# TOSHKENT AMALIY FANLAR UNIVERSITETI

# Ma’lumotlar tuzilmasi va algoritmlar fani

“Kompyuter injiniring” kafedrasi
Katta o’qituvchi Kendjayeva Dildora Xudayberganovna

# Ustivor navbatlar.  Binar uyum (kucha) - piramida (binary heap), Uyum (kucha)larni saralash (Heap-Sort)

11 - Ma’ruza

# MA’RUZA REJASI

Ustivor navbatlar.

Binar uyum (kucha) - piramida (binary heap)

Uyum (kucha)larni saralash (Heap-Sort)

# *Ustivor navbatlar*

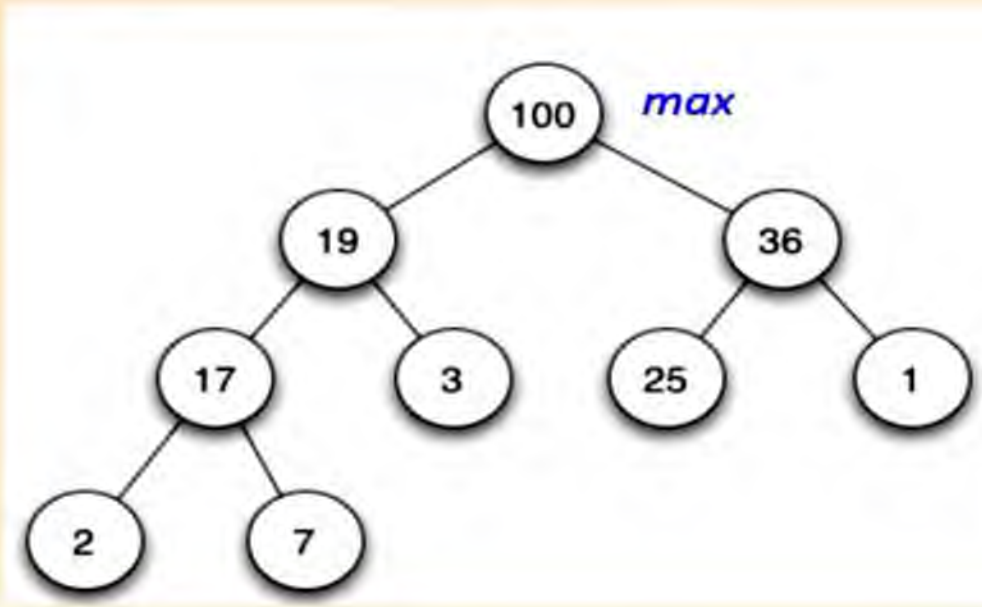
- ***Ustivor navbat*** - bu yozuvlar bir-biri bilan chiziqli taqqoslanadigan kalitlarga (masalan, raqamlar) ega bo'lgan va ikkita amalni realizatsiya qiladigan axborot tizimidir. Bu ikki amal tizimga tasodifiy yozuvni kiritish va yozuv tizimidan eng kichigi bilan tanlov kalit.
- Ustivorda navbatda qo‘llab-quvvatlanadigan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

**Binar uyum (binary heap)** bu quyidagi shartlarni qanoatlantiradigan binar daraxtdir:

- Har qanday uchning ustivorligi, uning avlodlarining ustivorligidan kichik emas.
- Daraxt to'liq ikkilik daraxt bo‘lishi uchun (complete binary tree) - barcha darajalar chapdan o'ngga to'ldiriladi.

