buscaVetor referenciada pelo código descrito em “a)” é esta, além de que, para os casos máximos e mínimos, tem -se os seguintes números de operações primitivas, considerando a modificação descrita em “c)”, para diferentes valores de n:

(I) função mínima:

. Ocorrerá se os números forem iguais e estiverem presentes em ordem consecutiva de posições nos vetores, mas de preferência nas posições 0 e 1, ou seja, $X = v[0] = v[1]$.

Assim:

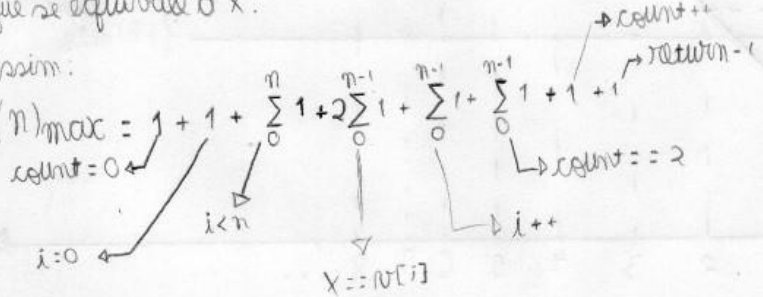


$\therefore$ $f(n)_{min} = 14$, $n \geq 2$

(II) função máxima:

. Ocorrerá se houver apenas um valor qualquer em v que se equivalha a X.

Assim:



$\therefore f(n)_{max} = 4 + 4(n-1+1) + (n+1)$

$\Rightarrow f(n)_{max} = 5n + 5$, $n \geq 1$

. Obs¹: As operações anexadas pelas setas em cada membro das funções indicam quantas vezes foram executadas em cada caso de execução.

(III) Caso geral

$$14 \leq f(n)_{geral} \leq 5n + 5, \quad n \geq 2$$

. Obs²: n representa o tamanho genérico do vetor v declarado em "a)": `int *v = new int[n];`

Tem-se os respectivos gráficos de *n* por número de operações primitivas: