UNIVERSIDADE FEDERAL DE MINAS GERAIS

INSTITUTO DE CIÊNCIAS EXATAS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

BRUNO ANTONIO VIEIRA

TRABALHO PRÁTICO 0 DE AEDS 3

BELO HORIZONTE

2017

SUMÁRIO

1 INTRODUÇÃO

Nesse trabalho prático é apresentado um problema que requer a solução dos possíveis operadores de uma expressão matemática em notação polonesa reversa. Para isso, deve ser introduzida uma entrada que contenha a expressão sem os operadores de soma ou multiplicação, os quais devem ser descobertos, e no fim o algoritmo deve retornar as possíveis combinações de operadores que tornam a expressão matemática verdadeira.

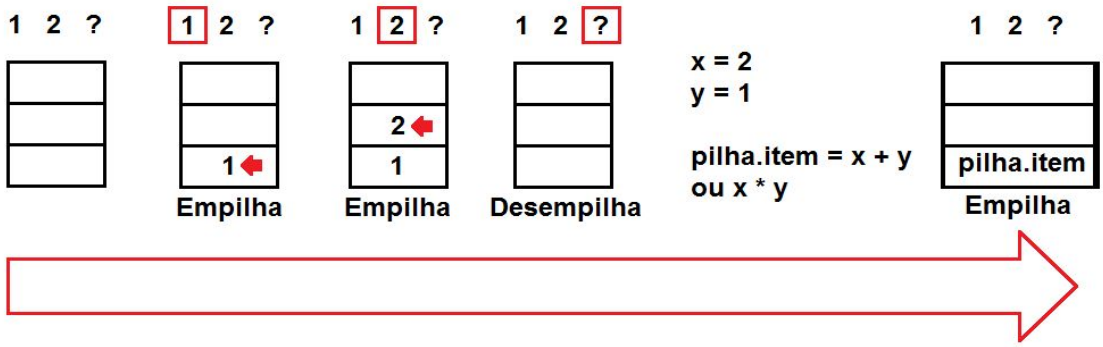
A solução apresentada faz uso de uma pilha dinâmica, que armazena os números e os desempilha quando é necessário realizar uma operação de teste, e de uma função que transforma o valor da iteração atual em um número binário, para que todas as possibilidades sejam avaliadas. Mais detalhes sobre a implementação da solução serão descritas no tópico seguinte.

2 SOLUÇÃO DO PROBLEMA

Inicialmente, uma leitura dos valores de entrada é feita e armazenada numa string, para posteriormente esses valores serem convertidos num vetor de inteiros que possibilita os cálculos e comparações.

Um laço é executado 2 elevado a n vezes, onde n é a quantidade de operadores, que podem ser de soma (+) ou de multiplicação (*), existentes no vetor de entrada. Esse é o número de possibilidade de combinações que a expressão matemática pode ter para chegar ao resultado final.

A cada iteração do laço “for” o algoritmo avalia se o conteúdo da posição atual do vetor de entrada é um número natural positivo, que faz parte da expressão, ou um operador, representado pelo valor -1. Se o conteúdo for um número positivo ele é armazenado em uma pilha. Caso seja um operador, os dois últimos números armazenados na pilha são desempilhados e submetidos a uma operação de soma ou multiplicação. O resultado parcial é então novamente colocado na pilha.



Para saber quais ordens de operadores devem ser testadas, o algoritmo transforma o valor da iteração atual de decimal para binário, com isso, um bit 0 (zero) representa o operador de soma (+) e um bit 1 (um) representa um

operador de multiplicação (*). Conforme o laço “for” vai sendo executado, os bits da string binária vão variando, até que todas as possibilidades tenham sido testadas.

