YANGI O‘ZBEKISTON, YANGI TADQIQOTLAR JURNALI
Volume 2 Issue 8 06.05.2025
https://phoenixpublication.net/ Online ISSN: 3030-3494

TREE MA’LUMOTLAR TUZILMASI VA UNING METODLARI BILAN ISHLASH

РАБОТА СО СТРУКТУРОЙ ДАННЫХ "ДЕРЕВО" И ЕЁ МЕТОДАМИ

WORKING WITH THE TREE DATA STRUCTURE AND ITS METHODS

Abdullayev Shaxboz Solijon o‘g‘li
FarDU Axborot texnologiyalari kafedrasi katta o'qituvchisi
shaxbozfardu2023@gmail.com
ORCID ID 0000-0001-9382-732X
Sharoffidinov Shahobiddin Norinboy o’g’li
FarDU Axborot tizimlari va texnologiyalari yo’nalishi 1-kurs talabasi
sharofiddinovshahobiddin062@gmail.com

Annotatsiya . *Mazkur maqolada daraxt (Tree) ma’lumotlar tuzilmasi chuqur o‘rganilib, uning zamonaviy dasturlashda tutgan o‘rni, afzalliklari va amaliy qo‘llanilish sohalari yoritiladi. Daraxt tuzilmasi ma’lumotlarni mantiqiy va ierarxik tartibda saqlash imkonini berib, murakkab qidiruv, tartiblash hamda tahlil jarayonlarini samarali tashkil etish uchun xizmat qiladi. Maqolada daraxtning turlari — oddiy Binary Tree, Binary Search Tree, AVL Tree, Red-Black Tree va B-Tree kabi ko‘p uchraydigan variantlari tahlil qilinadi. Shuningdek, har bir turdagi daraxt bilan ishlashda qo‘llaniladigan asosiy metodlar — tugun qo‘shish, o‘chirish, qidirish, pastga va yuqoriga yurish (traversal) usullari yoritiladi. Har bir algoritm uchun Big-O murakkablik darajasi keltirilib, ular o‘zaro solishtiriladi. Ushbu maqola nafaqat nazariy asoslarni, balki real dasturlashda daraxtlar qanday qo‘llanilishini ko‘rsatishga qaratilgan bo‘lib, talabalar, dasturchilar va ilmiy tadqiqotchilar uchun foydali bo‘lishi mumkin.*

Аннотация. *В данной статье подробно рассматривается структура данных «дерево», её значение в современной программной инженерии, а также преимущества и области практического применения. Деревья позволяют хранить данные в логической и иерархической форме, что делает возможным эффективную реализацию поиска, сортировки и анализа данных. В статье*

рассматриваются различные типы деревьев, включая бинарное дерево, дерево бинарного поиска, AVL-дерево, красно-черное дерево и B-дерево. Подробно описываются основные методы работы с деревьями: добавление, удаление, поиск узлов, а также методы обхода дерева (traversal). Для каждого алгоритма приводится оценка сложности по Big-O и проводится сравнительный анализ. Статья направлена не только на раскрытие теоретических основ, но и на демонстрацию практического применения деревьев в реальных программных решениях, что делает её полезной для студентов, разработчиков и исследователей.

Abstract. *This article provides a comprehensive analysis of the tree data structure, emphasizing its importance in modern software engineering, advantages, and practical applications. Trees enable data to be stored in a logical and hierarchical format, allowing for efficient implementation of searching, sorting, and data analysis processes. The paper explores common tree types such as Binary Trees, Binary Search Trees, AVL Trees, Red-Black Trees, and B-Trees. It also covers essential operations including node insertion, deletion, searching, and traversal methods (in-order, pre-order, post-order, level-order). Each algorithm is evaluated based on Big-O time and space complexity, and comparative insights are presented. This work aims to bridge theoretical foundations with practical implementations, making it valuable for students, software developers, and academic researchers alike.*

Kalit so‘zlar: *daraxt ma’lumotlar tuzilmasi, binary tree, AVL daraxti, tugun (node), qidiruv algoritmi, traversal metodlari, ma’lumotlarni saqlash, Big-O tahlil, rekursiv algoritmlar, daraxtlarni solishtirish, algoritmik samaradorlik.*

