

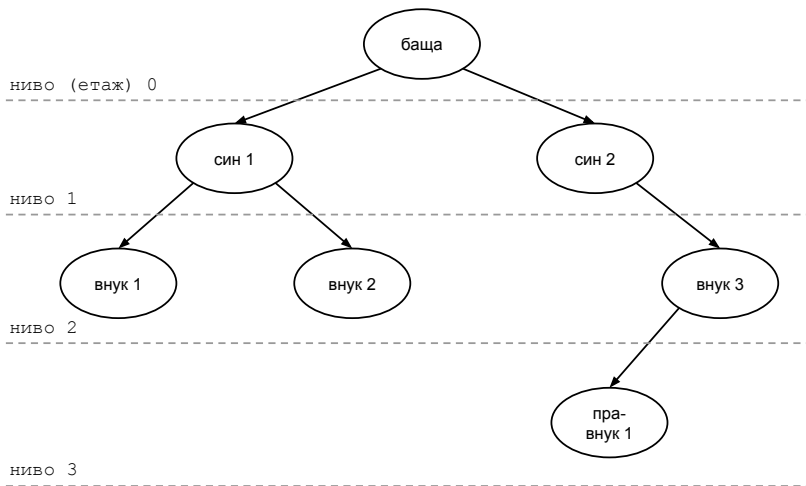
# Дървета

Калин Георгиев

20 октомври 2016 г.

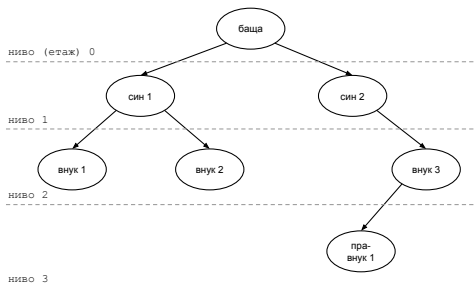
## Интуиция. Йерархична структура от данни

## Фамилно дърво

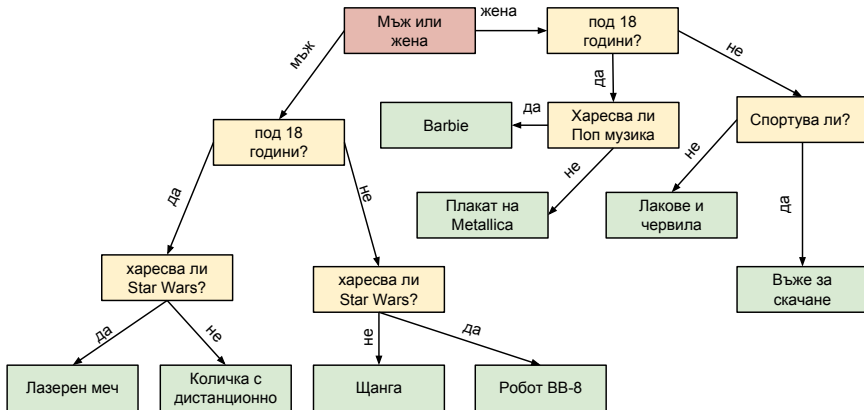


# Фамилно дърво

- родител, наследник
- корен
- ЛИСТО
- ПЪТ
- НИВО



## Decision Tree



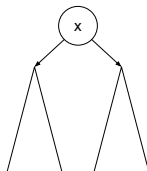
## Формална дефиниция и абстракция

# Дефиниции

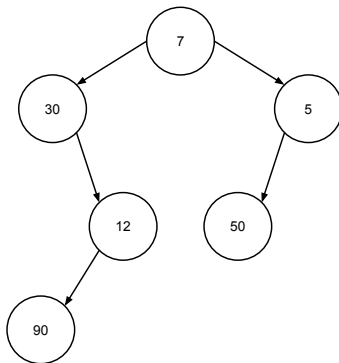
- Свързан неориентиран граф без цикли

## Индуктивна дефиниция

- Фиксираме елемент  $(x) \notin D$  и го наричаме "празно дърво"
- Празното дърво е дърво
- Ако  $L_T$  и  $R_T$  са дървета, а  $x$  е елемент ( $x \in D$ ), то *тройката (структурата)*  $T = (x, L_T, R_T)$  наричаме двоично дърво  $T$  с *корен*  $x$ , ляво поддърво  $L_T$  и дясно поддърво  $R_T$ .

