3SUM - Força Bruta

Imagem 1: Função “treeSumForcaBruta()” utilizada:

```
8   /* treeSumForcaBruta(): */
9   void treeSumForcaBruta(int A[], int n) {
10
11      int numeroTripla = 0;
12
13      for(int i=0; i<n-2; i++){
14          qtdOperacoes3SumFB++;
15
16          for(int j=i+1; j<n-1; j++){
17              qtdOperacoes3SumFB++;
18
19              for(int k=j+1; k<n; k++){
20                  qtdOperacoes3SumFB++;
21
22                  if(A[i] + A[j] + A[k] == 0){
23                      numeroTripla++;
24                      printf( " %d Tripla Encontrada: [%d, %d, %d]\n", numeroTripla, A[i], A[j], A[k]);
25                  }
26
27              }
28          }
29      }
30
31      printf( " Total Triplas Encontradas pela Forca Bruta: %d", numeroTripla);
32
33
34  }
```

Explicação do algoritmo proposto:

O algoritmo se baseia na criação de três estruturas de repetição “for”, que percorrem o array em busca de combinações que resultem em 0. O início das estruturas de repetição são baseadas na estrutura de repetição anterior para evitar que o programa retorne triplas repetidas.

Análise de Complexidade do algoritmo proposto:

3 SUM - FORÇA BRUTA	
Linha	Custo
10	0
11	1
12	0
13	$2n - 2$
14	$n - 2$
15	0
16	$2n^2 - 5n + 2$
17	$n^2 - 4n + 4$
18	0
19	$2n^3 - 10n^2 + 16n - 8$
20	$n^3 - 6n^2 + 12n - 8$
21	0
22	$n^3 - 6n^2 + 12n - 8$
23	$n^3 - 6n^2 + 12n - 8$
24	$n^3 - 6n^2 + 12n - 8$
25	0
26	0
27	0
28	0
29	0
30	0
31	1
32	0
33	0
T(n)	$6n^3 - 31n^2 + 58n - 36$
$T(n) \in \Theta$	$\Theta(n^3)$

3SUM - Melhorado

Imagem 2: Função “treeSumMelhorado()” utilizada:

```
36  /* treeSumMelhorado(): */
37  void treeSumMelhorado(int A[], int n) {
38
39      int numeroTripla=0;
40
41      for(int i=0; i<n-1; i++){
42          qtdOperacoes3SumMelhorado++;
43
44          for(int j=i+1; j<n; j++){
45              qtdOperacoes3SumMelhorado++;
46
47              int numeroPreciso = 0 - A[i] - A[j];
48              int posicaoNumero = BuscaBinaria(numeroPreciso, A, j+1, n-1);
49
50              if(posicaoNumero != -1){
51                  numeroTripla++;
52                  printf( " \n%d Tripla Encontrada: [%d, %d, %d]", numeroTripla, A[i], A[j], A[posicaoNumero]);
53              }
54
55          }
56      }
57
58  }
```

Explicação do algoritmo proposto:

O algoritmo se baseia na criação de dois “for’s”, que percorrerão todo o array servindo como os 2 primeiros elementos das triplas. Com estes 2 primeiros elementos, o algoritmo procura pelo terceiro elemento em todo o array, a partir da função “BuscaBinaria()”, que buscara pelo terceiro elemento apenas na parte do array à frente da parte já percorrida pelas estruturas “for”.

Análise de Complexidade do algoritmo proposto:

3 SUM - MELHORADO	
Linha	Custo
38	0
39	1
40	0
41	2n
42	n-1
43	0
44	$2n^2 - 2n$
45	$n^2 - 2n + 1$
46	0
47	$n^2 - 2n + 1$
48	$\lg n * (n^2 - 2n + 1)$
49	0
50	$n^2 - 2n + 1$
51	$n^2 - 2n + 1$
52	$n^2 - 2n + 1$
53	0
54	0
55	0
56	0
57	0
58	0
T(n)	$\lg n * (n^2 - 2n + 1) + 7n^2 - 9n + 5$
$T(n) \in \Theta$	$\Theta (n^2 \lg n)$