Ключевые слова: *структура данных дерево, бинарное дерево, AVL-дерево, узел (node), алгоритм поиска, методы обхода, хранение данных, анализ Big-O, рекурсивные алгоритмы, сравнение деревьев, алгоритмическая эффективность.*

Keywords: *tree data structure, binary tree, AVL tree, node, search algorithm, traversal methods, data storage, Big-O analysis, recursive algorithms, tree comparison, algorithmic efficiency.*

KIRISH

Zamonaviy dasturlashda ma’lumotlar bilan samarali ishlash muhim ahamiyatga ega bo‘lib, bu jarayonda turli ma’lumotlar tuzilmalarining o‘rni beqiyosdir. Ushbu tuzilmalar orasida daraxt (Tree) ma’lumotlar tuzilmasi alohida o‘rin egallaydi. Daraxt tuzilmasi ma’lumotlarni ierarxik shaklda saqlash va ularga tezkor murojaat qilish imkonini beradi. Bu tuzilma nafaqat nazariy jihatdan, balki amaliy dasturlashda ham keng qo‘llaniladi — fayl tizimlari, ma’lumotlar bazasi indekslari, sun’iy intellekt, kompyuter grafikalari, kompyuter tarmoqlari va kompilyatorlar yaratishda daraxtlar asosiy rol o‘ynaydi.

Daraxt tuzilmasi oddiy ko‘rinishda Binary Tree sifatida boshlanib, yanada murakkabroq shakllar — Binary Search Tree (BST), AVL daraxti, Red-Black Tree va B-Tree kabi turlarga bo‘linadi. Har bir turli daraxt tuzilmasi o‘zining maxsus qo‘llanilish doirasiga, tezkorlik darajasiga va samaradorligiga ega. Ular bilan ishlashda tugun (node) qo‘shish, o‘chirish, qidirish va daraxtni turli usullar bilan aylanib chiqish (traversal) kabi metodlardan foydalaniladi.

Mazkur ishda daraxt tuzilmasining asosiy turlari, ular bilan ishlashda qo‘llaniladigan metodlar, ularning afzallik va kamchiliklari, shuningdek har bir algoritmning Big-O murakkablik tahlili keng yoritiladi. Shu bilan birga, daraxtlar boshqa ma’lumotlar tuzilmalariga nisbatan qanday ustunliklarga ega ekani va real dasturlashdagi muammolarga qanday yechim taklif qilishi tahlil qilinadi. Ishning asosiy maqsadi — daraxt tuzilmasini chuqurroq o‘rganish, algoritmik fikrlashni shakllantirish hamda dasturlash amaliyotida optimal tuzilmalarni tanlay olish ko‘nikmalarini rivojlantirishdir.

Dasturning kodi :

```
#include <iostream>
using namespace std;
struct Tugun {
    int qiymat;
    Tugun* ong;
    Tugun* chap;
    Tugun(int qiymat) {
        this->qiymat = qiymat;
        ong = chap = nullptr;}};
```

```
Tugun* qosish(Tugun* ildiz, int qiymat) {
    if (ildiz == nullptr) {
        return new Tugun(qiymat); }
    if (qiymat < ildiz->qiymat)
        ildiz->chap = qosish(ildiz->chap, qiymat);
    else if (qiymat > ildiz->qiymat)
        ildiz->ong = qosish(ildiz->ong, qiymat);
    return ildiz;}
void tartib_bilan_chiqarish(Tugun* ildiz) {
    if (ildiz != nullptr) {
        tartib_bilan_chiqarish(ildiz->chap);
        cout << ildiz->qiymat << " ";
        tartib_bilan_chiqarish(ildiz->ong);}}
bool qidirish(Tugun* ildiz, int qiymat) {
    if (ildiz == nullptr) return false;
    if (ildiz->qiymat == qiymat) return true;
    if (qiymat < ildiz->qiymat)
        return qidirish(ildiz->chap, qiymat);
    else
        return qidirish(ildiz->ong, qiymat);}
int main() {
    Tugun* ildiz = nullptr;
    ildiz = qosish(ildiz, 50);
    qosish(ildiz, 30);
    qosish(ildiz, 70);
    qosish(ildiz, 20);
    qosish(ildiz, 40);
    qosish(ildiz, 60);
    qosish(ildiz, 80);
    cout << "Tartib bilan chiqarish: ";
    tartib_bilan_chiqarish(ildiz);
    cout << endl;
    int qiymat;
```

```
    cout << "Qidirilayotgan qiymatni kiriting: ";
    cin >> qiymat;
    if (qidirish(ildiz, qiymat))
        cout << qiymat << " daraxtda mavjud.\n";
    else
        cout << qiymat << " daraxtda mavjud emas.\n";
    return 0;  }
```

