EP 02

Ronaldo Fumio Hashimoto

Data de Entrega: 02 de maio de 2010

BSTs

1 Introdução

Neste EP, você deverá implementar uma Árvore de Busca Binária (*Binary Search Tree* - BST). Com a BST implementada, você deverá escrever um programa que recebe como dados vários livros disponíveis no Projeto Gutenberg, digamos algumas centenas de livros, e que produz um *índice invertido* (veja esta entrada no *Wikipedia*) para que, posteriormente, um usuário possa rapidamente localizar todas as ocorrências de qualquer palavra nestes livros.

2 Tarefas

A sua tarefa tem três partes:

1. implementar uma BST;
2. implementar um índice invertido usando sua BST;
3. escrever um modo de interação com o usuário, através do qual ele poderá realizar buscas de palavras (como você faz no *google*).

3 Implementação da BST

A sua implementação de BST deve ter os seguintes arquivos:

- `Item.h`
- `BST.h`
- `BST.c`

Inclua adequadamente as macros condicionais `\#ifndef` para evitar definição duplicada de constantes e váriáveis globais. Além disso, sua implementação em `BST.c` **deve conter** pelo menos funções que executam as seguintes manipulações:

- Aloca um novo nó;
- Inicializa uma BST vazia;
- Busca um item na BST (implementada de forma não recursiva);
- Insere um item na BST (implementada de forma não recursiva).
- Remove um item da BST (implementada de forma não recursiva).

4 Índice Invertido

O usuário vai especificar um conjunto de arquivos texto, como por exemplo, livros do Projeto Gutenberg, `19033.txt` (*Alice's Adventures in Wonderland*, Lewis Carrol). Executando seu programa com a linha de comando

```
$ ./ep3 19033.txt
```

ou

```
$ ./ep3 -flista.txt
```

vemos dois casos possíveis. No primeiro caso, `19033.txt` será usado para montar o índice invertido. No segundo caso, supomos que `lista.txt` é um arquivo que contém o nome dos arquivos que devem ser lidos para se montar o índice invertido (por simplicidade, vamos supor que os nomes dos arquivos vêm um por linha).

Você deve supor que o conjunto de arquivos pode ser armazenado totalmente em memória RAM. Usando sua BST, você deve implementar um índice invertido para o conjunto de palavras que ocorrem nestes livros. Para cada palavra, você deve armazenar onde ela ocorre (número da linha) em

cada arquivo, de forma que, quando necessário, o programa poderá produzir o conjunto de todas as linhas de todos os arquivos em que a palavra ocorre.

5 Buscas do Usuário

Uma vez montado o índice invertido, seu programa deve entrar em modo interativo, respondendo com um *prompt*:

```
ep3 >
```

Supondo que foi fornecido `19033.txt` como entrada, o usuário poderia digitar:

```
ep3 > sister
```

para receber a saída:

```
19033.txt:
    77:Alice was beginning to get very tired of sitting by her sister on the
    79:book her sister was reading, but it had no pictures or conversations in
    1323:head in the lap of her sister, who was gently brushing away some dead
    1326:"Wake up, Alice dear!" said her sister.  "Why, what a long sleep you've
    1330:sister, as well as she could remember them, all these strange adventures
```

onde os números indicam as linhas impressas.

5.1 Calculadora RPN

Você deverá implementar uma "calculadora" RPN (*Reverse Polish Notation*) para manipular os conjuntos de linhas que o usuário criará com suas consultas: o usuário poderá querer as linhas nas quais ocorre a palavra `sister`, unida com o conjunto das linhas em que ocorre a palavra `nonsense`, obtendo um conjunto de linhas S. Da mesma forma, o usuário poderá calcular o conjunto T das linhas em que ocorrem as palavras `she` ou `Alice` (note que T será grande). O usuário então poderia calcular a intersecção de S e T e poderia pedir para ver tal conjunto. A saída seria algo como:

```
19033.txt:
   77:Alice was beginning to get very tired of sitting by her sister on the
   1305:"Stuff and nonsense!" said Alice loudly.  "The idea of having the
   1326:"Wake up, Alice dear!" said her sister.  "Why, what a long sleep you've
```

```
1330:sister, as well as she could remember them, all these strange adventures
```

A implementação da calculadora RPN deve ter os seguintes arquivos:

