Juliano Garcia de Oliveira Nº USP: 9277086

Este relatório descreve o funcionamento do algoritmo implementado para o EP1 da disciplina de MAC121, os conceitos matemáticos utilizados para simplificação, redução no tempo de processamento, espaço alocado, etc.

- **Conceitos Matemáticos**

A única otimização que se baseia em conceitos puramente matemáticos é quando o número **n** for ímpar, efetuar a operação:

$$n = \frac{3*n+1}{2}$$

e incrementar o contador em 2 passos, já que um número ímpar sempre se torna par na sequência de Collatz (exceto o 1).

- **Algoritmo(s) e otimizações**

A versão final do algoritmo se baseia em três pontos principais para reduzir o tempo e o espaço utilizado para o processamento:

1. **Programação Dinâmica**

Sejam **i** e **j** os números do intervalo especificado na entrada. Se for possível, o programa aloca as posições de um vetor para armazenar o número de passos da seq. De Collatz de cada número, começando a partir do **i**.

Caso em algum número **n** tal que **n > i** aparecer algum número entre **i** e **n** no cálculo da sequência de Collatz, o algoritmo não irá calcular novamente o número, mas irá somar a quantidade de passos que já foi calculada para **n** com o valor armazenado no vetor (que já foi calculado anteriormente).

Por exemplo, **i = 4** e **j = 7**. A sequência de Collatz para **n = 4** é: 4 → 2 → 1 . (2 passos)

Para **n = 5**: 5 → 16 → 8 → 4 → 2 → 1 (5 passos).

Usando a programação dinâmica, ao invés de calcular novamente a sequência de Collatz do número 4, o programa simplesmente retorna o que já foi calculado, aumentando consideravelmente o tempo de execução do programa.

2. Limite para alocação de memória

A versão do programa mais rápida calculava a sequência de Collatz de 1 até 2147483647 (**INT MAX**) em aproximadamente 44 segundos. Mas para isso, usava mais de 4 GB de RAM na Programação Dinâmica.

Para diminuir o espaço utilizado, utiliza-se então uma constante ('MAX_ALLOC') que define o maior espaço que vai ser alocado pelo programa. A constante pode ser modificada no código fonte, e está pré definida com um limite um pouco menor mas que permite um tempo médio razoavelmente rápido, e não usa tanto espaço quanto antes. No momento, o programa gasta no máximo ~580 MB de RAM, mas isso pode ser reduzido ou aumentado apenas modificando o valor da constante 'MAX_ALLOC' (*como **não foi** especificado um valor ideal de consumo de espaço, julguei que o valor escolhido é bastante razoável para o processamento MÁXIMO).*

3. Tabela da sequência

Inicialmente, a Programação Dinâmica só iria salvar os números de **i** até **j**. Isso gera um problema em que casos com **i** e **j** muito altos (mesmo que próximos entre si) demoravam mais para processar do que casos cuja amplitude de intervalo (**j − i**) era maior.

Um exemplo: O programava rodava mais rápido para calcular de 1 até **INT MAX** do que de 1000000000 (Um bilhão) até **INT MAX** (valor máximo de INT). Isso se deve ao fato que vários números na sequência de Collatz chegam em números menores (i.e. abaixo de um milhão, por exemplo) do que números da ordem de bilhões.

A solução para otimizar esse problema é, no início do programa, calcular todos os passos na seq. de Collatz de 1 até um número pré definido. No programa implementado utiliza-se a constante 'DESVIO' para definir qual o é este número **k**, sendo que no início do programa é calculado todos os passos da seq. de Collatz de **1 até k**.

O padrão é calcular até 10 milhões, já que é um intervalo praticamente instantâneo de se calcular e diminui razoavelmente o processamento nos outros casos.

- Observações interessantes

Para se escolher o tamanho dos números, foram feitas várias observações e testes para evitar *overflow* ou gasto desnecessário de tamanho.

Primeiramente, verificou-se através de uma simples modificação no algoritmo que, entre 1 e INT MAX, o número de passos não passava de 1024. Logo o vetor da Programação Dinâmica poderia ter simplesmente tamanho de 10 bits. Testou-se a utilização de um *bit field* para 10 bits, mas como o C ANSI emite um warning ao usar *bit fields*, optou-se por utilizar o tipo *unsigned short int*.

Em números muito altos, no cálculo da seq. de Collatz havia *overflow* de INT, portanto para esse cálculo se usa o tipo *unsigned long int*.

- Outras informações e resumo do algoritmo

O maior intervalo calculado foi o maior possível de acordo com as especificações do EP, isto é, de 1 até INT MAX.

O algoritmo implementado cria um array dinâmico e um array de tamanho 'DESVIO'.

A tabela pré definida são os valores gerados automaticamente no início do programa, e ficam armazenados no array '*numbersCalc*'. O array '*dynamicAloc*' contém os outros valores armazenados no programa.

Ou seja, dentro da função de cálculo da sequência de Collatz para um determinado número n em uma iteração qualquer, temos as seguintes situações:

- n < 'DESVIO' : É só retornar o número de passos somado ao que já foi calculado na tabela pré definida;
- 'DESVIO' < n < 'MAX_ALLOC' : Retorna o número alocado dinâmicamente (se já tiver sido calculado);
- n > 'MAX_ALLOC': Neste caso o número n continua no *loop* da função até cair em um dos dois casos acima.

- Referências utilizadas

https://en.wikipedia.org/wiki/Collatz_conjecture

http://blog.adnanmasood.com/2012/02/23/optimizing-collatz-sequence-with-dynamic-programming/

https://pythonandr.com/2015/09/01/collatz-conjecture-what-you-need-to-know/