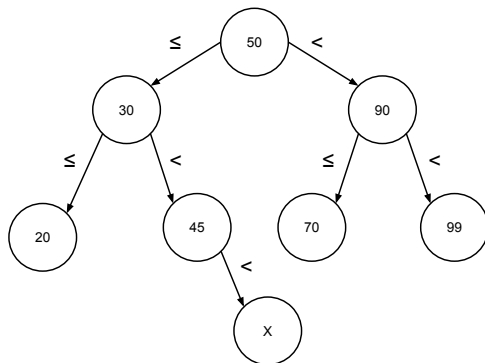


# Дърво с числа





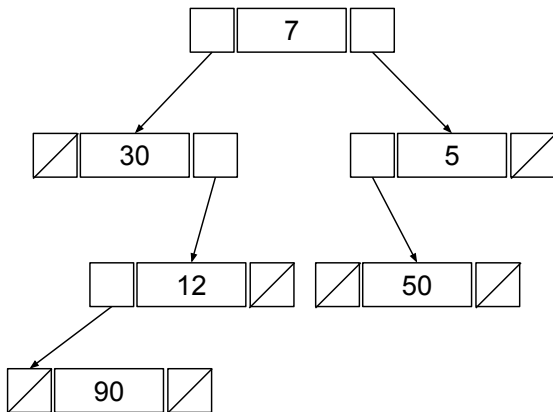
# Двоично наредено дърво



50	70	20	45	99	70	30
20	30	45	50	70	90	99

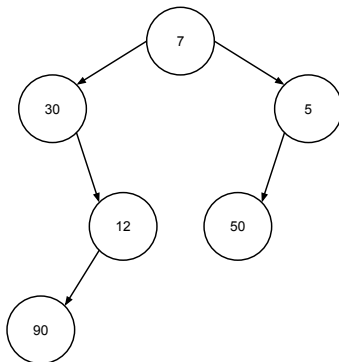
# Представяне

# “Тройна кутия”

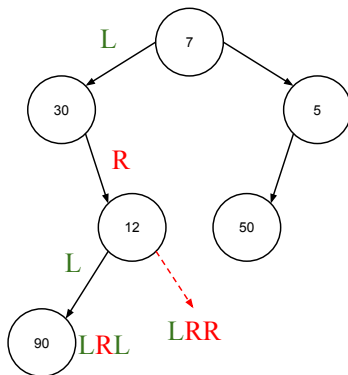


# Операции

## Добавяне на елемент

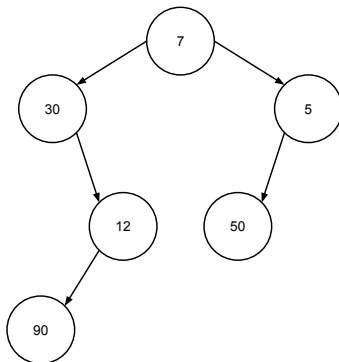


## Следа. Добавяне и намиране на елемент



## Рекурсивни операции

## Търсене на елемент

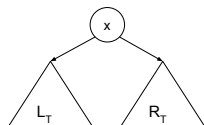




# Проверка за принадлежност

- Среца ли се елементът  $y$  сред елементите на дървото  $T$ ?

- Празното дърво е дърво
- Ако  $L_T$  и  $R_T$  са дървета, а  $x$  е елемент ( $x \in D$ ), то *тройката* (структурата)  $T = (x, L_T, R_T)$  наричаме двоично дърво  $T$  с корен  $x$ , ляво поддърво  $L_T$  и дясно поддърво  $R_T$ .



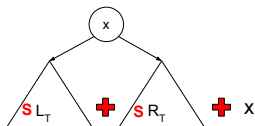
`bool member (int y, T):`

- Ако дървото е празно, то  $y$  не е елемент на дървото. `return false`
- Ако  $T = (x, L_T, R_T)$  е непразно, то  $y$  е елемент на  $T$ , ако  $y == x$  или  $y$  е елемент на  $L_T$  или  $y$  е елемент на  $R_T$ .
- `return y==root(t) || member (y, LT) || member (y,RT)`

## Сума на елементите

- Каква е сумата на елементите на дървото?

- Празното дърво е дърво
- Ако  $L_T$  и  $R_T$  са дървета, а  $x$  е елемент ( $x \in D$ ), то тройката (структурата)  $T = (x, L_T, R_T)$  наричаме двоично дърво  $T$  с корен  $x$ , ляво поддърво  $L_T$  и дясно поддърво  $R_T$ .

