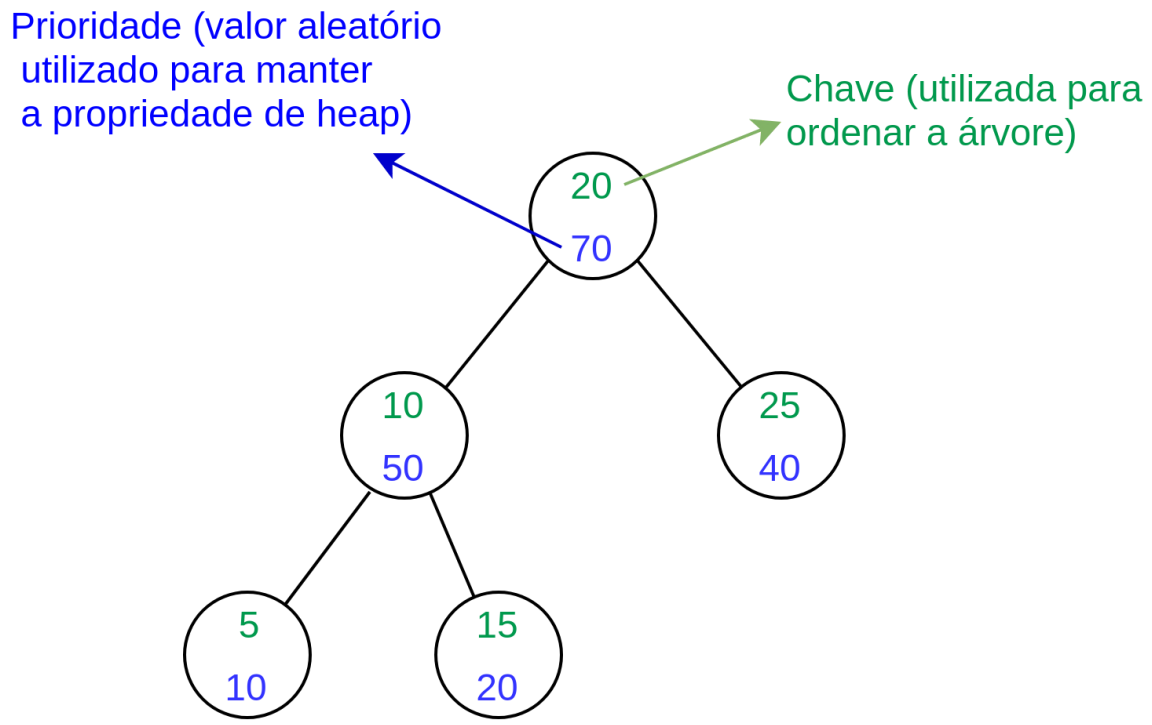
MC202 — ESTRUTURAS DE DADOS

Lab 09 — Árvores Balanceadas

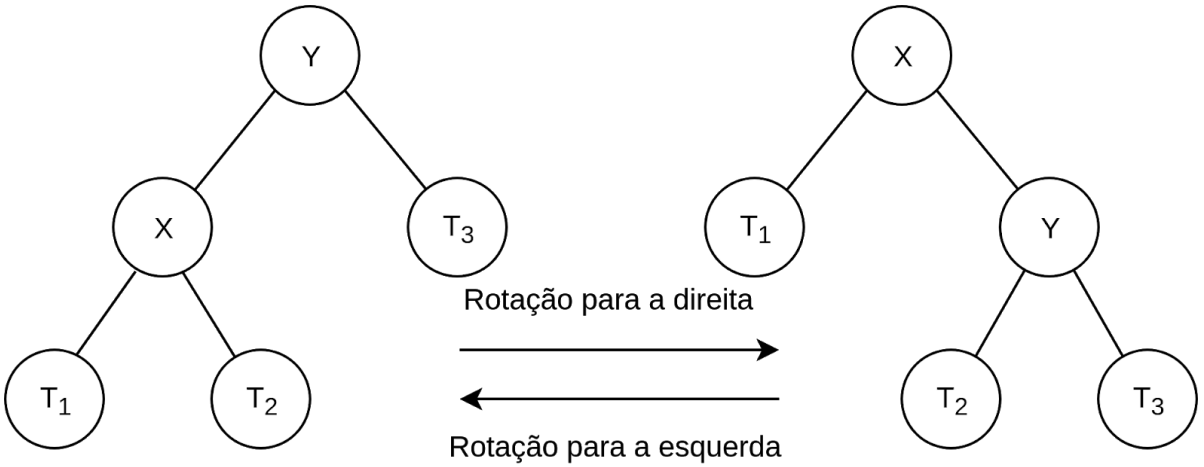
Árvore Binária Aleatória de Busca

Uma Árvore Binária Aleatória de Busca do tipo Treap, assim como as árvores AVL e Rubro-Negra, é uma Árvore Binária Balanceada de Busca, mas não possui altura garantidamente de no máximo $O(log\ N)$. A ideia desse tipo de árvore é usar a aleatoriedade e a propriedade de Heap binária máxima para manter a árvore balanceada com alta probabilidade. Dessa forma, o tempo esperado para inserção, remoção e busca é $O(log\ N)$. Na Treap, cada nó possui uma *chave* e uma *prioridade*. A chave é o valor inserido na árvore e segue a mesma ideia de uma árvore binária comum (os menores valores vão para esquerda e os maiores para a direita). Já a prioridade de cada nó é dada por um valor obtido aleatoriamente durante a inserção e é utilizada para aplicar a propriedade de Heap à árvore. Observe um exemplo de Treap na imagem a seguir.



