

## CUSTO COMPUTACIONAL

Como o programa baseia-se na geração de uma matriz e nos passos dados até chegar à posição  $n;n$ , pode-se dizer que o custo computacional é o custo da geração da matriz e inicialização das variáveis da struct, somado ao custo da caminhada realizada na matriz.

### CUSTO PARA A GERAÇÃO DA MATRIZ E INICIALIZAÇÃO DA STRUCT

```
void inicio(int matriz[MAX][MAX], int tamanho, contador *c)
{
    printf("\n\n");
    int cont_e, cont_i;
    srand(time(NULL));
    for(cont_e=0; cont_e<tamanho; cont_e++)
    {
        printf(" ");
        for(cont_i=0; cont_i<tamanho; cont_i++)
        {
            matriz[cont_e][cont_i]=rand()%100;
            printf(" %d", matriz[cont_e][cont_i]);
        }
        printf("\n");
    }

    c->soma=matriz[0][0];
    c->cont_steps=0;
    c->tamanho=tamanho;
    c->vet_pos[0]=0;
}
```

Handwritten annotations for the `inicio` function:

- A bracket next to the inner loop (lines 11-15) is labeled  $n^2$ .
- A bracket next to the initialization of `c` (lines 16-20) is labeled 4.
- A large bracket on the right side of the function is labeled  $n^2 + 5$ .

O custo para a geração da matriz é  $n^2 + 5$  com  $n$  (dimensão da matriz) variando de 2 a 10.

### CUSTO PARA ANDAR UMA CASA

```
c->cont_steps++;
c->vet_pos[c->cont_steps]=c->vet_pos[c->cont_steps-1]+1;
c->soma=c->soma+valor_d;
```

```
void andar(int matriz[MAX][MAX], contador *c)
{
    int valor_atual, valor_e, valor_d, valor_b;
    valor_atual=matriz[c->vet_pos[c->cont_steps]/10][c->vet_pos[c->cont_steps]%10];
    valor_e=matriz[(c->vet_pos[c->cont_steps]/10)][(c->vet_pos[c->cont_steps]%10)-1];
    valor_d=matriz[(c->vet_pos[c->cont_steps]/10)][(c->vet_pos[c->cont_steps]%10)+1];
    valor_b=matriz[(c->vet_pos[c->cont_steps]/10)+1][(c->vet_pos[c->cont_steps]%10)];

    if(((c->vet_pos[c->cont_steps]/10)<c->tamanho-1) || ((c->vet_pos[c->cont_steps]%10)<c->tamanho-1))
    {
        else
    }
}
```

Handwritten annotations for the `andar` function:

- A bracket next to the first three lines is labeled 3.
- A bracket next to the four lines of matrix access is labeled 4.
- A large bracket on the right side of the function is labeled 7.

Como o programa pode percorrer toda a matriz para chegar ao final e a matriz tem que ter no mínimo 2 de dimensão e no máximo 10, o custo é  $7 \cdot n$  de forma que  $2 \leq n \leq 91$ .

Concluindo o custo do programa é  $n^2 + 5 + 7n_1$  sendo que  $n$  m é a dimensão da matriz e  $n_1$  é o número de passos dados e depende de  $n$ . Caso a dimensão da matriz pudesse tender ao infinito o custo poderia ser definido por  $n^2$ .

## **HÁ OUTRA FORMA DE RESOLVER ESSE PROBLEMA?**

Existe sim outras formas de resolver esse problema através de programas que verificam instruções.

## **HÁ ALGUM ALGORITMO FAMOSO EM LITERATURA QUE RESOLVA ISSO?**

Esse problema pode ser resolvido pelos algoritmos míopes que buscam o melhor resultado local visando a melhor escolha para a solução global