

1. Uyumlar(Heap) va ustuvor navbatlar(Priority Queue)

Uyum (Heap)-bu to'liq ikkilik daraxtiga asoslangan maxsus ma'lumotlar tuzilishi bo'lib hisoblanadi. Ustuvor navbat-bu har bir element ustuvor qiymat bilan bog'langan navbatning maxsus turi va elementlarga ularning ustuvorligi asosida xizmat ko'rsatiladi. Ya'ni, birinchi navbatda yuqori ustuvor elementlarga xizmat ko'rsatiladi.

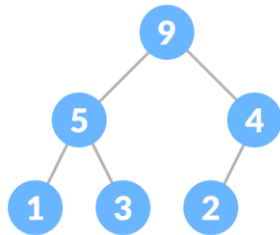
1.1 Uyum ma'lumotlar strukturasi operatsiyalari:

- Heapify:

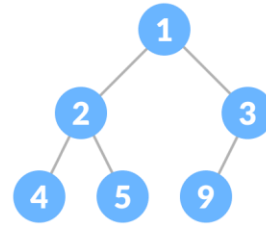
Massivdan uyum yaratish jarayoni. Odatda, uyumlar ikki xil bo'lishi mumkin:

1.Max-Heap (Max-Uyum): Max-Heapda ildiz tugunidagi kalit barcha bolalarda mavjud bo'lgan kalitlar orasida eng katta bo'lishi kerak. Xuddi shu xususiyat ikkilik daraxtdagi barcha pastki daraxtlar uchun rekursiv ravishda to'g'ri bo'lishi kerak.

2.Min-Heap (Min-Uyum): Min-Heapda ildiz tugunidagi kalit barcha vorislari mavjud bo'lgan kalitlar orasida minimal bo'lishi kerak. Xuddi shu xususiyat ikkilik daraxtdagi barcha pastki daraxtlar uchun rekursiv ravishda to'g'ri bo'lishi kerak. Bu operatsiyalarni bajarish $O(\log n)$ vaqtini oladi.



Rasm- 1 Max-Heap

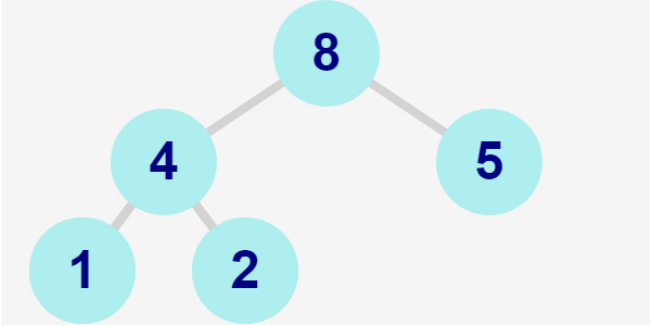
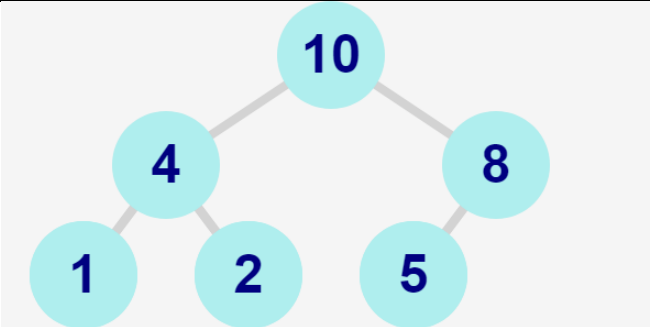


Rasm- 2 Min-Heap

- Kiritish:

Agar biz uyumga yangi element qo'shsak, u uyumning xususiyatlarini buzadi, shuning uchun biz uyumning xususiyatini saqlab qolishi uchun heapify operatsiyasini bajarishimiz kerak. Bu operatsiyalarni bajarish $O(\log N)$ vaqtini oladi.

