Universidade de Brasília Faculdade UnB Gama

Estruturas de Dados 1

Trabalho 2

Aplicação de pilhas: notação polonesa reversa

Prof. John Lenon C. Gardenghi

26 de junho de 2019

1 Introdução

Neste trabalho, estamos interessados em avaliar uma expressão matemática qualquer que contenha

- variáveis representadas por letras (maiúsculas) de A a Z,
- operações aritméticas de adição (+), subtração (−), divisão (/), multiplicação (∗) e potenciação (∧) e
- alteração da prioridade de avaliação usando parênteses.

O trabalho encontra-se no juiz eletrônico CD-MOJ[1] e consiste de três itens:

1. Verificar se uma expressão composta por chaves, colchetes e parênteses está corretamente parentizada.
2. Converter uma expressão da notação infixa para notação pós-fixa (polonesa reversa).
3. Avaliar uma expressão matemática.

São itens complementares: o segundo depende do primeiro e o terceiro, dos dois primeiros.

2 Expressão parentizada

Considere uma expressão matemática qualquer composta por chaves, colchetes e parêntese. Dizemos que uma expressão está **corretamente parentizada** se, para cada chave, colchete ou parêntese aberto, o correspondente é fechado corretamente. Por exemplo,

```
A+B/{C+[D-E*(F+G)]}
```

é uma expressão corretamente parentizada, mas

```
A+B/{C+[D-E)*(F+G)}
```

não é.

Note que uma expressão está corretamente parentizada se o operador que "fecha" é correspondente ao *último* que abriu. Deste modo, usamos uma pilha para verificar se a expressão está corretamente parentizada. O pseudoalgoritmo a seguir descreve o procedimento que retorna 1 se a expressão estiver corretamente parentizada e 0, caso contrário.

[1] `https://moj.naquadah.com.br/cgi-bin/contest.sh/jl_eda1c_t2_2019_1`.

Algoritmo 1: Verificando se uma expressão está corretamente parentizada

```
Entrada: Uma expressão matemática exp[1..n].
para cada caracter c em exp[1..n] faça
    se c = '(' ou c = '[' ou c = '{' então
        empilha(c)
    senão
        se c = ')' e desempilha() ≠ '(' então retorna 0
        se c = ']' e desempilha() ≠ '[' então retorna 0
        se c = '}' e desempilha() ≠ '{' então retorna 0
    fim
fim
se pilha_vazia() então
    retorna 1
senão
    retorna 0
fim
```

3 Conversão de notação infixa para pós-fixa

Na notação infixa, os operadores são escritos entre os operandos, enquanto na notação pós-fixa, o operador é escrito depois dos operandos. Por exemplo,

```
(A+B)/C
```

está na notação infixa, enquanto

```
AB+C/
```

é a notação pós-fixa correspondente. Embora a notação pós-fixa pareça muito mais complicada, do ponto de vista computacional ela é muito mais fácil de ser avaliada que a notação infixa, pois, entre outros fatores,

- Dispensa o uso de parêntesis
- É uma forma mais sistemática, portanto simplifica a elaboração de procedimentos computacionais para avaliar a expressão.

  A notação pós-fixa é gerada a partir da infixa usando uma pilha. A ideia é que

- toda variável é imediatamente transcrita para a expressão pós-fixa,
- a pilha armazene operadores por ordem crescente de prioridade e
- operadores sejam desempilhados e transcritos para a expressão pós-fixa sempre que a prioridade for menor ou igual a do operador sendo lido da expressão infixa.

  Veja o pseudocódigo a seguir e os exemplos no final deste arquivo.

Algoritmo 2: Conversão da notação infixa para pós-fixa.

