Trabalho Prático 0: Notação Polonesa Reversa

Algoritmos e Estruturas de Dados III – 2017/1

Tatiana Santos Camelo de Araujo – 2015086298

1. Introdução

A notação polonesa inversa coloca o operador de uma operação após os valores associados, eliminando parênteses sem deixar ambiguidades.

O problema dado neste trabalho se trata de criar um algoritmo que ache possíveis operadores, apenas entre soma e multiplicação, de dada operação dada. O caractere dado pela operação é transformado em "?" no arquivo de entrada.

A saída é ordenada lexicograficamente e o caractere "+" tem prioridade em relação ao *, ou seja, havendo duas possibilidades de resposta como $+*$ e ++, a expressão ++ deve ser impressa primeiro no arquivo de saída.

2. Solução do problema

A ideia deste algoritmo é ler o vetor inicial, o resultado e salvá-los. Lendo o vetor, os valores diferentes de "?" são lidos e empilhados em outro vetor. Assim, se tivermos por exemplo "321?" na string original, o valor na pilha será "123". Essa implementação é ótima para este problema, pois garante que a operação, se iniciada no topo, será sempre feita com os dois últimos números que entraram na pilha.

A estrutura usada é a lógica de árvore, embora uma árvore não foi implementada. Quando o programa lê "?", ele para de empilhar os caracteres – até então são só números, na ordem que se dá a operação – e faz a primeira operação. O caractere de interrogação só pode representar adição ou multiplicação. Assim, uma função recursiva faz ambas operações. À direita são as operações de soma e à esquerda, as de multiplicação.

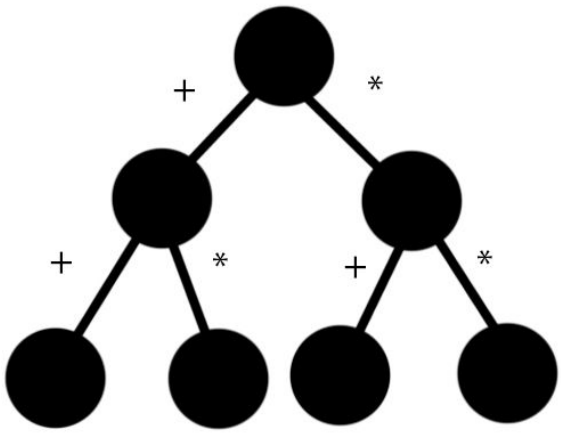

Imagem 1. Caminho da árvore em relação aos operadores.

Este implementação da chamada recursiva garante que, imprimindo da esquerda para direita, a ordem lexicográfica vai ser respeitada.

O arquivo "operacao" tem as funções usadas no arquivo main. imprimeArray é uma função que percorre uma string toda, imprimindo-a. copiaString copia os valores de uma string para outra. Na função resolve, se a string de valores for igual o valor final, imprime os operadores; copia a string original para uma nova; separa cada substring por caractere espaço; insere os números na string de valores; faz o cáculo para os dois números do início da string; reempilha o novo valor; recursivamente faz isso para todos os casos no caminho da árvore. Esta é a função mais complexa do programa e que resolve o problema proposto.

Como rodar:
A pasta compactada contém o aquivo makefile e é rodado com o comando "make run". São usadas as flags "-Wall -Wextra -Werror -std=c99 -pedantic".

3. Análise de complexidade

3.1 Funções

Nome da função	O que faz	Complexidade
imprimeArray	Função que percorre um array de 0 até o valor máximo (size) e imprime a cada leitura.	O(n)
copiaString	Função que copia uma string para outra.	O(n)
resolve	Resolve o problema. Se a string de valores for igual o valor final, imprime os operadores; copia a string original para uma nova; separa cada substring por espaço; insere os números na string de valores; faz o cáculo para os dois números do início da string; reempilha o novo valor; recursivamente faz isso para todos os casos. A complexidade desta função é devido a quantidade de vezes que a função recursiva é chamada. Ao final da árvore, serão n^2 nós, ou seja, o máximo de possibilidades que podem ser impressas é relativo a quantidade de operações na string original – ou seja, a quantidade de "?".	O(nˆ2) outras ações da função tem complexidade O(1)
main	Lê os valores de entrada (string original e resultado); conta quantas separações por espaço tem (cada número ou "?"); quanta quantos níveis tem na árvore;	O(n)+O(nˆ2)= O(nˆ2)

4. Avaliação experimental

O trabalho foi testado com todos os toys e teve a saída correta. Abaixo das imagens dos testes, com as entradas, saídas e o comando de execução.

```
[tatiana@zireael:~/Dropbox/UFMG/aeds3/tp0$ make run
./exec
1 5 1 ? 1 ? ?
7
++*
+*+
*++
[tatiana@zireael:~/Dropbox/UFMG/aeds3/tp0$ make run
./exec
4 4 ? 3 5 ? ? 1 ?
16
+++*
```