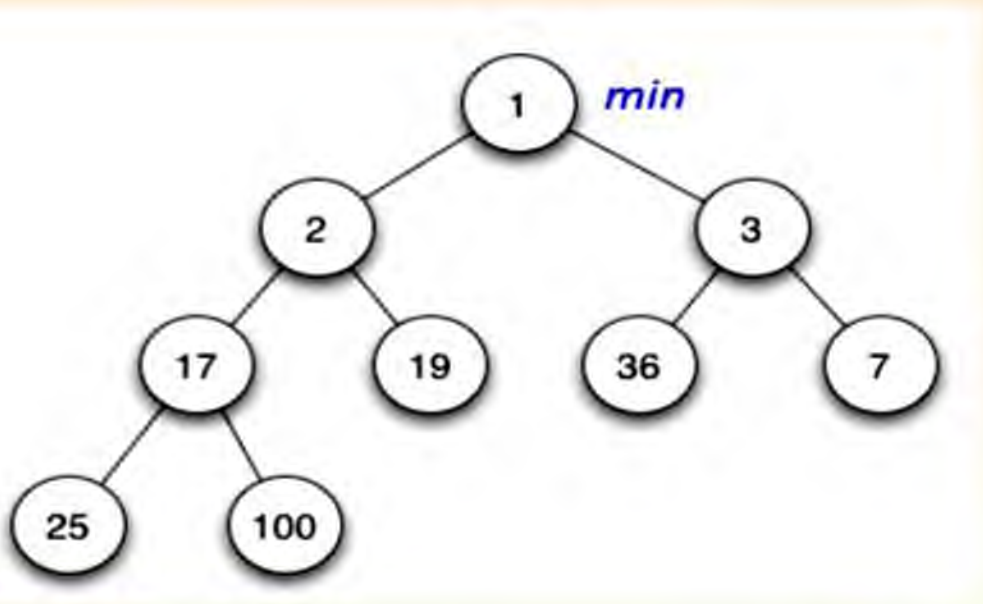
**O’smaydigan piramida**




max-heap

Har qanday uchning ustuvorligi avlodlarning ustuvorligidan kichik emas

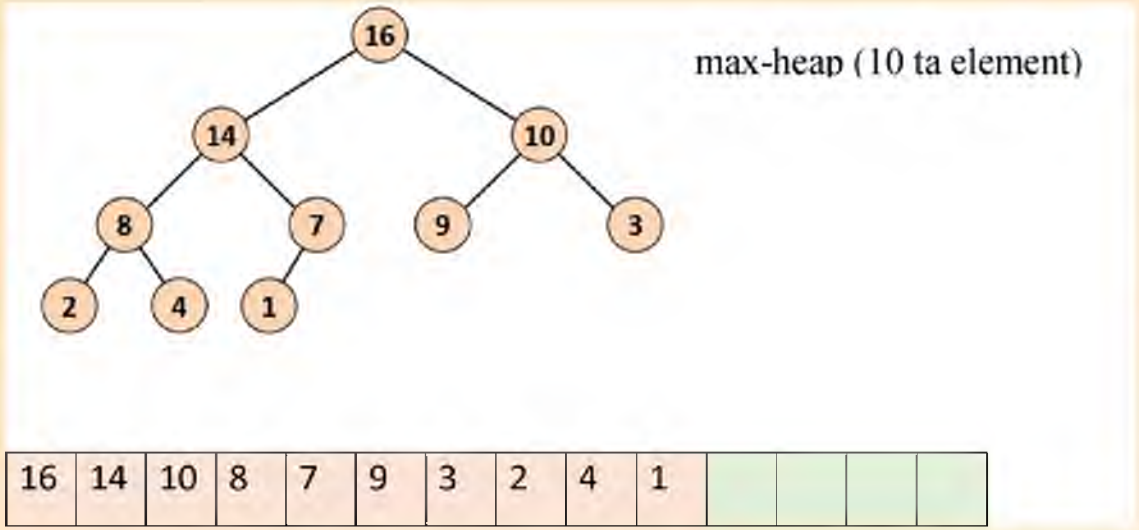
**Kamaymaydigan piramida**




min-heap

Har qanday uchning ustuvorligi avlodlarning ustuvorligidan katta emas

# Massivlar orqali binar uyum (kucha) ni realizatsiya qilish

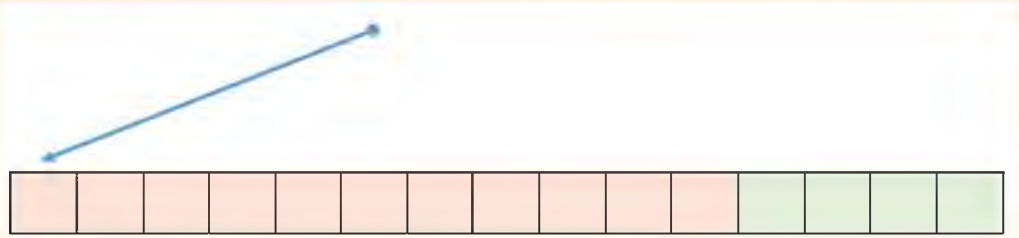



```
struct heapnode {
    int key;          /* kalit */
    char *value;      /* qiymat*/
};

struct heap {
    int maxsize;     /* massiv o‘lchami */
    int nnodes;      /* Kalitlar soni */
    struct heapnode *nodes; /* Nodes: [0..maxsize] */
}
```