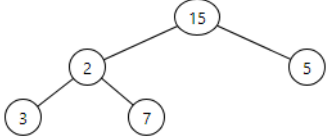
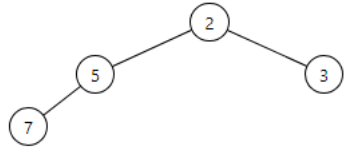
1-Misol:

<p>Faraz qiling dastlabki uyum (olish Max-uyum) quyidagicha</p>	
<p>Endi uyumga 10 ni kiritsak</p>	

- Olib tashlash:

Agar biz elementni uyumdan o'chirsak, u har doim daraxtning ildiz elementini o'chiradi va uni daraxtning oxirgi elementi bilan almashtiradi. Biz ildiz elementini uyumdan o'chirib tashlaganimiz sababli, u uyumning xususiyatlarini buzadi, shuning uchun biz uyum xususiyatini saqlab qolishi uchun heapify operatsiyalarini bajarishimiz kerak. Eng yuqori ustuvor elementi o'chirish, va tartiblash vaqt murakkabligi $O(\log N)$.

2-Misol:

Faraz qiling dastlabki uyum (olish Maks-uyum) quyidagicha	 <pre> graph TD 15((15)) --- 2((2)) 15 --- 5((5)) 2 --- 3((3)) 2 --- 7((7)) </pre>
Endi uyumdan 15 ni olib tashlasak	 <pre> graph TD 2((2)) --- 5((5)) 2 --- 3((3)) 5 --- 7((7)) </pre>

- Peek: Uyumning birinchi (yoki yuqori qismini aytish mumkin) elementini tekshirish yoki topish.

1. getMax (max-heap uchun) yoki getMin (min-heap uchun): mos ravishda maksimal element yoki minimal elementni topadi va biz bilganimizdek minimal va maksimal elementlar har doim min-heap va max-heap uchun ildiz tugunining o'zi bo'ladi. Bu $O(1)$ vaqt murakkabligiga ega.

2. removeMin yoki removeMax: Ushbu operatsiya maksimal element va minimal elementni mos ravishda max-heap va min-heap-dan qaytaradi va o'chiradi.

1.2 Uyum ma'lumotlar tuzilishining implementatsiyasi.

max Heapify-bu Max Heap xususiyatini tiklash uchun mas'ul bo'lgan funktsiya. U tugunni tartibga soladi va uning pastki daraxtlari shunga mos ravishda uyum xususiyati saqlanib qolishi uchun qo'llaniladi.

```

#include <bits/stdc++.h>
using namespace std;
// Max-heap snifi
class MaxHeap {
    //Heap dagi massivning elementlarining ko'rsatkichlari
    int* arr;
    // Max heap ning maximum o'lchami
    int maxSize;

```

```

        // Hozirgi max heap dagi elementlar soni
        int heapSize;
public:
    // funktsiya konstruktsiyasi
    MaxHeap(int maxSize);

    // ichki daraxtlar
    // ildiz sifatida berilgan indeks.
    void MaxHeapify(int);
    // O'ta ona indexini qaytarish
    int parent(int i)
    {
        return (i - 1) / 2;
    }
    // chap voris indexini qaytarish
    int lChild(int i)
    {
        return (2 * i + 1);
    }
    //O'ng voris indexini qaytarish
    int rChild(int i)
    {
        return (2 * i + 2);
    }

    //Ildizni olib tashlash
    int removeMax();

    // Indeks i tomonidan berilgan kalit qiymatini yangi
    qiymatga oshiradi.
    void increaseKey(int i, int newVal);

    // Max heap-dan maksimal kalitni (ildizdagi kalit)
    qaytaradi.
    int getMax()
    {
        return arr[0];
    }

    int curSize()
    {

```

```

        return heapSize;
    }

    // index i dan kalitni o`chiradi
    void deleteKey(int i);

    // Yangi x kalitini kiritish
    void insertKey(int x);
};

// Constructor funktsiyasi berilgan qator a bir to`p quradi []
belgilangan hajmi.
MaxHeap::MaxHeap(int totSize)
{
    heapSize = 0;
    maxSize = totSize;
    arr = new int[totSize];
}

// x yangi kalitini kiritish
void MaxHeap::insertKey(int x)
{
    // Kalitini kiritish mumkin
    //mumkin emasligini tekshirish
    if (heapSize == maxSize) {
        cout << "\nKalitni kiritib bo`lmaydi\n";
        return;
    }
    // yangi kalitni oxiriga joylashtirish
    heapSize++;
    int i = heapSize - 1;
    arr[i] = x;

    // The max heap xususiyatlarini tekshirish
    while (i != 0 && arr[parent(i)] < arr[i]) {
        swap(arr[i], arr[parent(i)]);
        i = parent(i);
    }
}

// kalitni oshirish

