11 - AMALIY MASHG‘ULOT. *USTIVOR NAVBATLAR.*

Ishdan maqsad: Talabalarda Ustuvor navbatlar haqida bilim va ko‘nikmalarini oshirish. Dasturlash tillarida ustuvor navbatlardan foydalanishni o‘rganish va qo‘llay olish.

Nazariy qism: Ko’pgina ilovalar kalitlarga ega bo’lgan elementlarni qayta ishlashni talab qiladi, lekin ular to’liq tartibda va birdaniga hammasi emas. Ko’pincha, biz bir qator narsalarni to’playmiz, so’ngra eng katta kalit bilan ishlov beramiz, keyin ko’proq narsalarni to’playmiz, so’ngra hozirgi eng katta kalit bilan ishlov beramiz va hokazo. Bunday muhitda tegishli ma’lumotlar turi ikkita amalni qo’llab-quvvatlaydi: maksimal miqdorni o‘chirish va joylashtirish. Bunday ma’lumotlar turi ustivor navbat deb nomlanadi.

Ustivor navbatlar odatdagi navbat yoki stek ma’lumotlar tuzilmasiga o’xshash abstrakt ma’lumotlar turi bo’lib, unda har bir element qo’shimcha ravishda bog‘liq bo’lgan "ustivorlikka" ega. Ustivor navbatda yuqori ustivor element past ustivor elementdan oldin xizmat qiladi. Ustivor navbatlar ko’pincha uyum (kucha) bilan amalga oshirilsa-da, ular konseptual jihatdan uyumlardan farq qiladi. Ustivor navbat - bu "ro’yxat" yoki "karta" ga o’xshash narsa; Ro’yxat bog‘langan ro’yxat yoki massiv yordamida amalga oshirilishi mumkin bo’lganidek, ustivor navbat uyum yoki tartiblanmagan massiv kabi boshqa usullar yordamida amalga oshirilishi mumkin.

Ustivor navbat - bu yozuvlar bir-biri bilan chiziqli taqqoslanadigan kalitlarga (masalan, raqamlar) ega bo’lgan va ikkita amalni realizatsiya qiladigan axborot tizimidir. Bu ikki amal tizimga tasodifiy yozuvni kiritish va yozuv tizimidan eng kichigi bilan tanlov kalit.

Dasturiy ta’minot tizimlarida ustivor navbatlar juda keng tarqalgan va dasturlarning ishlashi to’g‘ridan-to’g‘ri ularni amalga oshirish samaradorligiga bog‘liq.

Ustivorda navbatda qo‘llab-quvvatlanadigan amallar quyidagilar hisoblanadi:

1) ***Insert*** - navbatga element qo’shish;

2) ***Max*** - ustivorligi yuqori bo’lgan elementni qaytaradi;

3) ***ExtractMax*** - navbatdagi eng ustivor elementni olib tashlaydi;

4) ***IncreaseKey*** - berilgan elementning ustivor qiymatini o’zgartiradi;

5) ***Merge*** - ikkita navbatni bittaga birlashtiradi.

Binar uyum (kucha) - piramida (binary heap)

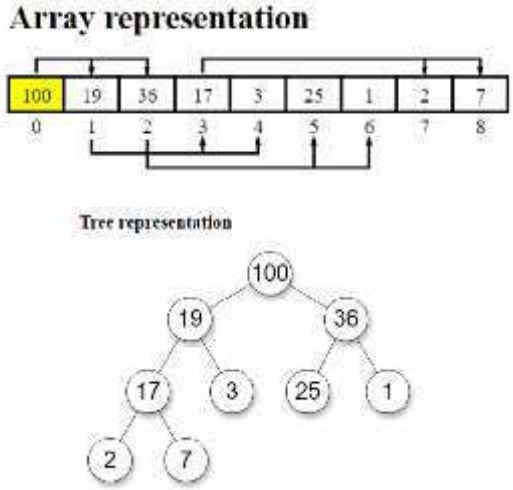
Binary heap, ustuvor binarning bir turi bo’lib, asosan odamlar tomonidan raqamlar yoki obyektlar kabi ma’lumotlar tuzilishida ishlatiladi. Ustuvor binlar ko’p dasturlashda keng qo’llaniladi, masalan, qisqa yo’ldan eng katta yoki eng kichik elementni topish, ko’pgina algoritmlarni amalga oshirish va boshqa ishlar uchun.