Dastur tahlili

#include <iostream> - Bu kutubxona fayli konsolga ma'lumot chiqarish (cout) va konsoldan ma'lumot olish (cin) imkoniyatlarini beradi. Bu satr dasturda ishlatilgan barcha konsol bilan aloqador funktsiyalarni chaqirishi uchun zarur.

using namespace std; - Bu satr std nomli kutubxonani dasturga kiritadi. std namespace ichidagi barcha elementlardan bevosita foydalanish imkonini beradi. Masalan, cout, cin, string kabi elementlar to'g'ridan-to'g'ri chaqiriladi va "std::" prefiksini qo'llash kerak emas.

int main() - Dastur bajarilishi shu funksiyadan boshlanadi.

struct Tugun { ... }; - bu yerda biz **Tugun** nomli tuzilmani yaratmoqdamiz. **Tugun** daraxtning tuguni (node) bo‘lib, har bir tugunda uchta maydon mavjud:

int qiymat; - bu tugunning qiymatini saqlash uchun integer o‘zgaruvchisi.

Tugun* ong; va **Tugun* chap; -** bu yerda, ong va chap pointerlar (ko‘rsatkichlar) bo‘lib, ular chaqiruvchi tugunning chap va o‘ng farzandlarini ko‘rsatadi. nullptr boshlang‘ich qiymat sifatida tayinlangan, bu pointerlar hech narsa ko‘rsatmaydi.

Tugun(int qiymat) { ... } - bu konstruktor. U yangi **Tugun** yaratish uchun chaqiriladi va tugun qiymatini o‘rnatadi, shuningdek ong va chap ko‘rsatkichlarini nullptr ga o‘rnatadi.

Tugun* qosish(Tugun* ildiz, int qiymat) - Bu funksiya binary search tree (BST) ga yangi tugun qo‘shish uchun ishlatiladi.

if (ildiz == nullptr) - Agar ildiz nullptr bo‘lsa (ya'ni daraxt bo‘sh bo‘lsa), yangi **Tugun** yaratiladi va qaytariladi.

if (qiymat < ildiz->qiymat) - Agar kiritilgan qiymat ildizning qiymatidan kichik bo‘lsa, qiymat chap bolaga qo‘shiladi.

else if (qiymat > ildiz->qiymat) - Agar kiritilgan qiymat ildizning qiymatidan katta bo‘lsa, qiymat o‘ng bolaga qo‘shiladi. Funksiya oxirida **ildiz** qaytariladi.

void tartib_bilan_chiqarish(Tugun* ildiz) - Bu funksiya daraxtni inorder usulida (chap-ildiz-o‘ng) chiqaradi.

if (ildiz != nullptr): Agar tugun mavjud bo‘lsa, rekursiv tarzda chap bolaga o‘tadi.

tartib_bilan_chiqarish(ildiz->chap); - Chap bolani chiqargandan keyin tugunning qiymatini chiqaradi.

cout << ildiz->qiymat << " "; - Ildiz tugunning qiymatini chiqaradi.

tartib_bilan_chiqarish(ildiz->ong); - O‘ng bolani chiqargandan keyin recursive tarzda daraxtni yuritishni davom ettiradi.

bool qidirish(Tugun* ildiz, int qiymat) - Bu funksiya binary search tree (BST) da kiritilgan qiymatni qidiradi.