Entrada: Uma expressão matemática `exp[1..n]`

```
para cada caracter c em exp[1..n] faça
    se c estiver entre A e Z então
        posfixa[j] ← c
        j ← j + 1
    senão
        se c = '(' então
            empilha(c)
        senão
            se c = ')' então
                op ← desempilha()
                enquanto op ≠ '(' faça
                    posfixa[j] ← op
                    j ← j + 1
                    op ← desempilha()
                fim
            senão
                t ← desempilha()
                se prioridade(c) > prioridade(t) então
                    empilha(t)
                    empilha(c)
                senão
                    enquanto prioridade(c) ≤ prioridade(t) faça
                        posfixa[j] ← t
                        j ← j + 1
                        t ← desempilha()
                    fim
                    empilha(t)
                    empilha(c)
                fim
            fim
        fim
    fim
fim
enquanto pilha_vazia() = false faça
    posfixa[j] ← desempilha()
    j ← j + 1
fim
```

Importante: No pseudocódigo acima, não contemplamos um caso especial: a potenciação. Diferentemente das demais operações, a prioridade da potênciação é ser feita da **direita para esquerda**. Por exemplo, enquanto a expressão

```
A*B*C
```

é traduzida para pós-fixa como

```
AB*C*
```

já que a multiplicação é avaliada da esquerda para a direita, a expressão

```
A^B^C
```

é traduzida para pós-fixa como

```
ABC^^
```

pois a potenciação é avaliada da direita para a esquerda.

4 Avaliação de uma expressão

A avaliação de uma expressão também se dá pelo auxílio de uma pilha usando uma expressão na forma pós-fixada. A ideia é que

- Valores das variáveis sejam empilhados.
- Quando um operador é lido, dois valores são desempilhados, e o resultado é empilhado.
- O resultado final será o último item restante da fila, após processar toda a expressão pós-fixa.

O algoritmo é exibido a seguir.

Algoritmo 3: Avaliação de uma expressão na notação pós-fixa.

Entrada: Uma expressão matemática `exp[1..n]` na notação pós-fixa e valores para todas as variáveis de `exp`.

para cada *caracter c em* `exp[1..n]` **faça**
  se *c estiver entre A e Z* **então**
    `empilha(c)`
  senão
    `n2 ← desempilha()`
    `n1 ← desempilha()`
    Faça a operação `n1 c n2` e empilhe o resultado
  fim
fim
`res ← desempilha()`

Sugestão: Tome exemplos e execute os algoritmos acima no papel antes de implementar, para ajudar no entendimento. Acompanhe os exemplos abaixo manuscritos também com o auxílio dos algoritmos, e refaça-os.