- RPN.h
- RPN.c

5.2 Mais que um Livro

No caso de estarmos tratando de vários livros, sua saída também tem que especificar os livros. Por exemplo, se além de `19033.txt`, é fornecido o arquivo `tmwht10.txt` (*The Man Who Was Thursday, a nightmare*, Gilbert Keith Chesterton), e o usuário pede as ocorrências de `trees`, a saída seria algo como

```
19033.txt:
   765:"I've tried the roots of trees, and I've tried banks, and I've tried
   791:down again into its nest.  Alice crouched down among the trees as well as
   993:all my life!" Just as she said this, she noticed that one of the trees
   1324:leaves that had fluttered down from the trees upon her face.
tmwht10.txt:
   418:trees like some fierce and monstrous fruit.  And this was strongest
   2078:boy.  But as he turned that corner, and saw the trees and the
   2359:sunlit trees.
   3062:exactly upside down, that all trees were growing downwards and
   4168:trees, and a woman had just stopped singing.  On Syme's heated head
   4464:high up in the air as a high wind sings in the trees.  He thought of
   4811:wood, and disappeared among the twinkling trees.
   5973:are a man of science.  Grub in the roots of those trees and find
   6079:railings shadowed with trees, the six friends were startled, but
   6565:sky, right itself, and then sink slowly behind the trees like a
   6598:He strode off towards the distant trees with a new energy, his rags
   6628:he found himself.  The hedges were ordinary hedges, the trees seemed
   6629:ordinary trees; yet he felt like a man entrapped in fairyland.
   6669:Syme drove through a drifting darkness of trees in utter
   6814:pattern upon whose garment was a green tangle of trees.  For he
```

5.3 Adição e Remoção de Livros

O usuário deverá ser capaz também de adicionar e remover livros durante a interação com seu programa. Por exemplo, o usuário poderia ter executado seu programa com

```
$ ./ep3 19033.txt
```

mas posteriormente, ele poderia querer adicionar `tmwht10.txt` (a partir deste ponto, todas as buscas devem ser feitas nos dois arquivos). Mais tarde, ele poderia remover `19033.txt`. O seu programa deverá permitir tais operações.

6 Especificação do Modo Interativo

Descrevemos aqui como o usuário irá se comunicar com seu programa no modo interativo. Já vimos que seu programa será executado de uma das seguintes formas:

```
$ ./ep3 19033.txt
```

ou

```
$ ./ep3 -flista.txt
```

No modo interativo, seu programa deve apresentar um *prompt*, por exemplo

```
ep3 >
```

Ao digitar uma palavra (e `Enter`):

```
ep3 > <palavra>
```

seu programa deve calcular o conjunto S das linhas em que a `<palavra>` ocorre (uma palavra é uma cadeia maximal de letras, isto é, uma sequência contígua com elementos em {A..Za..z} que não está contida em uma tal cadeia maior). Tal conjunto S deve ser empilhado na pilha dos resultados de sua calculadora RPN. Dessa forma o usuário cria vários conjuntos de linhas. O usuário pode operar com estes conjuntos através dos operadores

- `+`
- `-`
- `\`

Ao receber `+`,

```
ep3 > +
```

sua calculadora deve desempilhar os dois conjuntos S e T no topo da pilha, deve computar $Z = S \cup T$, e deve empilhar Z (o conjunto Z não deve ser

impresso). Os operadores - e \ são semelhantes: eles servem para o usuário obter $S \cap T$ e $S \backslash T$, respectivamente. O usuário poderá também pedir para ver o conjunto S que está no topo da pilha:

```
ep3 > .p
```

Note que o caractere `.` serve para dizer ao seu programa que `p` é um comando. Ao receber `.p`, o seu programa deve imprimir o conjunto de linhas que está atualmente no topo da pilha; o conjunto deve continuar na pilha. Para remover o conjunto da pilha, o usuário digitará

```
ep3 > .D
```

Para adicionar um novo arquivo aos dados, o usuário digitará

```
ep3 > .+<nome do arquivo>
```

Para remover um arquivo do conjunto de dados, o usuário digitará

```
ep3 > .-<nome do arquivo>
```

#Às vezes é útil trocar a ordem dos dois elementos no topo da

Para saber quantos elementos estão na pilha, o usuário dirá

```
ep3 > .NP
```

Para o término da execução, o usuário deve digitar

```
ep3 > .quit
```

7 Observações

1. Este EP é estritamente individual. Programas semelhantes receberão nota 0 (zero).
2. Seja cuidadoso com sua programação (correção, documentação, apresentação, clareza de código, indentação, etc.), dando especial atenção a suas estruturas de dados. A correção será feita levando isso em conta.
3. Entregue seu EP no PACA.
4. Não deixe para fazer seu EP na última hora.
5. Não deixe de incluir em seu código, um *relatório* para discutir seu EP. Discuta as estruturas de dados usadas, os algoritmos usados, etc. Se

você escrever claramente como funciona seu EP, a monitora terá mais facilidade em corrigi-lo e assim você terá uma nota mais alta.

6. Enviem suas dúvidas para a lista de discussão da disciplina (eventualmente, algum colega seu vai respondê-las :-).