Nome: Rafael de Almeida NºUSP: 11872028

Código:

```
int busca(int* vet, int chave, int n){
   for(int i = 0; i < n; i++){
       if(vet[i] == chave) return i;
   }
   return -1;
}
```

Considerando as operações relevantes para este problema como:

F - Chamada de função;
A - Atribuição;
O - Operação aritmética;
L - Operação lógica;
V - Acesso a vetor;
R - Retorno de função.

Temos dois casos:

Melhor Caso:

int busca(int vet, int chave, int n){* 1 acesso a função: F

for(int i = 0; i < n; i++){ 1 atribuição e 1 comparação: A + L

if(vet[i] == chave) return i; 1 acesso a vetor, 1 comparação e 1 retorno de função: V + L + R

}
return -1;
}

Somando tudo temos: 1F + 1A + 2L + 1V + 1R,
Considerando todos as operações iguais a 1 temos:

$$1 + 1 + 2 + 1 + 1$$
$$6$$

6 é o número de operações primitivas do melhor caso.

Pior Caso:

int busca(int vet, int chave, int n){*	1 acesso a função: F
for(int i = 0; i < n; i++){	1 atribuição, n+1 comparações(ele compara "n" também só não acessa o for a partir daí e encerra o loop) , n operações aritméticas e n atribuições(i++ --> i = i + 1): A + L*(n+1) + O*(n) + A*(n)
if(vet[i] == chave) return i;	n acessos a vetor e n comparações(sem retorno de função para este caso): V*(n) + L*(n)
}	
return -1;	1 retorno de função: R
}	

Somando todos os casos temos: [F] + [A + L*(n+1) + O*(n) + A*(n)] + [V*(n) + L*(n)] + [R]
Considerando todos as operações iguais a 1 temos:

$$1 + [1 + n+1 + n + n] + [n + n] + 1$$
$$2 + 2 + 3n + 2n$$
$$5n + 4$$

5n + 4 é o número de operações primitivas do pior caso.