Contando Casas – Algoritmo de Soma Consecutiva

Carlos Leandro Silva Machado

Universidade do Vale do Rio dos Sinos (Unisinos)
São Leopoldo – RS – Brasil

Abstract. *This meta-article demonstrates the solutions found to find the consecutive sum of the numbers and finding pairs with equal previous and posterior summation, in addition it will be demonstrated by means of pseudocode and a brief description on the algorithms with time comparisons and the number of pairs found for each solution, finally, a brief conclusion about this project will be described.*

Resumo. *Este meta-artigo demonstra as soluções encontradas para descobrir a soma consecutiva dos números e encontrando pares com somatório anterior e posterior iguais, além disso será demonstrado por meio de pseudocódigo e uma breve descrição sobre os algoritmos com comparações de tempo e o número de pares encontrados para cada solução, por fim, será descrito uma breve conclusão sobre este projeto.*

1. Introdução

O objetivo desse projeto é desenvolver um algoritmo capaz de gerar pares de valores inteiros, para gerar esses pares o algoritmo deve escolher um número inicial e após isso efetuar o somatório de todos os valores que antecedem o número escolhido, em seguida deve efetuar um somatório para os valores posteriores consecutivos ao número escolhido até que o somatório posterior seja igual ao anterior e assim gerando o par com o número escolhido e o último número do somatório posterior.

Esse processo deve ser feito para encontrar pelo menos os quinze primeiros pares e começará com a casa inicial seis.

2. Primeira Solução

O algoritmo da primeira solução foi construído da seguinte forma, é utilizado um laço para percorrer um valor inteiro muito grande e cada iteração do laço funciona como o primeiro valor do par (casa), o laço inicia com valor seis.

Para cada iteração é utilizado a fórmula do somatório de *Gauss* para somar todos os valores que antecedem a casa atual.

Em seguida é utilizado um segundo laço que soma os números posteriores a casa atual, o laço é executado enquanto o somatório posterior for inferior ao somatório anterior, caso o somatório posterior seja superior ao somatório anterior o laço é interrompido sem retorno.

Caso o somatório posterior seja igual ao somatório anterior então terá como saída os pares representados pela casa numérica atual do primeiro laço e o ultimo valor acrescentado ao somatório posterior.

```
1.  funcao geradorDePares
2.  max <- 1000000000000
3.  para casa de 6 até max faça
4.      soma_anterior <- ((casa-1) * casa) /2
5.      ultimo_par <- casa + 1
6.      soma_proxima <- casa + 1
7.      enquanto soma_proxima menor que soma_anterior então
8.          ultimo_par <- ultimo_par + 1
9.          soma_proxima <- soma_proxima + ultimo_par
10.         se soma_proxima igual a soma_anterior então
11.             escreva os pares casa e ultimo_par com um espaçamento de 20 linhas a direita
12.             interromper laço
13.         fim se
14.     fim enquanto
15. fim para
```

Figura 1 – Pseudocódigo da primeira solução.

Devido ao longo tempo de processamento do algoritmo, o mesmo foi interrompido com aproximadamente 222 segundos de execução resultando em oito saídas.

3. Segunda Solução

Assim como o algoritmo anterior, ele percorre um laço iniciado com o valor seis até um número inteiro muito grande.

A cada iteração do laço é utilizado o somatório de Gauss para calcular o somatório anterior à casa atual, em seguida é utilizado uma equação quadrática afim de encontrar o par final que fará o somatório anterior ser igual ao somatório posterior.

Em seguida é utilizado uma variação do somatório de Gauss para encontrar o somatório posterior partindo do próximo número após a casa atual e o valor encontrado na equação quadrática anterior.

Caso o somatório anterior seja igual ao somatório posterior terá como saída os pares representados pela casa numérica atual e o valor encontrado na primeira equação quadrática.

```
1.  funcao geradorDePares
2.  max <- 1000000000000
3.  casa <- 6
4.  enquanto casa menor que max faça
5.      soma_anterior <- ((casa-1) *casa) /2
6.      i <- casa + 1
7.      delta <- 1 - (4 * (-((soma_anterior * 2) + (i*(i - 1)))))
8.      x <- ((-1) + (raiz quadrada de delta)) / 2
9.      soma_proxima <- (((x*(x + 1)) - (i*(i - 1))) /2)
10.     se soma_proxima igual a soma_anterior então
11.         escreva os pares casa e x com um espaçamento de 20 linhas a direita
12.     fim se
13. fim enquanto
```

Figura 2 – Pseudocódigo da segunda solução.

Devido ao longo tempo de processamento do algoritmo, o mesmo foi interrompido com aproximadamente 294 segundos de execução resultando em onze saídas.

4. Terceira Solução

Essa solução é no seu total quase idêntica a solução anterior, mas há uma pequena alteração que faz uma grande diferença no resultado final.

Foi verificado que após multiplicar o segundo número do par pelo valor quatro, essa soma tende a se aproximar do valor da próxima casa com um somatório anterior igual ao somatório posterior.