Rotação

Assim como as demais árvores balanceadas, a Treap utiliza as operações de rotação para a esquerda e rotação para a direita durante a inserção e remoção. Em uma árvore Treap, essas operações são utilizadas para manter a propriedade de Heap máxima.

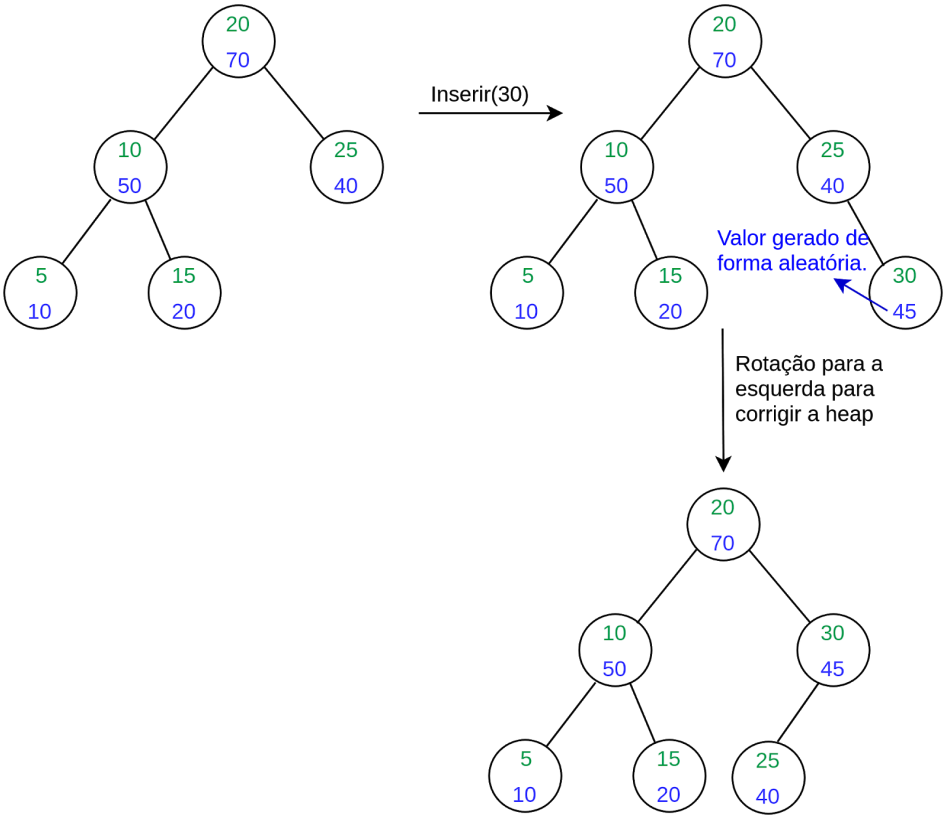


Busca

A busca em uma árvore Treap é realizada da mesma maneira que a busca em uma árvore binária comum.

Inserção

Durante a inserção é criado um novo nó com a chave recebida como parâmetro e com prioridade aleatória. Esse nó é inserido na árvore da mesma maneira que a inserção é realizada em uma árvore binária comum. Após a inserção do novo nó, a propriedade de Heap máxima pode ter sido perdida, nesse caso é necessário utilizar as operações de rotação para refazer a Heap.