```

```

void MaxHeap::increaseKey(int i, int newVal)
{
    arr[i] = newVal;
    while (i != 0 && arr[parent(i)] < arr[i]) {
        swap(arr[i], arr[parent(i)]);
        i = parent(i);
    }
}

// Maksimal uyumning maksimal elementini o'z ichiga olgan
ildiz tugunini olib tashlash uchun.
int MaxHeap::removeMax()
{
    // heap bo'sh yoki yo'qligini tekshirish
    if (heapSize <= 0)
        return INT_MIN;
    if (heapSize == 1) {
        heapSize--;
        return arr[0];
    }
    // maximum elementni o'chirish uchun joylashtirish
    int root = arr[0];
    arr[0] = arr[heapSize - 1];
    heapSize--;
    .
    MaxHeapify(0);

    return root;
}

void MaxHeap::deleteKey(int i)
{
    increaseKey(i, INT_MAX);
    removeMax();
}

//rekursiv heap
void MaxHeap::MaxHeapify(int i)
{
    int l = lChild(i);
    int r = rChild(i);
    int largest = i;

```

```

        if (l < heapSize && arr[l] > arr[i])
            largest = l;
        if (r < heapSize && arr[r] > arr[largest])
            largest = r;
        if (largest != i) {
            swap(arr[i], arr[largest]);
            MaxHeapify(largest);
        }
    }
}

// Bosh funktsiya
int main()
{
    // Uyum maximum o'lchami 15
    MaxHeap h(15);

    // Kalitlarni joylashtirish
    int k, i, n = 6, arr[10];
    cout << "Kalitlar:- 3, 10, 12, 8, 2, 14 \n";
    h.insertKey(3);
    h.insertKey(10);
    h.insertKey(12);
    h.insertKey(8);
    h.insertKey(2);
    h.insertKey(14);

    cout << "Hozirgi uyumning o'lchami: "
        << h.curSize() << "\n";

    cout << "Hozirgi maximum element: " << h.getMax()
        << "\n";

    h.deleteKey(2);

    cout << "Hozirgi uyumning o'lchami: "
        << h.curSize() << "\n";

    h.insertKey(15);
    h.insertKey(5);
    cout << "Hozirgi uyumning o'lchami: "
        << h.curSize() << "\n";
}

```

```
cout << "Hozirgi maximum element: " << h.getMax()  
    << "\n";  
return 0;  
}
```

Dastur natijasi:

```
Kalitlar:- 3, 10, 12, 8, 2, 14  
Hozirgi uyumning o'lchami: 6  
Hozirgi maximum element: 14  
Hozirgi uyumning o'lchami: 5  
Hozirgi uyumning o'lchami: 7  
Hozirgi maximum element: 15
```

Uyumlarning afzalliklari:

- Maksimal/minimal elementga tezkor murojat $O(1)$.
- Samarali kiritish va o'chirish operatsiyalari ($O(\log n)$).
- Moslashuvchan o'lcham.
- Massiv sifatida samarali foydalanish mumkin.
- Haqiqiy ilovalar uchun qulay tanlov.

Uyumlarning kamchiliklari:

- Eng yomon holatda maksimal/minimal dan boshqa elementni qidirish $O(n)$.
- Uyum tuzilishini saqlab qolish uchun qo'shimcha xotira talab qilinadi.
- Ustuvor bo'lmagan navbat operatsiyalari uchun massivlar va bog'langan ro'yxatlar kabi boshqa ma'lumotlar tuzilmalariga qaraganda sekinroq.

Mavzu yuzasidan savollar:

- 1. Uyum bu nima ?
- 2. Max heap va Min heap farqi nimada ?
- 3. Ustuvor navbat bu nima?
- 4. Ustuvor navbat operatsiyalari.
- 5. Ustuvor navbatga elementlarni kiritish.