Árvores

Árvores Binárias de Busca na STL

Prof. Edson Alves - UnB/FGA

2018

Sumário

Introdução

Árvores Binárias de Busca na STL

- A STL (*Standard Template Library*) da linguagem C++ não oferece uma implementação básica de árvores binárias de busca que permita o acesso direto aos nós e seus ponteiros
- Entretanto, ela oferece tipos de dados abstratos cuja implementação usa árvores binárias de busca auto-balanceáveis
- O padrão da linguagem não especifica qual árvore deve ser utilizada na implementação, e sim as complexidades assintóticas esperadas para cada operação
- Segundo o site CppReference[1], em geral são utilizadas árvores *red-black*
- Os principais tipos de dados abstratos implementados são os conjuntos (*sets*) e os dicionários (*maps*)

[1] https://en.cppreference.com/w/

Set

Set

- O conjunto (`set`) é um tipo de dado abstrato que representa um conjunto de elementos únicos
- Estes elementos são mantidos em ordem crescente, de acordo com a implementação do operator $<$ do tipo de elemento a ser armazenado
- O tipo de dado a ser armazenado é paramétrico, e deve ser definido na instanciação do conjunto
- A principal característica dos conjuntos é a eficiência nas operações de inserção, remoção e busca
- Todas as três tem complexidade $O(N)$, onde N é o número de elementos do conjunto

Construção de um set

- O padrão C++11 oferece cinco construtores distintos para um `set`
- O primeiro deles, denominado *default constructor*, não tem parâmetros, e constrói um conjunto vazio
- O segundo, *range constructor*, permite a construção de um conjunto a partir de dois iteradores, `first` e `last`, que determinam um intervalo de valores a serem inseridos, do primeiro ao penúltimo
- Este construtor também permite a definição de um alocador de memória customizado
- O terceiro, *copy constructor*, cria uma cópia exata do `set` passado como parâmetro
- O quarto, *move constructor*, move o conteúdo do `set` passado como parâmetro para o novo conjunto
- O quinto, *initializer-list constructor*, cria um novo `set` com os elementos passados na lista de inicialização

Exemplo de uso dos construtores do set

```cpp
#include <set>
#include <string>

#include <iostream>

int main()
{
    std::set<int> s1;       // Conjunto de inteiros vazio

    std::string s { "Teste" };
    std::set<char> s2(s.begin() + 1, s.end());  // s2 = { 'e', 's', 't' }

    std::set<char> s3(s2);                      // s3 == s2

    std::set<char> s4(std::move(s3));           // s4 == s2, s3 vazio

    std::set<double> s5 { 2.0, 1.5, 3.7 };      // s5 = { 1.5, 2.0, 3.7 }

    return 0;
}
```

Referências

1. CppReference – Map, acesso eme 03/04/2019.
2. CppReference – Set, acesso em 03/04/2019.