if (ildiz == nullptr) - Agar daraxt bo‘sh bo‘lsa, qiymat mavjud emas, shuning uchun false qaytariladi.

if (ildiz->qiymat == qiymat) - Agar tugunning qiymati kiritilgan qiymatga teng bo‘lsa, true qaytariladi.

if (qiymat < ildiz->qiymat) - Agar kiritilgan qiymat tugunning qiymatidan kichik bo‘lsa, chap bolada qidirish davom ettiriladi.

else - Agar qiymat katta bo‘lsa, o‘ng bolada qidirish davom ettiriladi.

int main(): Dasturning asosiy qismi.

Tugun* ildiz = nullptr; - Ildiz tuguni (daraxt) boshlang‘ich qiymat sifatida nullptr ga o‘rnatiladi.

ildiz = qosish(ildiz, 50); - Daraxtga 50 qiymatini qo‘shish.

qosish(ildiz, 30);, qosish(ildiz, 70); - Daraxtga boshqa qiymatlar qo‘shiladi.

tartib_bilan_chiqarish(ildiz); - Daraxtni inorder usulida chiqarish.

cout << "Qidirilayotgan qiymatni kiriting: "; - Klaviaturadan qiymat kiritish so‘raladi.

cin >> qiymat; - Kiritilgan qiymatni olish.

if (qidirish(ildiz, qiymat)) - Kiritilgan qiymat daraxtda bormi, deb tekshiriladi.

return 0; - return 0 satri dasturning to'liq yakunlangani va xatolik yo'qligini bildiradi.

Dasturning ishlash prinsipi:

Agar kiritilgan son daraxtda mavjud bo’lsa:

```
Tartib bilan chiqarish: 20 30 40 50 60 70 80
Qidirilayotgan qiymatni kiriting: 40
40 daraxtda mavjud.

Process returned 0 (0x0)   execution time : 4.946 s
Press any key to continue.
```

Agar kiritilgan son daraxtda mavjud bo’lmasa:

```
Tartib bilan chiqarish: 20 30 40 50 60 70 80
Qidirilayotgan qiymatni kiriting: 102
102 daraxtda mavjud emas.

Process returned 0 (0x0)   execution time : 8.511 s
Press any key to continue.
```

XULOSA

Ushbu dastur binary search tree (BST) ma'lumotlar tuzilmasini yaratish va ishlatishga doir oddiy va samarali bir yechimni taqdim etadi. Dasturda daraxtga yangi qiymatlar qo‘shish, daraxtni inorder usulida chiqarish va kiritilgan qiymatni daraxtda qidirish kabi asosiy operatsiyalar ko‘rsatilgan. Inorder traversal orqali daraxtni chiqarish natijasida qiymatlar kichikdan katta bo‘lgan tartibda joylashadi, bu esa BST tuzilmasining asosiy xususiyatlaridan biridir.

Dasturda kiritilgan qiymatni qidirish uchun ishlatiladigan algoritmning samaradorligi yuqori bo‘lib, qidirish va qo‘shish operatsiyalari o‘rtacha **O(log n)** murakkablikka ega. Bu esa kattaroq daraxtlar bilan ishlaganda ham samarali ishlash imkonini beradi. Dastur foydalanuvchilarga daraxtning tuzilishini yaxshiroq tushunishga yordam beradi va ma'lumotlar tuzilmalari bo‘yicha bilimlarni kengaytiradi.

FOYDALANILGAN ADABIYOTLAR:

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C.
Introduction to Algorithms. — MIT Press, 3rd Edition, 2009.
2. Weiss, M. A. Data Structures and Algorithm Analysis in C++. — Pearson, 4th Edition, 2014.
3. Sedgewick, R., & Wayne, K.
Algorithms (4th Edition). — Addison-Wesley, 2011.
4. Knuth, D. E.
The Art of Computer Programming, Volume 1: Fundamental Algorithms. — Addison-Wesley, 3rd Edition, 1997.
5. Wikipedia contributors
Tree (data structure) — Wikipedia, The Free Encyclopedia.
6. GeeksforGeeks