Sendo assim, ao adicionar mais casas decimais ao valor quatro a aproximação é muito maior, sendo assim foi incluído no algoritmo que após o somatório anterior ser igual ao somatório posterior terá como saída os pares representados pela casa numérica atual e o valor encontrado na primeira equação quadrática e em seguida será definido como o número da próxima casa a ser testada o segundo número do par multiplicado pelo valor constante 4,1213203, onde com essa constante foi possível um tempo quase constante para encontrar quinze pares.

```
1.  funcao geradorDePares
2.  max <- 1000000000000
3.  casa <- 6
4.  enquanto casa menor que max faça
5.      soma_anterior <- ((casa-1) *casa) /2
6.      i <- casa + 1
7.      delta <- 1 - (4 * (-((soma_anterior * 2) + (i*(i - 1)))))
8.      x <- ((-1) + (raiz quadrada de delta)) / 2
9.      soma_proxima <- (((x*(x + 1)) - (i*(i - 1))) /2)
10.     se soma_proxima igual a soma_anterior então
11.         escreva os pares casa e x com um espaçamento de 20 linhas a direita
12.         casa <- x * 4.1213203
13.     senão
14.         casa <- casa + 1
15.     fim se
16. fim enquanto
```

Figura 3 – Pseudocódigo da terceira solução.

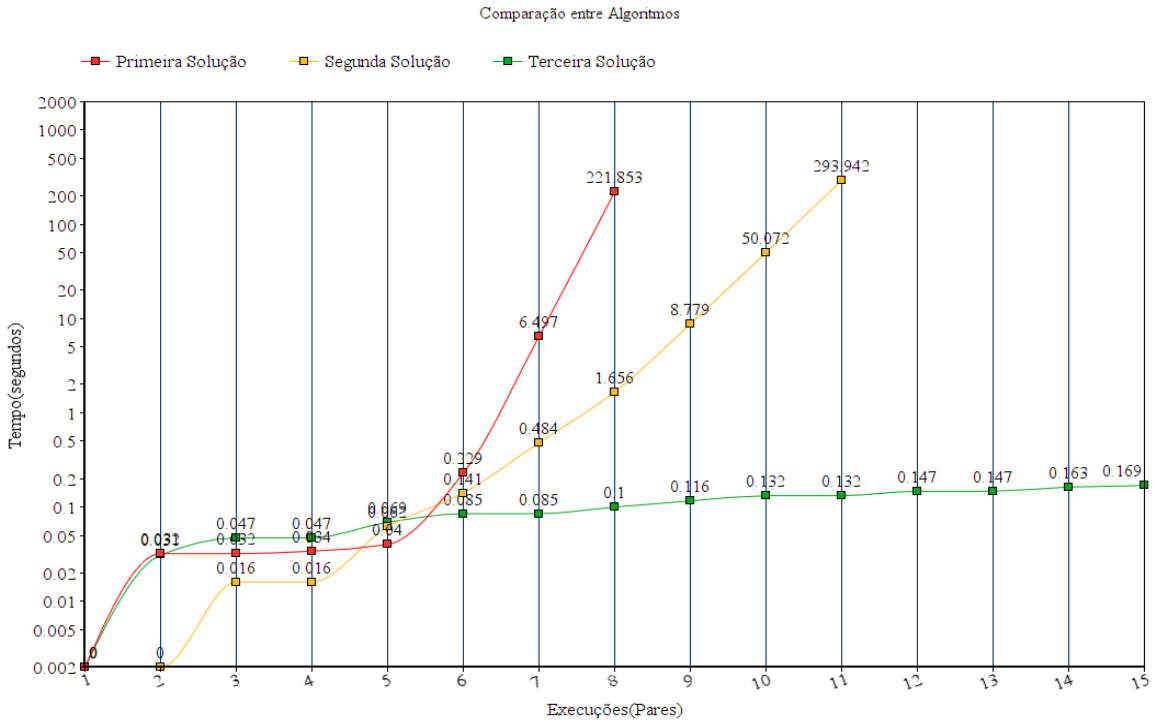
Por fim, essa última solução encontrou mais de quinze pares conforme a imagem abaixo demonstra.

1	6	8
2	35	49
3	204	288
4	1189	1681
5	6930	9800
6	40391	57121
7	235416	332928
8	1372105	1940449
9	7997214	11309768
10	46611179	65918161
11	271669860	384199200
12	1583407981	2239277041
13	9228778026	13051463048
14	53789260175	76069501249
15	313506783024	443365544448
16	1827251437969	2584123765441
17	10650001844790	15061377048200
18	62072759630771	87784138523761
19	361786555939836	511643454094368
20	2108646576008245	2982076586042449

Figura 4 – Total de pares encontrados

5. Conclusão

Nas soluções apresentadas acima podemos ver que o maior problema para gerar os pares é o tempo, abaixo vemos um gráfico comparando o tempo de execução dos algoritmos para cada iteração dos pares.



Podemos concluir que apesar de ser um algoritmo que gera valores muito grandes não chega a ser complexo, obviamente isso varia para cada tipo de implementação, além disso, esse trabalho teve uma importância significativa pois em meio a comparações foi possível diferenciar os níveis de complexidade dos algoritmos desenvolvidos.