Problema

Diariamente são realizadas N transferências do Banco MC202 para outros bancos. A fim de reajustar a tarifa cobrada nessas transações, a diretoria do Banco MC202 quer saber quanto é transferido para outros bancos diariamente e quanto é arrecadado em tarifas nessas transações. Para isso, decidiu que um relatório deverá ser gerado ao final de cada dia com essas informações. Atualmente, os clientes do Banco MC202 pagam uma tarifa K para realizar uma transferência para outro banco.

Durante o dia, todas as transferências do Banco MC202 para outros bancos são registradas, com o identificador do banco de destino e o valor a ser transferido.

Você trabalha no Banco MC202 e precisa implementar um algoritmo que, dado o registro de transferências bancárias do dia atual entre o Banco MC202 e outros bancos, imprima, em ordem de identificador do banco de destino, quanto deve ser transferido para cada banco e quanto de tarifa foi arrecadado pelo Banco MC202 em transferências para cada banco.

Entrada

A primeira linha é composta por dois valores: um inteiro N, $1 \leq N < 10^5$, indicando o número de transferências presentes no registro do banco e um número real K, $1 \leq K \leq 20$, indicando o valor da tarifa cobrada para a realização de uma transferência.

As próximas N linhas são compostas por dois valores: um inteiro I, $1 \leq I < 10^{10}$, que é o identificador do banco para qual o valor deve ser transferido e um número real R, $1 \leq R < 10^6$, que é o valor a ser transferido na transação.

Saída

A saída é composta por B linhas, em que B representa o número de bancos distintos para os quais o Banco MC202 realizou transferências naquele dia. Cada linha deve apresentar o seguinte formato: "`Banco` **`<id>`** `R$` **`<valor>`** `Tarifa R$` **`<T>`**", em que **`<id>`** é o identificador do banco para o qual o dinheiro será transferido, **`<valor>`** é o total transferido para o banco da linha correspondente e **`<T>`** é o valor arrecadado com tarifas em transferências para esse banco. Os valores **`<valor>`** e **`<T>`** devem ser impressos com precisão de duas casas decimais. As linhas devem ser impressas por ordem de identificador dos bancos, do menor para o maior.

Exemplo

Entrada

```
4 10.00
100 600.00
101 250.00
100 100.00
101 50.00
```

Saída

```
Banco 100 R$ 700.00 Tarifa R$ 20.00
Banco 101 R$ 300.00 Tarifa R$ 20.00
```

Observações

- Neste laboratório é obrigatório o uso de uma Árvore Aleatória de Busca Treap.
- Você deverá submeter os seguintes arquivos no SuSy: **lab09.c**, **arv_treap.h** e **arv_treap.c**.
- Para as turmas A, B e C, esse laboratório tem peso 3.
- Observações sobre SuSy:
    - Versão do GCC: 4.4.7 20120313 (Red Hat 4.4.7-17).
    - Flags de compilação:
        - `-ansi -Wall -pedantic-errors -Werror -g -lm`
    - Utilize comentários do tipo `/* comentário */`;
      comentários do tipo `//` serão tratados como erros pelo SuSy
    - Tempo máximo de execução: 2 segundos.
    - Evite usar o sistema como compilador ou apenas para testar as instâncias.
    - Sempre use o software valgrind para detectar erros decorrentes do uso incorreto da memória dinâmica **em todos os testes abertos**. Relembrando:
        - `valgrind --leak-check=full ./lab09 < arq01.in`
- Além das observações acima, esse laboratório será avaliado pelos critérios:
    - Indentação de código e outras boas práticas, tais como:
        - uso de comentários (apenas quando são relevantes);
        - código simples e fácil de entender;
        - sem duplicidade (partes que fazem a mesma coisa).
    - Organização do código:
        - arquivo de cabeçalho (.h) contendo definições de constantes, tipos de dados criados pelo usuário, protótipo de funções etc;
        - arquivos de implementação (.c) que implementam as funções definidas no arquivo de cabeçalho;

- tipos de dados criados pelo usuário e funções bem definidas e tão independentes quanto possível.

- Corretude do programa:
  - programa correto e implementado conforme solicitado no enunciado;
  - desalocar toda memória alocada dinamicamente durante a execução do programa;
  - se não realiza leitura ou escrita em blocos de memória não alocados;
  - inicialização de variáveis sempre que for necessário;
  - dentre outros critérios.

Apêndice

Remoção em Treap

Suponha que desejamos remover o nó j da árvore. Se j for uma folha, basta removê-lo. Caso contrário, atribuímos $-\infty$ à prioridade de j e refazemos a Heap utilizando operações de rotação. Dessa forma o nó j passará a ser uma folha e então podemos removê-lo.