

PANC: Projeto e Análise de Algoritmos

Projeto 01: Análise e Projeto do Problema 3SUM

Breno Lisi Romano

<http://sites.google.com/site/blromano>

**Instituto Federal de São Paulo – IFSP São João da Boa Vista
Bacharelado em Ciência da Computação – 3º Semestre**



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista



Problema: 3SUM (1)

- Dado um array A com N números inteiros, verificar quantas triplas de elementos a , b e c , presente no array A , tal que $a+b+c = 0$!
 - **Objetivo:** Encontrar quantas triplas únicas do array A que se obtém a soma igual a Zero!
 - **Exemplo:**
 - $A = [] = \{-1, 0, 1, 2, -1, -4\}$
 - Solução:
 - 02 triplas únicas:
 - $[-1, 0, 1]$
 - $[-1, -1, 2]$



Problema: 3SUM (2)

- Desenvolver uma **solução**, na **linguagem C**, utilizando-se o conceito de **força-bruta** para resolver o **problema 3SUM** apresentado no slide anterior, verificando **todas** as **possibilidades de triplas existentes**. Deve-se retornar:
 - Quais são as **triplas encontradas** (a, b e c);
 - **Quantidade** de **triplas** encontrada
 - **Sugestão**: Utilizar laços de repetição
- Apresentar uma análise da complexidade do algoritmo projetado – $T(n)$



Problema: 3SUM (2)

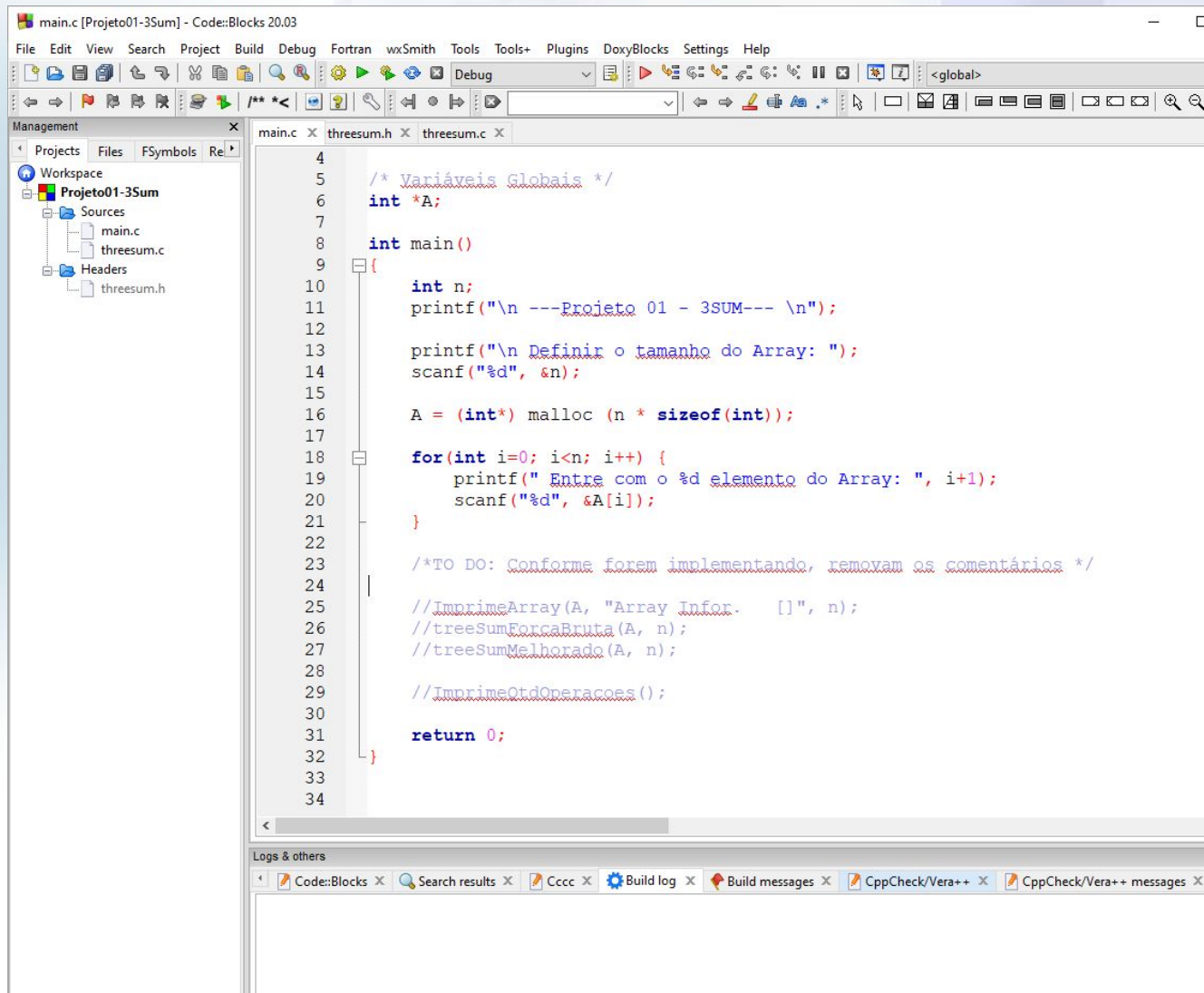
- Complementar a **solução**, na **linguagem C**, melhorando a complexidade $T(n)$, em relação a solução baseada no conceito de **força-bruta** para resolver o **problema 3SUM**
 - **Sugestão:**
 - 1-) **Ordenar** o Array A
 - 2-) Para cada par de elementos do Array A, apoia-se no algoritmo de **busca binária** para verificar a existência de um terceiro elemento que determina se a tripla tem soma ZERO!
 - **Dica:** Quem deve ser encontrado pela busca binária para que a soma com o par de elementos, inicialmente selecionado, seja ZERO?
 - Deve-se retornar:
 - Quais são as **triplas encontradas** (a, b e c);
 - **Quantidade** de **triplas** encontrada
 - Apresentar uma nova análise da complexidade do algoritmo projetado – $T(n)$



