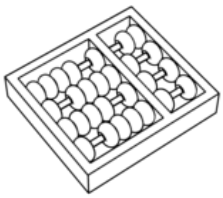


Instituto de Computação
Universidade Estadual de campinas

MC202 - Estruturas de Dados

Laboratório 5

Árvores Binárias

Data de publicação: Sexta feira, 6 de maio de 2016
Prazo máximo de submissão: Sexta feira, 13 de maio de 2016 às 23h59m

Professor: Neucimar J. Leite <neucimar@ic.unciamp.br>

Monitores:
- Juan Hernández (PED) <juan.albarracin@students.ic.unicamp.br>
- Leonardo Yvens (PAD) <leoyvens@gmail.com>

Grupo do curso: https://groups.google.com/d/forum/mc202bc_2016s1
Sítio eletrônico da submissão do código: https://susy.ic.unicamp.br:9999/mc202bc

Enunciado

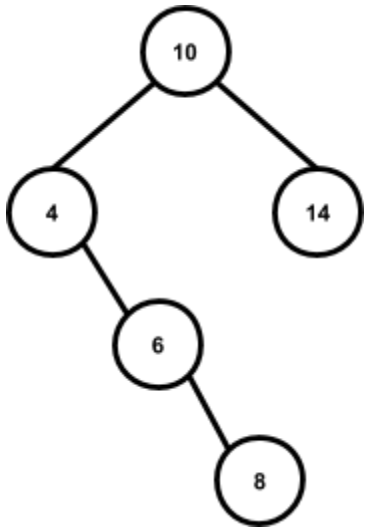
Imagine um motorista que ao se encontrar em uma bifurcação, no seu caminho, tem como saber o preço dos pedágios imediatamente seguintes a cada um dos dois rumos possíveis, e opta pela opção mais barata. Esse agir ingênuo nem sempre faz com que o motorista pague o preço total mínimo em pedágios, já que pode haver opções muito mais baratas após uma opção cara. Seu objetivo é determinar quanto dinheiro o motorista economizaria se considerasse os pedágios de todas as rotas possíveis para decidir qual rumo tomar.

Descrição do problema

Vamos representar as rotas possíveis com uma árvore binária cujos nós representam o preço de um pedágio. Cada folha da árvore representa o último pedágio que o motorista teria de pagar se tivesse escolhido a rota que leva à folha. A árvore tem uma particularidade: para cada nó, o filho da esquerda deve ter um valor menor ou igual a ele, e o filho da direita deve ter um valor estritamente maior que ele. A seguinte tabela apresenta exemplos de árvores corretas e árvores incorretas.

Correto	Incorreto
6 3 / \ / \ 4 7 3 8	6 3 5 / \ / \ / \ 8 9 1 3 4 2

A árvore abaixo



Descreve a seguinte situação:

- O motorista após partir, tem que pagar 10 em um pedágio que não pode evitar.
- Encontra uma bifurcação, e resolve ir pela esquerda, pois aí tem que pagar 4 e não 14.
- Nas seguintes paradas, ele não encontra mais bifurcações, portanto deve pagar 6 e 8 dando um total de 28.

Se ele tivesse escolhido a opção mais cara, teria pago apenas 24. O valor economizado seria 4.

Especificação de entrada e saída

A primeira linha contém o número de nós da árvore e a segunda, os valores de cada nó. As regras para construir a árvore são:

1. O primeiro número da lista será a raiz.
2. Para cada um dos valores seguintes, parte-se da raiz da árvore até atingir uma folha, fazendo o seguinte:
    a. Se o valor a inserir for menor ou igual ao valor do nó atual, desloque-se para o esquerdo. Caso contrário, desloque-se para o filho direito.

3. Ao atingir uma folha, crie um novo nó com o valor a inserir e, se esse valor for menor ou igual ao valor da folha, aloque o nó como filho esquerdo da folha. Caso contrário, aloque-o como filho direito.

A entrada referente à árvore do exemplo anterior seria, entre outra opções possíveis:

```
5
10 4 6 14 8
```

Cabe a você entender por que a entrada é esta e por que pode haver outras ordens de valores na construção da mesma árvore. Seguem alguns exemplos de entradas com suas respectivas saídas.

Entrada	**Saída**	**Cometários**
`5` `10 9 7 4 11`	`9`	A menor rota tem um valor de 21, enquanto a rota do motorista tem um valor de 30.
`10` `10 9 5 1 8 2 6 4 3 11`	`13`	A menor rota tem um valor de 21, enquanto a rota do motorista tem um valor de 34.

Estrutura da submissão

O código fonte deve ser submetido no sistema SuSy para ser executado e testado. O sistema receberá três arquivos:

Nome	**Função**
main.c	Programa principal que lê os dados de entrada, faz as chamadas às funções e escreve a saída.
btree.h	Contém unicamente a declaração do TAD e de funções para operações em árvores binárias. Não será fornecida uma estrutura de arquivo, portanto você tem a liberdade de fazer a sua implementação para demonstrar seu domínio sobre o tema. Espera-se, no mínimo, uma função de criação da árvore, uma função de liberação da árvore e funções para adicionar elementos à árvore.
btree.c	Implementação das funções declaradas em **btree.h**.

Observações

- É mandatório liberar memória dinamicamente alocada.

- Arquivos de teste serão fornecidos para a validação do programa. A avaliação no sistema será feita com esses e outros testes privados.

- O limite de submissões no SuSy é **15**. Recomenda-se executar de forma local seu programa, usando tanto os testes abertos quanto os exemplos aqui fornecidos.
  Submeta seu programa somente após haver testado localmente e de maneira bem sucedida todos os casos.

- A clareza do código e a sua documentação (por meio de comentários) serão avaliadas. O esforço dos monitores para entender o seu código deve ser mínimo. Com a documentação, você demonstra que compreendeu de maneira efetiva os conteúdos do curso referentes ao tema árvores binárias.

- Dúvidas podem ser esclarecidas nas aulas de laboratório ou no grupo do curso indicado no cabeçalho deste documento.

Critério de avaliação

$$nota = 5\frac{n_c}{c} + QD + AC$$

$$0 \leq QD \leq 1$$
$$0 \leq AC \leq 4$$

onde,

n	:	Número total de testes no SuSy
n_c	:	Número de testes corretos
QD	:	Qualidade do código
AC	:	Indicador de quanto o estudante tem demostrado dominar o tema de árvores binárias

A qualidade do código (*QD*) dependerá tanto da legibilidade quanto da documentação. Deve ser fácil para os monitores entender o que foi feito. Já o *AC* é uma nota que depende de quanto o estudante conseguiu convencer o revisor de que sabe implementar corretamente os TADs e operar com a estrutura em questão.

Considerações finais

- Embora haja várias maneiras de resolver os problemas indicados nos laboratórios, o estudante deve optar pela maneira que melhor exercite os conceitos trabalhados em sala de aula. **Não atender a esta advertência invalidará a parte da nota referente aos testes do SuSy**.

- Casos de plágio acarretam **média final zero na disciplina** para todos os envolvidos, sem exceção. O SuSy pode detectar casos de plágio no código, portanto evite compartilhar seu código com outros colegas, mesmo que seja apenas pequenos trechos.