**H[1..10] ustuvorliklar (kalitlar) massivi**

Daraxtning ildizi H [1] yacheykada saqlanadi - bu maksimal element;



*i* tugunning ajdod indeksi: *Parent(i)* = [*i/2*]

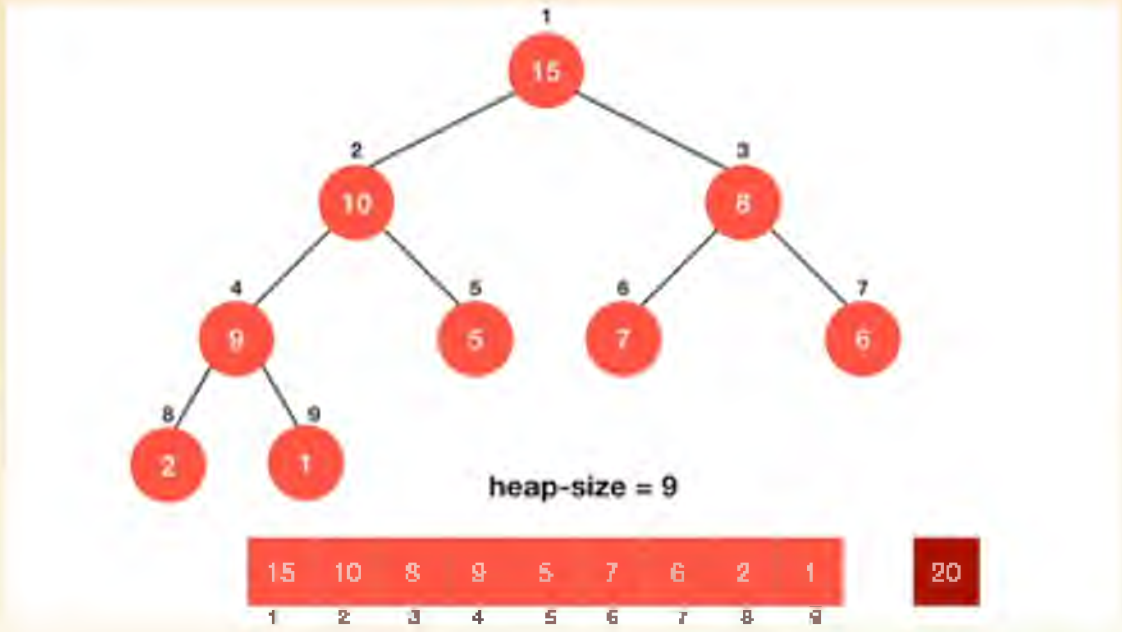
Chap avlod tugun indeksi: *Left ( i) = 2i;*

O‘ng avlod tugun indeksi: *Right(i) = 2 i + 1*;

H [*Parent(i)*] ≥ H [*i*] .

# Bo‘sh uyum (kucha) hosil qilish

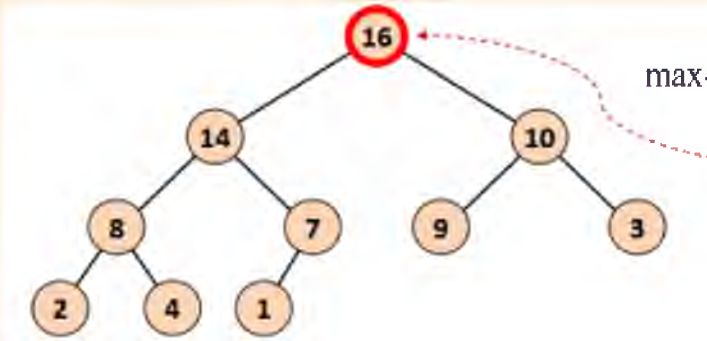
```
struct heap *heap_create(int maxsize)
{
        struct heap *h;
        h = malloc(sizeof(*h));
        if (h != NULL)
        {
        h->maxsize = maxsize;
        h->nnodes = 0;
    /* Heap nodes [0, 1, maxsize] */
        h->nodes = malloc(sizeof(*h->nodes) * (maxsize + 1));
        if (h->nodes == NULL)
        {
        free(h);
        return NULL;
        }
return h;
}
```