Orientações Gerais

- Para a **resolução do problema**:
 - O **array A** deve ser **fornecido** pelo **usuário**
 - **Quantificar a quantidade de operações** para encontrar as triplas, em cada um dos algoritmos, para o mesmo Array fornecido
 - Elaborar um **documento .doc** para apresentar a **explicação dos algoritmos propostos** e também as suas respectivas **análises da complexidades – $T(n)$**
 - Fazer **pesquisas** na **internet** e nos **livros** base fornecidos para **apoiar a resolução** do problema
 - Soluções idênticas serão penalizadas
- Para **execução**:
 - O **Projeto 01** pode ser desenvolvido **em triplas (no máximo)**
 - **Data de Entrega**: 03/05/2023
 - Obrigatório Utilizar o Modelo de Projeto Fornecido
 - **Local da Entrega**: Atividade específica no turma do Google Classroom

Exemplo de Implementação para o Projeto 01: 3SUM (3)



```
main.c [Projeto01-3Sum] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
Management
Projects Files FSymbols Rel
Workspace
Projeto01-3Sum
Sources
main.c
threesum.c
Headers
threesum.h
main.c x threesum.h x threesum.c x
4
5 /* Variáveis Globais */
6 int *A;
7
8 int main()
9 {
10     int n;
11     printf("\n ---Projeto 01 - 3SUM--- \n");
12
13     printf("\n Definir o tamanho do Array: ");
14     scanf("%d", &n);
15
16     A = (int*) malloc (n * sizeof(int));
17
18     for(int i=0; i<n; i++) {
19         printf(" Entre com o %d elemento do Array: ", i+1);
20         scanf("%d", &A[i]);
21     }
22
23     /*TO DO: Conforme forem implementando, removam os comentários */
24
25     //ImprimeArray(A, "Array Infor.   []", n);
26     //treeSumForçaBruta(A, n);
27     //treeSumMelhorado(A, n);
28
29     //ImprimeOtdOperacoes();
30
31     return 0;
32 }
33
34
Logs & others
Code::Blocks x Search results x Cccc x Build log x Build messages x CppCheck/Vera++ x CppCheck/Vera++ messages x
```




Exemplo de Implementação para o Projeto 01: 3SUM (4)

```
C:\Users\bv120133\Desktop\Projeto01-03SUM-ForcaBruta\bin\Debug\Projeto01-03S...

---Projeto 01 - 3SUM---

Definir o tamanho do Array: 8
Entre com o 1 elemento do Array: 30
Entre com o 2 elemento do Array: -40
Entre com o 3 elemento do Array: -20
Entre com o 4 elemento do Array: -10
Entre com o 5 elemento do Array: 40
Entre com o 6 elemento do Array: 0
Entre com o 7 elemento do Array: 10
Entre com o 8 elemento do Array: 5

Array Infor.  [] = 30 -40 -20 -10 40 0 10 5

---3SUM - Forca Bruta:---

1 Tripla Encontrada: [30, -40, 10]
2 Tripla Encontrada: [30, -20, -10]
3 Tripla Encontrada: [-40, 40, 0]
4 Tripla Encontrada: [-10, 0, 10]
Total Triplas Encontradas pela Forca Bruta: 4

---3SUM - Melhorado:---

Array Ord.  [] = -40 -20 -10 0 5 10 30 40
1 Tripla Encontrada: [-40, 0, 40]
2 Tripla Encontrada: [-40, 10, 30]
3 Tripla Encontrada: [-20, -10, 30]
4 Tripla Encontrada: [-10, 0, 10]
Quantidade de Operacoes - 3SUM - Forca Bruta: 56
Quantidade de Operacoes - 3SUM - Melhorado: 45
Process returned 0 (0x0)   execution time : 24.790 s
Press any key to continue.
```