Exemplo para uma entrada que possua 3 possíveis operadores:

O número de iterações do laço “for” será 2 elevado a 3 que é igual a 8:

- Primeira iteração, com i = 0

0	0	0

  serão testados os operadores: + + +
- Segunda iteração, com i = 1

0	0	1

  serão testados os operadores: + + *
- Terceira iteração, com i = 2

0	1	0

  serão testados os operadores: + * +
- Quarta iteração, com i = 3

0	1	1

  serão testados os operadores: + * *
- Quinta iteração, com i = 4

1	0	0

  serão testados os operadores: * + +
- Sexta iteração, com i = 5

1	0	1

  serão testados os operadores: * + *
- Sétima iteração, com i = 6

1	1	0

  serão testados os operadores: * * +
- Oitava iteração, com i = 7

1	1	1

  serão testados os operadores: * * *

Por fim, o algoritmo verifica se o resultado armazenado dentro da pilha é igual ao resultado esperado da expressão dado na entrada. Em caso positivo, é feita a impressão da combinação de operadores utilizada e o algoritmo continua rodando até que todos os casos tenham sido testados e os ideais tenham sido impressos ao fim do programa.

3 ANÁLISE DE COMPLEXIDADE

- Estrutura de dados Pilha:

  A complexidade de tempo da pilha dinâmica tanto na função de Empilhar quanto Desempilhar é O(1), pois não é necessário percorrer a pilha, apenas coloca-se o item na posição mais superficial ou retira-se da posição mais superficial sem a necessidade de nenhuma comparação.

  A complexidade de espaço do algoritmo é O(n), visto que, para cada novo elemento inserido na pilha é necessário alocar uma nova quantidade de memória.

- Função de conversão em binário:

  A complexidade de tempo do algoritmo é O(n), onde n é o tamanho do vetor que será passado como argumento à função. Após a conversão de decimal para binário que consiste em divisões sucessivas por 2, o vetor de tamanho n recebe n inserções.

  A complexidade de espaço também é O(n), já que são alocados n espaços para compor um vetor que armazena os bits da string binária.

- Algoritmo principal:

  A rotina principal do algoritmo tem um loop que é executado 2^m vezes para testar todas as possibilidades para resolução do problema, onde m é a quantidade de operadores. Dentro desse loop há outro loop que é executado n vezes, onde n é a quantidade de números presentes na expressão matemática de entrada e n > m. Com isso, a complexidade temporal do algoritmo é O(n 2^m). Em todos os casos (melhor e pior) a complexidade mantém-se a mesma.

  A complexidade espacial do algoritmo também é O(n 2^m), devido às operações de empilhar feitas a cada iteração que alocam um espaço na memória RAM.

4 TESTES

Os testes foram realizados no sistema operacional Ubuntu 17.04 rodando num processador Intel Core i3 de 3.10 Ghz e 6 GB de memória RAM. Para a medição do tempo de execução do algoritmo foi implementado um trecho de código no programa que retorna o tempo em segundos.

Teste 1: 7 operadores

Entrada:
2 2 ? 1 1 1 ? ? ? 1 1 ? 2 ? ?
8
Tempo: 0.000348 segundos

Teste 2: 10 operadores

Entrada:
1 1 ? 1 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ?
2
Tempo: 0.002481 segundos

Teste 3: 15 operadores

Entrada:
1 1 ? 1 1 ? 1 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ?
4
Tempo: 0.100052 segundos

Teste 4: 20 operadores

Entrada:
1 1 ? 1 1 ? 1 1 ? 1 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ?
1 ?
6
Tempo: 2.72854 segundos

Teste 5: 25 operadores

Entrada:
1 1 ? 1 1 ? 1 1 ? 1 1 ? 1 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1
? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ?
8
Tempo: 1:43 minutos

Teste 6: 30 operadores

Entrada:
1 1 ? 1 1 ? 1 1 ? 1 1 ? 1 1 ? 1 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ? 1 ?
10
Tempo: Após 6 minutos de execução o programa parou de responder e travou todo o sistema operacional, não chegando à finalização.

5 CONCLUSÃO

É possível concluir a partir dos testes realizados com o algoritmo que ele é capaz de processar entradas menores com facilidade, mas ao elevar o tamanho das entradas (para n > 25, por exemplo), o tempo de execução torna-se extremamente alto e inviável, característica de um algoritmo que possui complexidade exponencial e que resolve problemas utilizando força bruta.