**Arthur Pires da Fonseca nºUSP: 10773096**

**Relatório sobre o EP1 e a Conjectura de Collatz**

A partir da fórmula dada no enunciado do EP, sabe-se que o programa em questão deve contabilizar o número de passos que são dados até chegar ao 1 a partir de um número inicial dado contido na sequência referida pelo usuário do *software.* Seguindo a regra de transformar o número dado em metade do seu valor se aquele for par, e no sucessor do triplo dele se o mesmo for ímpar.

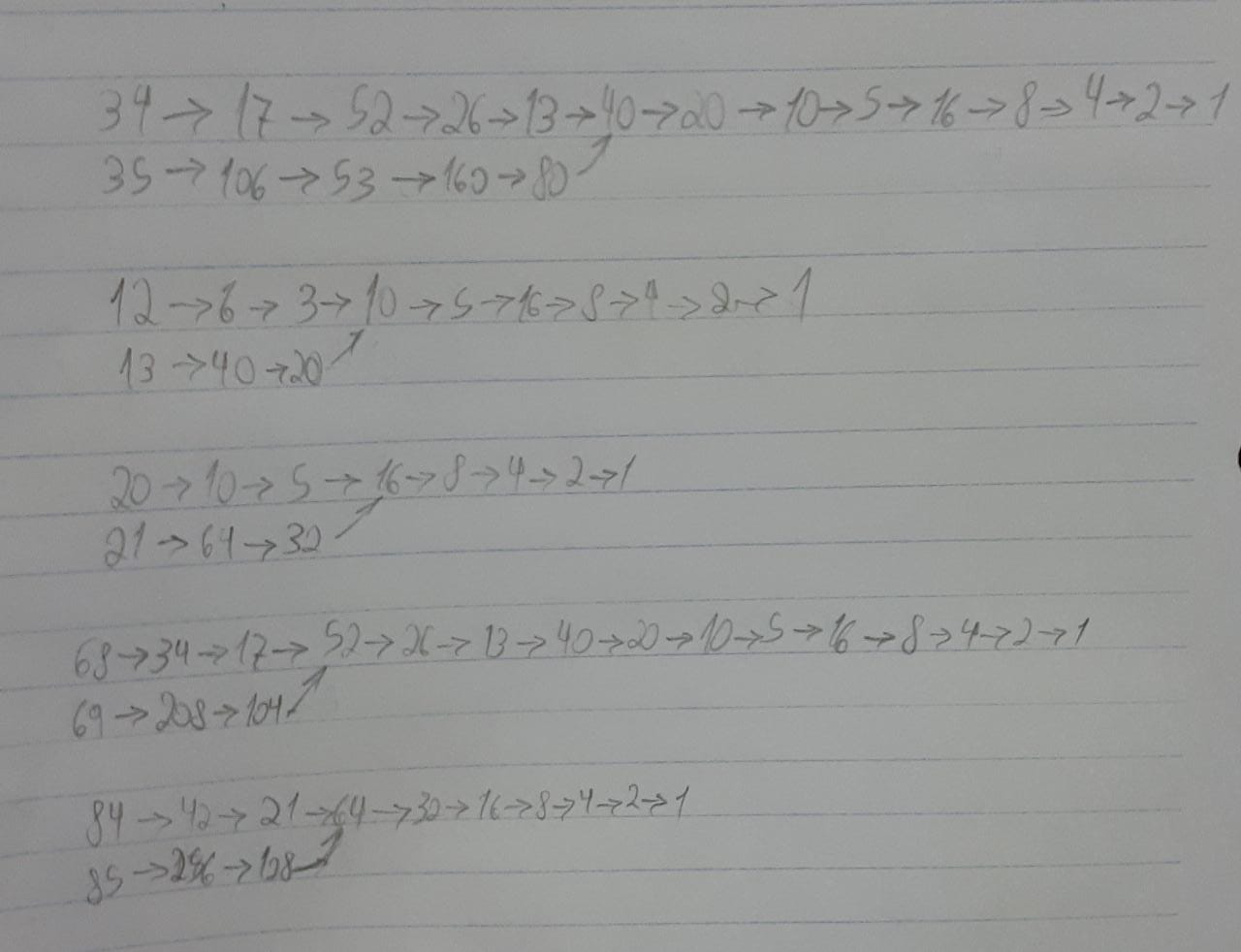
Pude observar, inicialmente, a partir da Aritmética Modular, que para todo número inteiro ímpar n, 3n+1 é congruente a 0 (mod 2), ou seja , 3n+1 é par quando n for ímpar.