Ustuvor binning ikkita muhim xususiati bor:

Tartiblanishi: Ustuvor bin hamma vaqtlar oddiy tartibda joylashadi. Agar bu minimum ustuvor bin bo'lsa, har bir ota elementi o'z farzandlari ostida katta bo'ladi, va buning oldida minimum qiymat o'rniga keladi. Agar bu maksimum ustuvor bin bo'lsa, har bir ota elementi o'z farzandlari ostida kichik bo'ladi, va buning oldida maksimum qiymat o'rniga keladi.

Struktura: Ustuvor bin mazmuni ro'yxatga o'xshash bo'lib, uning har bir elementi birinchi o'zgaruvchan holda joylashgan ro'yxat bilan bog'liq. Bu holda, ustuvor bin asosan massiv yoki ro'yxat strukturasini ishlatadi.

Ustuvor binning strukturasini ko'rish uchun quyidagi misolni ko'rib chiqamiz:



Ustuvor binning bir necha xususiyatlari mavjud:

- Har bir ota element o'z farzandlari ostida joylashgan.
- Elementlar oddiy tartibda joylashadi.
- Eng kichik (minimum ustuvor bin) yoki eng katta (maksimum ustuvor bin) element ustida joylashadi.

Amaliy qism:

Ustuvor binlar, ko'p algoritmlarda foydalaniladi, masalan, ustuvor qatori qurish, eng katta/eng kichik elementni topish, ustuvor qatordan element olib tashlash kabi. Ular hamda katta dasturlarning asosiy qismlarida ishlatiladi.

Quyidagi C++ kodida bu algoritm ko'rsatilgan:

```
#include <iostream>
#include <vector>
// Ustuvor bin tuzilmasi
class BinaryHeap {
private:
    std::vector<int> heap;
    // Ota va farzandlarning indekslarini hisoblash
    int parent(int i) { return (i - 1) / 2; }
```

```
    int leftChild(int i) { return 2 * i + 1; }
    int rightChild(int i) { return 2 * i + 2; }
    // Elementlarni almashtirish
    void swap(int &a, int &b) {
        int temp = a;
        a = b;
        b = temp;
    }
    // Ustuvor bin qoidalari bo'yicha ustuvor binning tasdiqlash
    void heapifyUp(int i) {
        while (i > 0 && heap[parent(i)] < heap[i]) {
            swap(heap[i], heap[parent(i)]);
            i = parent(i);
        }
    }
    void heapifyDown(int i) {
        int maxIndex = i;
        int l = leftChild(i);
        int r = rightChild(i);
        if (l < heap.size() && heap[l] > heap[maxIndex]) {
            maxIndex = l;
        }
        if (r < heap.size() && heap[r] > heap[maxIndex]) {
            maxIndex = r;
        }
        if (i != maxIndex) {
            swap(heap[i], heap[maxIndex]);
            heapifyDown(maxIndex);
        }
    }
public:
    // Element qo'shish
    void insert(int value) {
        heap.push_back(value);
        heapifyUp(heap.size() - 1);
    }
    // Eng katta elementni olish
    int getMax() {
        if (!heap.empty()) {
            return heap[0];
        }
```

```
        return -1; // Ustuvor bin bo'sh
    }
    // Eng katta elementni olib tashlash
    int extractMax() {
        int result = getMax();
        if (heap.size() > 1) {
            heap[0] = heap.back();
            heap.pop_back();
            heapifyDown(0);
        } else if (heap.size() == 1) {
            heap.pop_back();
        }
        return result;
    }
    // Ustuvor binning hajmi
    int size() {
        return heap.size();
    }
};
int main() {
    // Ustuvor bin obyektini yaratish
    BinaryHeap heap;
    // Ustuvor binning sinov qismi
    heap.insert(10);
    heap.insert(5);
    heap.insert(20);
    heap.insert(15);
    std::cout << "Ustuvor binning eng katta elementi: " << heap.getMax() <<
std::endl;
    std::cout << "Ustuvor binning eng katta elementini olib tashlash: " <<
heap.extractMax() << std::endl;
    std::cout << "Ustuvor binning hajmi: " << heap.size() << std::endl;
    return 0;
}
```

