

Пирамида(Неар)

Лекция 7 по СДА, Софтуерно Инженерство
Зимен семестър 2018-2019г
д-р Милен Чечев

Информация за Контролно 3

- Няма да влиза в оценяването
- Всеки изкарал точки, ще му се зачитат като бонус точки
- Задачата остана за домашна работа.

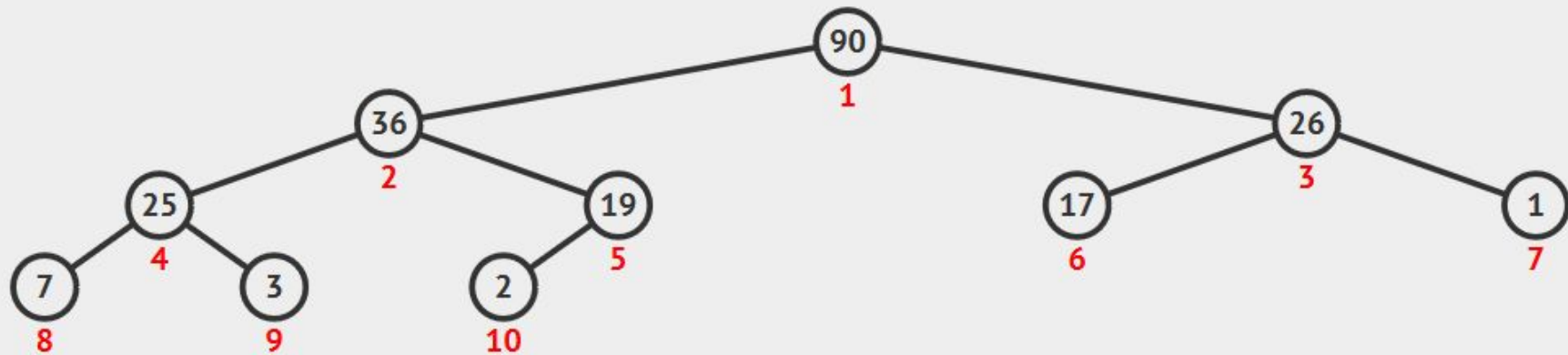
Подобрения:

- Възможност да се ползва IDE за следващото контролно
- Повече време за контролното
- Допълнителни консултации
- Писане на код на живо (експериментално)

План за днес

1. Heap.Priority Queue
2. Tutorial/Live coding на задачи за линейни и дървовидни структури

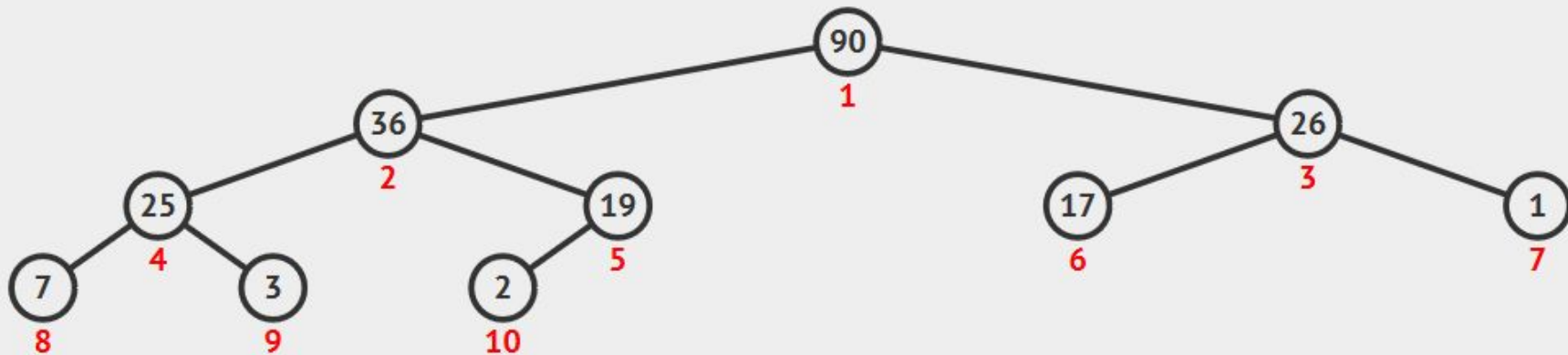
Пирамида



Какво е пирамида?