A partir dessa observação, implementei a primeira otimização no programa, a qual contabiliza 2 passos de uma vez sempre quando ele se depara com um número ímpar durante as transformações da função de Collatz.

Isso é feito da seguinte forma: no momento em que o valor da variável *num* da função *int Collatz* torna-se ímpar (isso é identificado pela condição *num % 2 == 1*), a função converte essa variável para *(num \* 3 + 1)/2* e adiciona 2 passos à variável *passos* (inicializada no começo da função com o valor 0). Essa primeira melhoria reduz as 2 operações de transformação da variável *num*  em 1 só quando ela é ímpar. Com esse aprimoramento, o programa aumentou muito seu desempenho.

Após esse passo, executei o programa de 1 a 1000 e percebi alguns padrões pontuais, como alguns números vizinhos possuindo a mesma quantidade de passos para chegar ao 1.

Em seguida, recorri ao papel para analisar os estados desses números e retificar se era apenas coincidência.

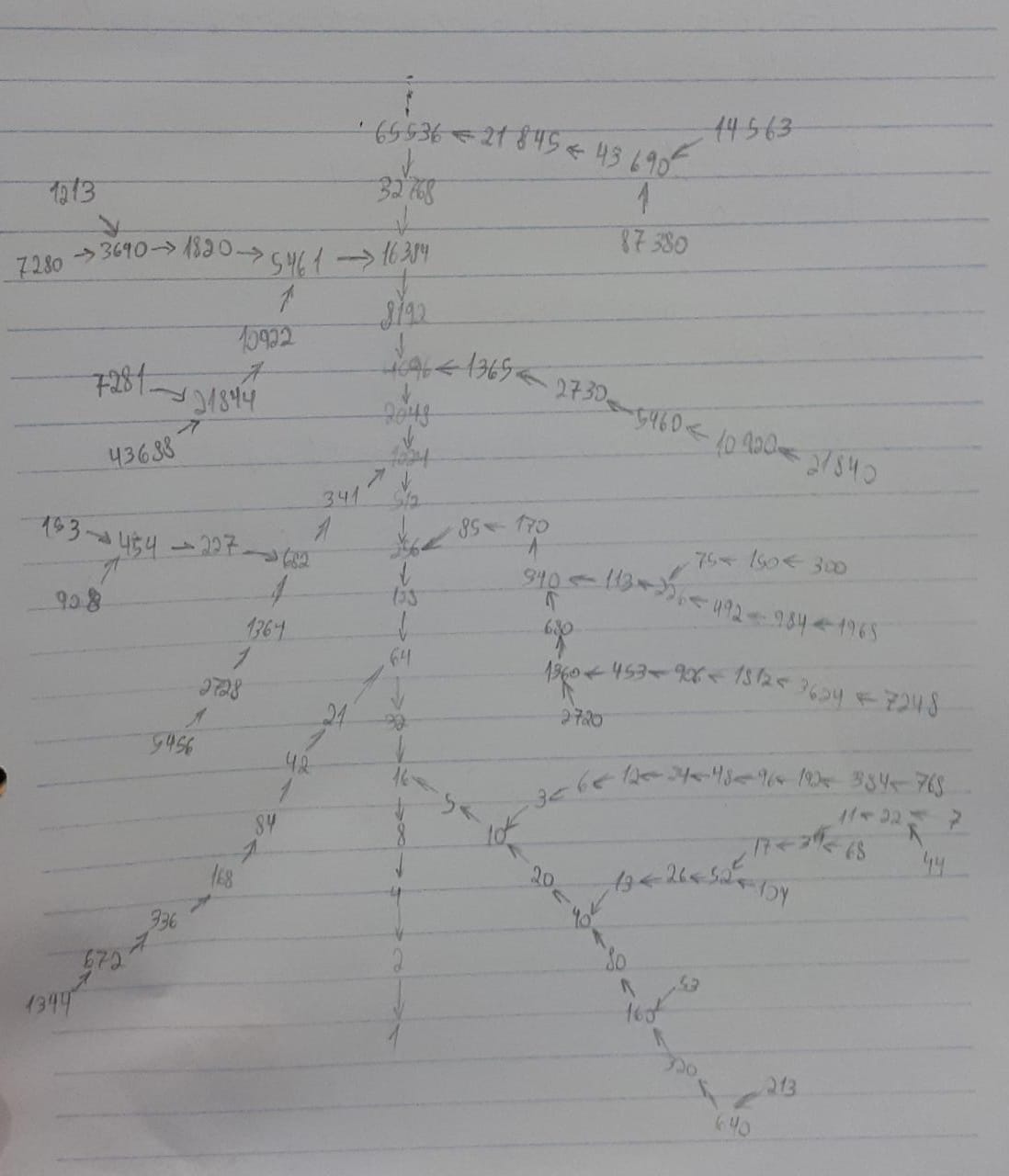


Percebi que os números por que se passa quando partimos do par 34 e 35 se distanciam e se aproximam a intervalos regulares, mas isso não é constante, porque há um momento em que convergimos para o número 40, de onde os passos seguintes são exatamente os mesmos. Conclui que existem números consecutivos que convergem para um mesmo ponto, e pensei que isso poderia ser útil para determinarmos mais rapidamente a quantidade de passos do número seguinte na sequência quando o ponto de convergência puder ser determinado.

A partir desse raciocínio, decidi analisar a conjectura “de trás para frente”, não pensando mais no f(n), mas sim no(s) números que dá(ão) origem ao f(n). Pensando assim, obtive duas observações:

- Todo número ímpar só pode vir de uma interação com um par dividido por 2.