Uyum (kucha)larni saralash (Heap-Sort)

Heapsort algoritmi ustuvor binlardan foydalanib ro'yxatni saralovchi algoritmdir. Ustuvor binlar (ya'ni kuchalar) struktura ravishida ma'lumotlar to'plamlarini saqlashda foydalaniladi va ba'zi operatsiyalar ustida bajariladi.

Ustuvor binlar qoidalarga mos ravishda o'rnatilgan ma'lumotlar to'plami bo'lib, qarshioldi qismida eng katta (ya'ni maksimum) qiymat joylashadi. Ustuvor binning

boshqaruvi asosida, elementlar qo'shilishi, olib tashlanishi va boshqa operatsiyalar amalga oshirilishi mumkin. Heapsort algoritmi quyidagi bosqichlardan iborat:

Boshlang'ich ustuvor bin tuzilishi: Boshlang'ich ustuvor bin tuzilmasi yaratiladi, kiritilgan ro'yxat ustida. Bu o'ngga tartiblangan ustuvor bin boshlang'ich tartiblanmagan ro'yxatni o'lchovlash imkonini beradi.

Ustuvor bin tuzilmasi hisoblanishi: Boshlang'ich ustuvor bin hisoblanganidan so'ng, ro'yxatning har bir elementi kuchalarga joylashadi va ustuvor binning tuzilmasi qayta tartiblanadi.

Ustuvor binning kuchalarni qayta tuzish: Har bir element ustuvor binning boshiga olib tashlanadi va boshqa elementlar qo'shilishi uchun bo'sh joyga joylashtiriladi. Keyin, kuchalar qayta tuziladi, joriy element eng katta element sifatida qo'shiladi.

Ro'yxat saralash: Kuchalarni qayta tartiblash jarayonidan so'ng, ro'yxat saralandi va tartiblangan ro'yxat beriladi.

Heapsort algoritmi asosan kuchalar ustida operatsiyalarni bajaradi, shuningdek saralash uchun O(n log n) vaqt kompleksliyatga ega bo'lgan eng yaxshi amaliy saralash algoritmlaridan biridir. Ustuvor binlar bilan amalga oshirilgan ro'yxatlarni saralishda, Heapsort kuchalarni qayta tuzish jarayonini o'rtacha vaqtim vaqtimga bajara oladi.

Heapsort algoritmi darhaqiqat saralash uchun ishlatiladi va xususan, ma'lumotlarning joylangan joylarni o'rnatolmaslik sharoitida yaxshi ishlashadi. Ustuvor binlar tuzilmasi va Heapsort algoritmi tajribaviy dasturchilar tomonidan ishlab chiqilgan yuqori darajadagi algoritm va strukturadir.