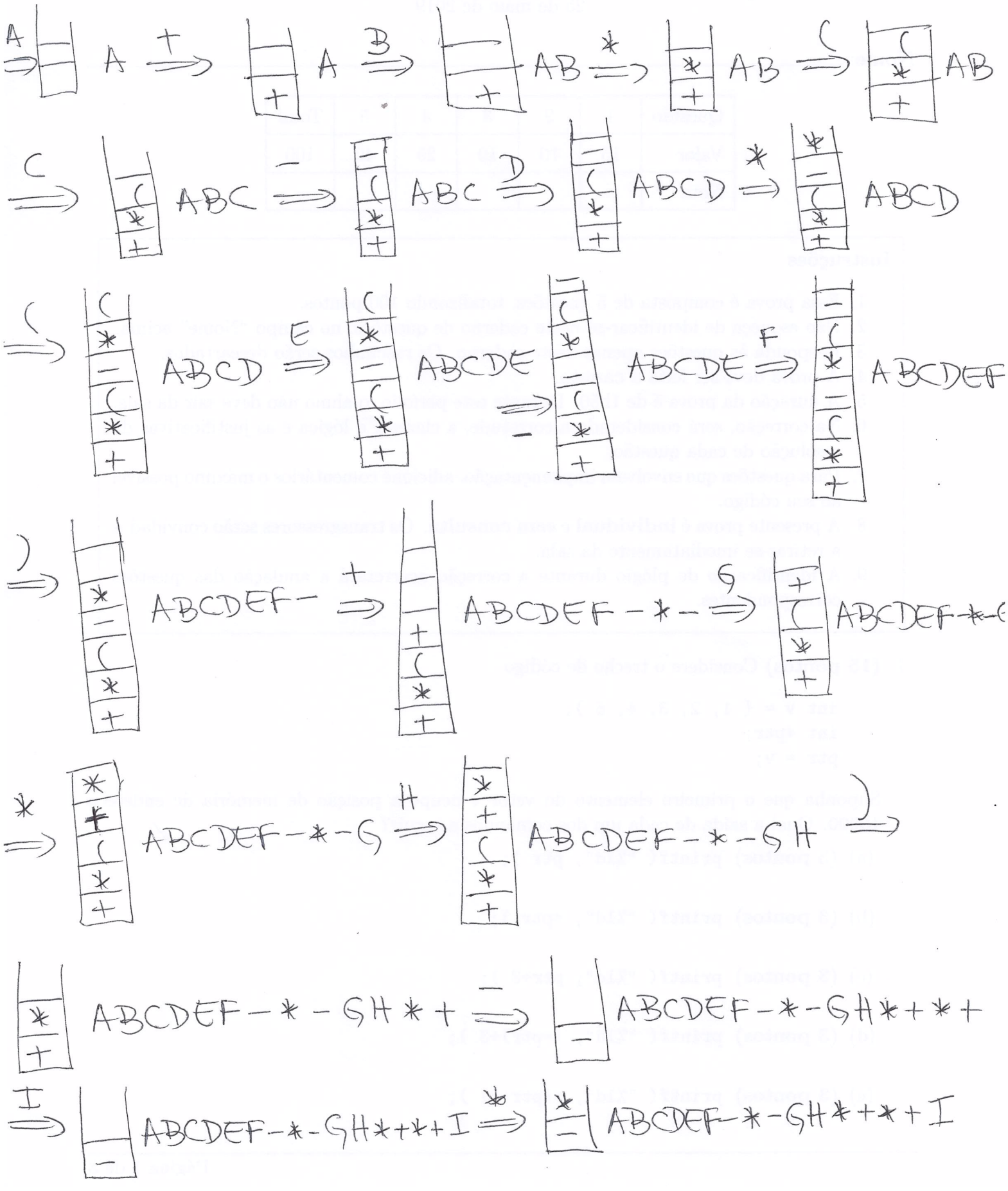
Bons estudos!

Prof. John Lenon C. Gardenghi
john.gardenghi@unb.br
Sala 22 - UED

Exemplo: infixa p/ posfixa

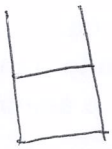
Expressão: A+B*(C-D*(E-F)+G*H)-I*K

Passo-a-passo: pilha e expressão pós-fixa



K ⇒ 

ABCDEF-*-GH*+*+IK

⇒  ABCDEF-*-GH*+*+IK*-

---

Exemplo: Avaliação de expressão

(A+B)/C

A=5

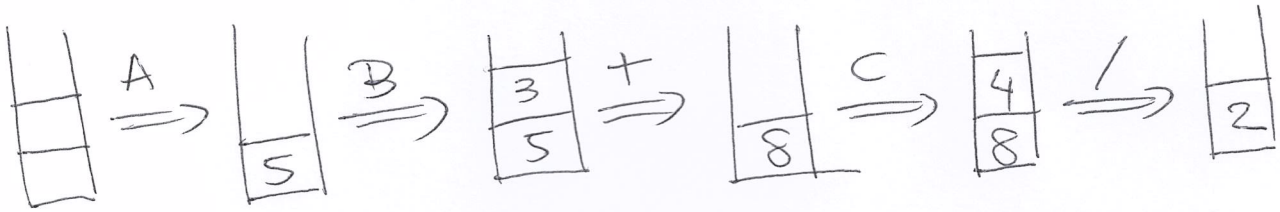
B=3

C=4

① Conversão infixa p/ pós-fixa

⇒ AB+C/

② Avaliação passo a passo: pilha



Resultado final: 2