Когато говорим за пирамида ще имаме в предвид двоична пирамида (въпреки, че има и други).

- Пирамида е почти пълно двоично дърво, като само най-дълбокото ниво не е пълно и то последователно (виж картинката)
- За пирамидата имаме свойството, че за всеки възел бащата е по-голям и от двете си деца (при максимална пирамида).



Операции

getMax(X)

extractMax(X)

insert(X)

delete(X)

<https://visualgo.net/en/heap>

Основни приложения

- Приоритетна опашка
- Пирамидално сортиране (HeapSort)

Приоритетна опашка

Опашка, в която се вкарват последователно елементи, а при изкарване елементите ги получаваме наредени по приоритет (например големина).

Ако всички са с еднакъв приоритет ще работи както обикновена опашка

Реализации на пирамида

- Свързано представяне стандартно за двоично дърво
- Представяне с масив (space efficient!)

Source: <https://www.geeksforgeeks.org/binary-heap/>

```
class MinHeap
{
    int *harr; // pointer to array of elements in heap
    int capacity; // maximum possible size of min heap
    int heap_size; // Current number of elements in min heap

    int parent(int i) { return (i-1)/2; }
    int left(int i) { return (2*i + 1); }
    int right(int i) { return (2*i + 2); }
```

```
int MinHeap::extractMin(){
    if (heap_size <= 0)
        return INT_MAX;
    if (heap_size == 1){
        heap_size--;
        return harr[0];
    }
    int root = harr[0];
    harr[0] = harr[heap_size-1];
    heap_size--;
    MinHeapify(0);

    return root;
}
```

```
void MinHeap::insertKey(int k){
    if (heap_size == capacity){
        cout << "\nOverflow: Could not insertKey\n";
        return;
    }
```

```
// First insert the new key at the end
```

```
heap_size++;
```

```
int i = heap_size - 1;
```

```
harr[i] = k;
```

```
// Fix the min heap property if it is violated
```

```
while (i != 0 && harr[parent(i)] > harr[i])
```

```
{
```

```
    swap(&harr[i], &harr[parent(i)]);
```

```
    i = parent(i);
```

```
}
```

```
}
```

Представяне с масив

- Удобно поради последователното запълване на пирамидата
- Корена е на позиция 0, а наследниците му са на позиция 1 и 2
- За всяка позиция от масива i наследниците са $2*i+1$ и $2*i+2$ (ако съществуват).

Пирамидално сортиране

Основна идея: Преобразува масива в пирамида, след което вади един по един елементите от върха на пирамидата и ги поставя сортирани последователно в края на масива.

Сортиране със сложност $O(n \log(n))$ и константна допълнителна памет!

Пирамидално сортиране

```
void heapSort(int arr[], int n)
{
    // Build heap (rearrange array)
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // One by one extract an element from heap
    for (int i=n-1; i>=0; i--)
    {
        // Move current root to end
        swap(arr[0], arr[i]);

        // call max heapify on the reduced heap
        heapify(arr, i, 0);
    }
}
```

```
void heapify(int arr[], int n, int i)
{
    int largest = i; // Initialize largest as root
    int l = 2*i + 1; // left = 2*i + 1
    int r = 2*i + 2; // right = 2*i + 2

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

        // Recursively heapify the affected sub-tree
        heapify(arr, n, largest);
    }
}
```


Следва решаване на задачи (Live coding)

Контролно 3:

<https://www.hackerrank.com/contests/sdatest3-hw/challenges/test3-zad1/problem>

Задачи за дървета:

<https://www.hackerrank.com/challenges/tree-preorder-traversal/problem>

<https://www.hackerrank.com/challenges/tree-height-of-a-binary-tree/problem> ?

<https://www.hackerrank.com/challenges/tree-top-view/problem> ?

<https://www.hackerrank.com/challenges/tree-level-order-traversal/problem>

Допълнително време за консултации

До края на семестъра зала 210 ще е запазена за допълнителни консултации и контролни по СДА. Носете си лаптоп ако искате да пишете/показвате код.