Quyidagi C++ kodida, ro'yxatni ustuvor bin qo'llanmasidan foydalanib saralovchi heapsort algoritmi keltirilgan:

```
#include <iostream>
#include <vector>
// Ustuvor bin tuzilmasi
class HeapSort {
public:
   // Heapsort algoritmi
   static void sort(std::vector<int>& arr) {
      int n = arr.size();
      // Boshlang'ich ustuvor bin tuzish
      for (int i = n / 2 - 1; i >= 0; i--) {
         heapify(arr, n, i);
      }
      // Elementlarni ustuvor bin tuzilmasida saralash
```

```
        for (int i = n - 1; i >= 0; i--) {
            // Eng katta elementni ustuvor bin boshiga olib chiqarish
            std::swap(arr[0], arr[i]);
            // Ustuvor binning qayta tuzilishi
            heapify(arr, i, 0);
        }
    }
private:
    // Ustuvor bin tuzilmasi hisoblanishi
    static void heapify(std::vector<int>& arr, int n, int i) {
        int largest = i; // Eng katta element indeksi
        int left = 2 * i + 1; // Chap farzand indeksi
        int right = 2 * i + 2; // O'ng farzand indeksi

        // Chap farzand eng katta elementga teng yoki katta bo'lsa
        if (left < n && arr[left] > arr[largest]) {
            largest = left;
        }
        // O'ng farzand eng katta elementga teng yoki katta bo'lsa
        if (right < n && arr[right] > arr[largest]) {
            largest = right;
        }
        // Agar eng katta element bu ota element emas bo'lsa
        if (largest != i) {
            // Eng katta elementni boshqa joyga almashtirish
            std::swap(arr[i], arr[largest]);
            // Qayta ustuvor binning qayta tuzilishi
            heapify(arr, n, largest);
        }
    }
};
int main() {
    std::vector<int> arr = {12, 11, 13, 5, 6, 7};
    // Heapsort algoritmini ishga tushiramiz
    HeapSort::sort(arr);
    // Saralgan ro'yxatni chiqaramiz
    std::cout << "Saralgan ro'yxat: ";
    for (int i = 0; i < arr.size(); i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
```

```
    return 0;
}
```

Ustuvor binning kuchalarni qayta tuzish va ro'yxatni saralash bosqichlari HeapSort::sort() funksiyasida amalga oshiriladi. Ustuvor binning qayta tuzilishi va ro'yxatni saralash amaliyoti quyidagi tartibda bajariladi:

1. Boshlang'ich ustuvor bin tuzilishi;
2. Ro'yxatni ustuvor binni tartibida saralash;
3. Eng katta elementni o'zidan olib tashlash va kuchalarni qayta tuzish;
4. Ro'yxatni saralash.

Mustaqil bajarish uchun topshiriqlar

1. Berilgan elementlar ro'yxatidan Binary Heap tuzilmasini yaratuvchi C++ klass yarating.
2. Berilgan bir Binary Heap tuzilmasi uchun elementni qo'shuvchi funksiya yozing.
3. Binary Heap tuzilmasida eng katta elementni aniqlash uchun funksiya yozing.
4. Binary Heap tuzilmasini chiqarish uchun funksiya yozing.
5. Binary Heap tuzilmasida elementni izlash uchun funksiya yozing.
6. Binary Heap tuzilmasida eng katta elementni olib tashlash uchun funksiya yozing.
7. Berilgan ro'yxatni Binary Heap tuzilmasiga o'zgartiruvchi funksiya yozing.
8. Binary Heap tuzilmasida eng kichik elementni aniqlash uchun funksiya yozing.
9. Berilgan ro'yxatni saralash uchun Heapsort algoritmini yozing.
10. Tasodifiy sonlar ro'yxatini yaratib, uni saralash uchun Heapsort algoritmini ishga tushiring.
11. Heapsort algoritmini qayta yozing, lekin bu safar to'rtinchi qismida tartiblangan elementni yuqoriga olib chiqaradi.
12. Berilgan tasodifiy sonlar ro'yxatini Heapsort algoritmi yordamida saralang. Natijani ekranga chiqaring va to'g'ri bo'lganligini tekshiring.
13. Berilgan matnlar ro'yxatini saralash uchun Heapsort algoritmini ishga tushiring.
14. Heapsort algoritmini ishlatib, berilgan matnlar ro'yxatini saralang. Natijani ekranga chiqaring va to'g'ri bo'lganligini tekshiring.
15. Heapsort algoritmini ishlatib, berilgan sonlar ro'yxatini saralang. Saralgan sonlar ro'yxatini chiqaring.
16. Berilgan sonlar ro'yxatini Heapsort algoritmi yordamida saralang. Natijani ekranga chiqaring va to'g'ri bo'lganligini tekshiring.