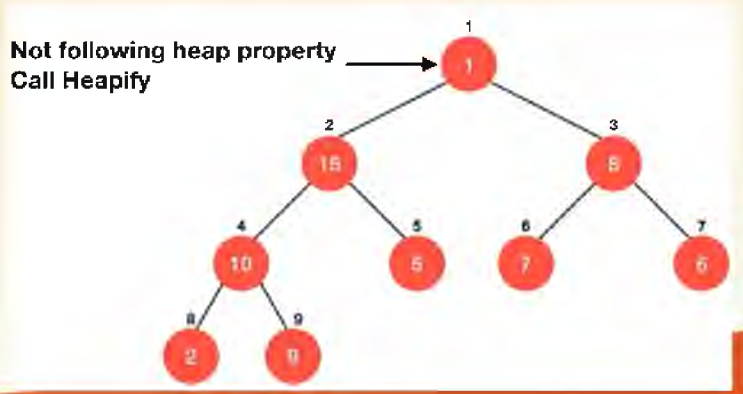
# Uyumni o‘chirish

```
void heap_free(struct heap *h)
{
free(h->nodes);
free(h);
}
void heap_swap(struct heapnode *a, struct heapnode *b)
{
struct heapnode temp;
temp = *a;
*a = *b;
```

```
struct heapnode *heap_max(struct heap *h
{
if (h->nnodes == 0)
    return NULL;
return &h->nodes[1];
}
```

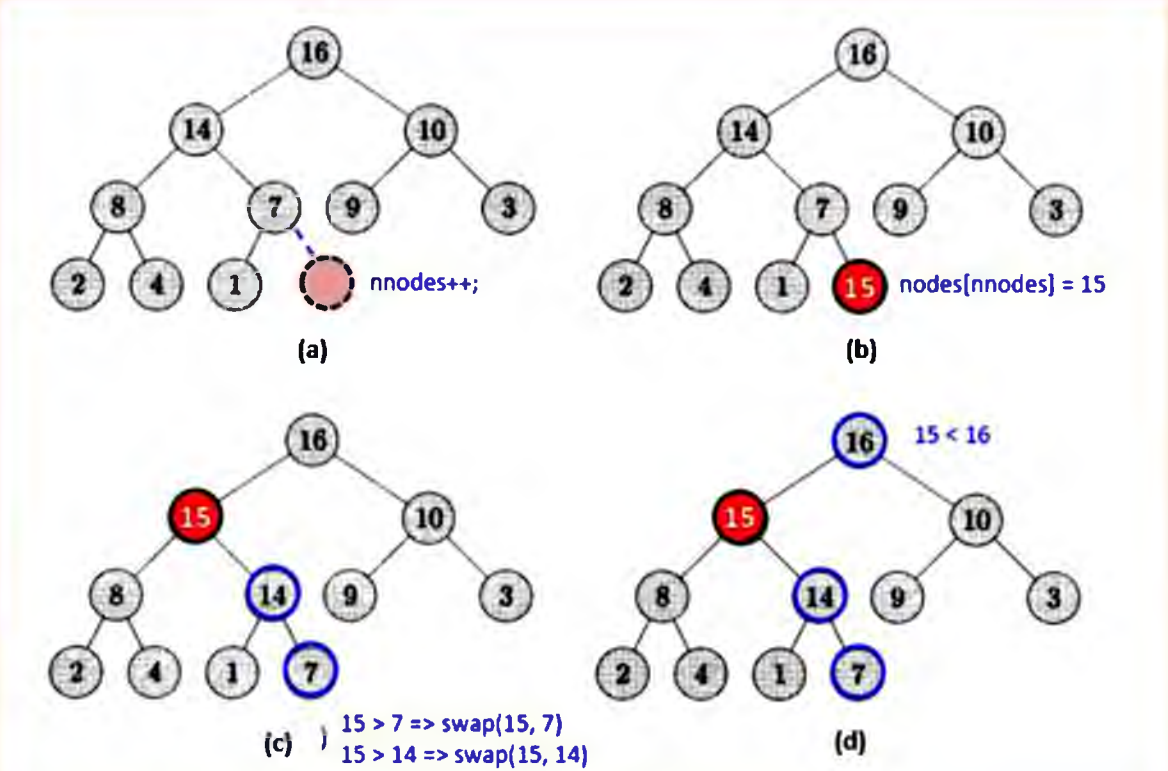



| 16 | 14 | 10 | 8 | 7 | 9 | 3 | 2 | 4 | 1 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|