- Todo número par vem de outro par, mas alguns também vêm de um ímpar, e essa última situação acontece apenas quando o par que se observa for um sucessor de um múltiplo de 3.



Construi, então, uma árvore numérica, ideia a qual obtive após assistir a um vídeo no *YouTube* sobre o assunto.**[ 1]**

Percebi que todo número que converge para o 1 converge necessariamente para o 16 também (desconsiderando os números com respostas triviais, isto é, as potências 2 menores que 16).

Adicionei, então, um verificador na função *Collatz* para parar a contagem caso o número convergisse para 8 após passar pelo verificador-modificador de números ímpares (pois o 16 tem origem no número 5, e é modificado para 8 logo após tornar-se 16 dentro da função *Collatz*). Junto a isso, inicializei a variável passos com o valor 3, pré-contabilizando os últimos 3 passos de todos os números que não 1,2 e 4.

Adjacente a esse verificador, adicionei um *while* para verificar números pares sempre que aparecerem, modificando-os até que se tornem ímpares ou iguais a 8, situação em que a função para a contagem.

Voltando a observar os casos de números consecutivos que compartilham da mesma quantidade de passos, percebi que poderia valer a pena armazenar em um vetor os passos relacionados a cada número por que passa a função partindo do número antecessor.

Modifiquei, então, o código, para poder implementar esse novo mecanismo. Com isso, fui obrigado a remover a implementação inicial, que contava automaticamente 2 passos quando a função *Collatz* passava por um número ímpar.

Não fui capaz de testar a eficiência do algoritmo, pois não tenho conhecimento para tal, mas a saída do programa mostrou-se correta para a maior parte das entradas.

**Referências**

**1 - https://www.youtube.com/watch?v=5mFpVDpKX70&t=26s**