# Binar uyum (kucha) ga element qo‘shish.

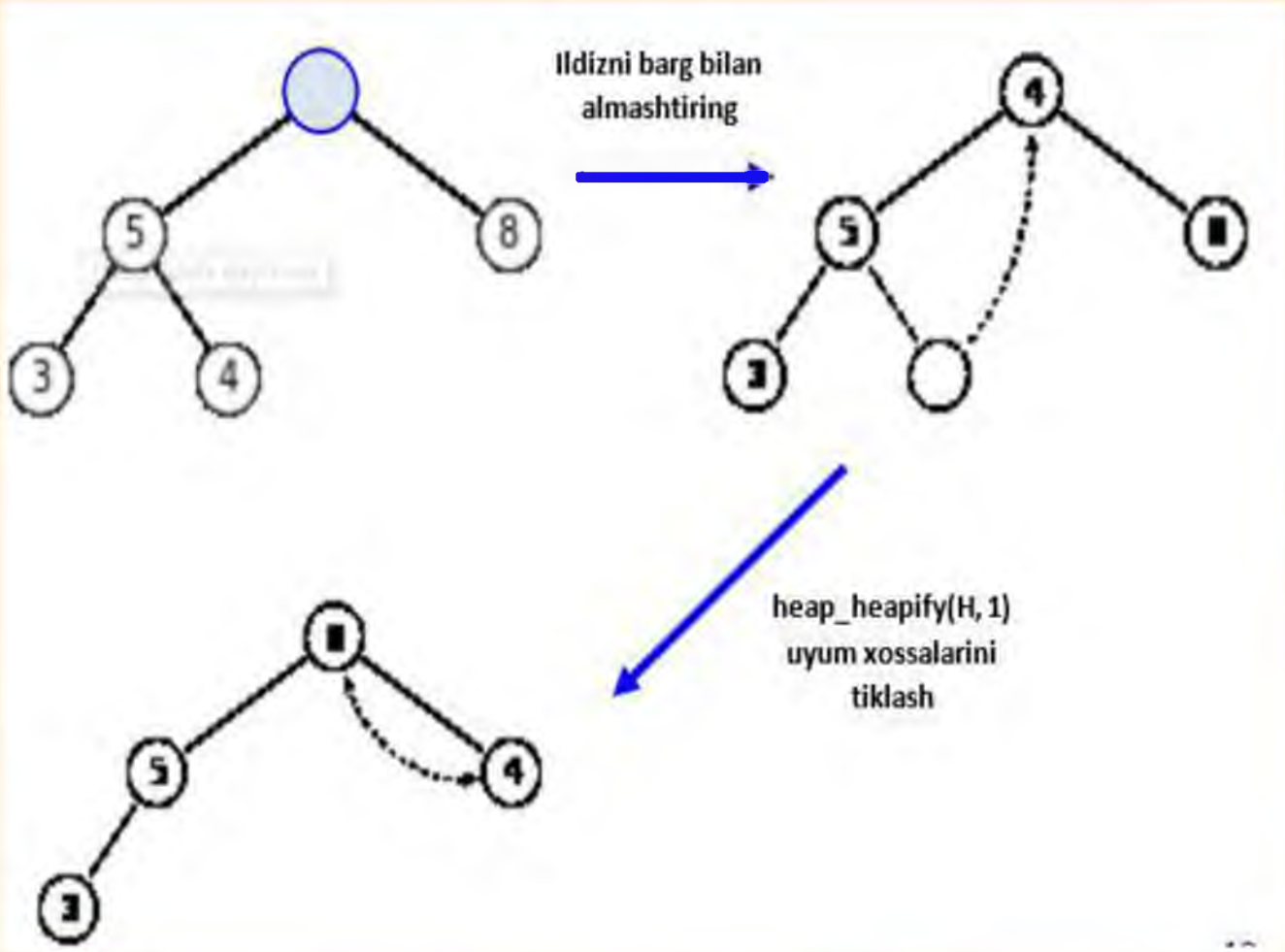
**Ustuvorligi 15 ga teng bo‘lgan elementni joylashtirish**




**Binar kuchaga element joylashtirish**

```
int heap_insert(struct heap *h, int key, char *value)
{
        if (h->nnodes >= h->maxsize) {
return -1;
}
h->nnodes++;
h->nodes[h->nnodes].key = key;
h->nodes[h->nnodes].value = value;
for (int i = h->nnodes; i > 1 &&
        h->nodes[i].key > h->nodes[i / 2].key; i = i/2)
        {
                heap_swap(&h->nodes[i], &h->nodes[i / 2]);
        }
return 0;
```

# Maksimal elementni o‘chirish




```
struct heapnode heap_extract_max(struct heap
{
if (h->nnodes == 0)
return (struct heapnode){0, NULL};
struct heapnode maxnode = h->nodes[l];
h->nodes[l] = h->nodes[h->nnodes];
h->nnodes--;
heap_heapify(h, 1);
return maxnode;
}
```

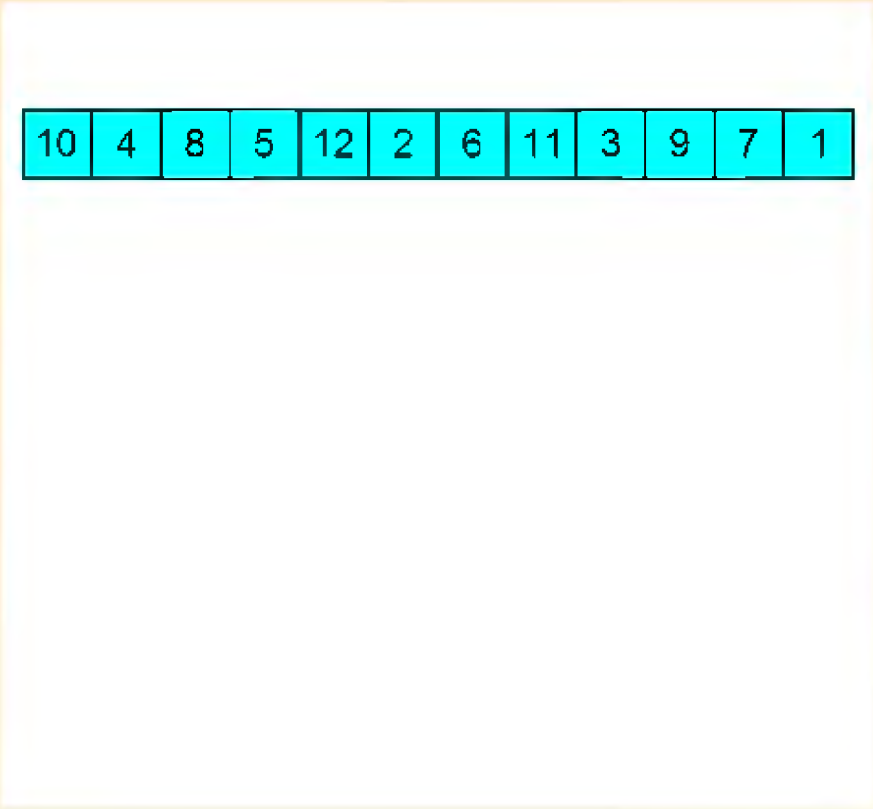
## Uyum xususiyatlarini tiklash

```
void heap_heapify(struct heap *h, int index)
{
    for (;;) {
int left = 2 * index;
int right = 2 * index + 1;
int largest = index;
if (left <= h->nnodes &&
h->nodes[left].key > h->nodes[index].key)
{ largest = left; }
if (right <= h->nnodes && h->nodes[right].key > h->nodes[largest].key)
    { largest = right; }
if (largest == index)
    break;
heap_swap(&h->nodes[index], &h->nodes[largest]);
index = largest;
    }
}
```

## Kalit qiymatini oshirish

```
int heap_increase_key(struct heap *h, int index, int key)
{
if (h->nodes[index].key > key)
    return -l;
h->nodes[index].key = key;
for ( ; index > 1 && h->nodes[index].key > h->nodes[index / 2].key; index = index
/ 2)
{
        heap_swap(&h->nodes[index], &h->nodes[index / 2]);
}
return index;
}
```

# *Uyum (kucha)larni saralash (Heap-Sort)*




- **Heapsort (Heapsort, "Heap sorting")** - n elementlarni saralashda **O(nlogn)** amallarda eng yomon, o'rtacha va eng yaxshi holda ishlaydigan saralash algoritmi. Ishlatiladigan qo'shimcha xotira miqdori massiv kattaligiga bog‘liq emas **(ya'ni O (1)).**
- Ushbu saralashni pufaksimon saralashning rivojlantirilgan ko‘rinishi deb qarash mumkin.
- Eng yomon vaqt - O(nlog(n))
- Eng yaxshi vaqt - O(nlog(n))
- O‘rtacha vaqt - O(nlog(n))

# *Mavzu yuzasidan savollar:*

1. Ustivor navbat nima?
2. Heap-Sort algoritmi haqida gapiring
3. Uyum tushunchasi.
4. Binar kucha bilan ishlash?
5. Ustivor navbat ma’lumotlar strukturasi qo‘llaniladigan sohalarga qaysilar kiradi?

*Do you have any questions?*