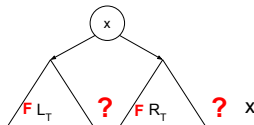


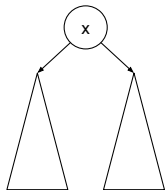
int sum (T):

- Ако дървото е празно, то сумата на елементите е 0. return 0
- Ако  $T = (x, L_T, R_T)$  е непразно, то сумата на елементите е сбора на сумата на елементите на  $L_T$ , на сумата на елементите на  $R_T$  и на числото  $x$ .
- return root(t) + sum (LT) + sum (RT)

## Други прости рекурсивни операции

- Най-голям елемент в дървото
  - Брой на елементите в дървото
  - Височина на дървото
  - Брой на листата в дървото
- Празното дърво е дърво
  - Ако  $L_T$  и  $R_T$  са дървета, а  $x$  е елемент ( $x \in D$ ), то *тройката (структурата)*  $T = (x, L_T, R_T)$  наричаме двоично дърво  $T$  с корен  $x$ , ляво поддърво  $L_T$  и дясно поддърво  $R_T$ .

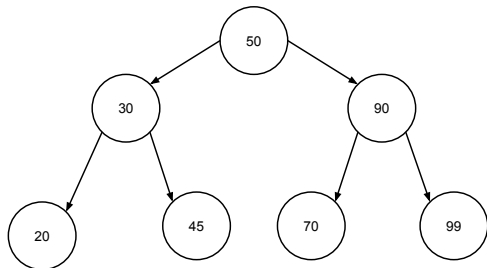




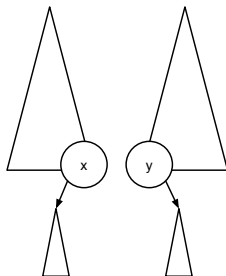
Сериализация

## Операции с двоични наредени дървета

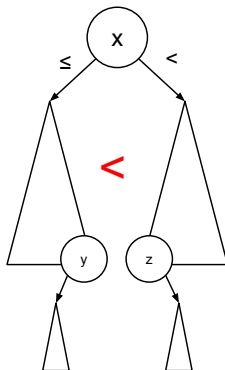
## Вмъкване



## Намиране на най-голям и най-малък елемент

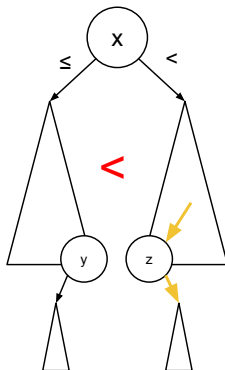


## Изтриване

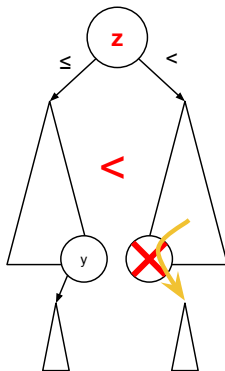




## Изтриване



## Изтриване



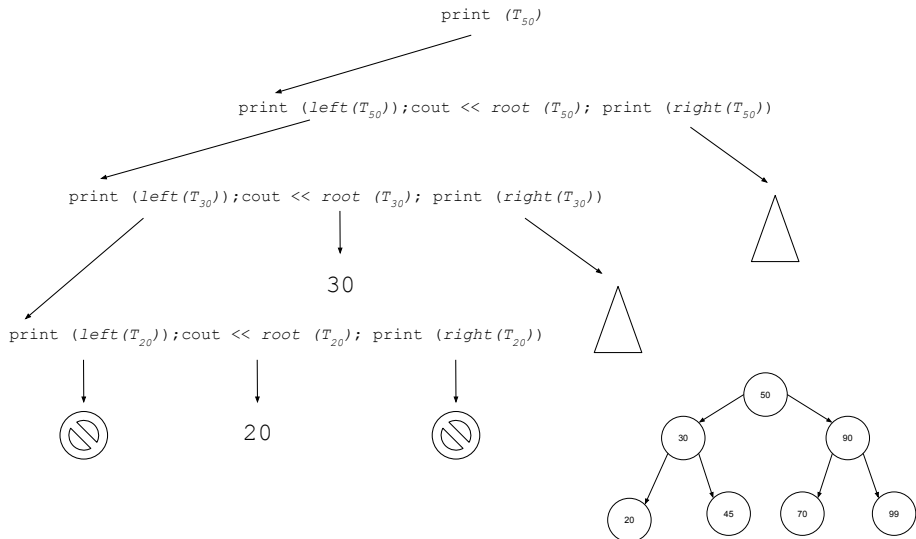
Обходане с рекурсия VS. обходане чрез стек

# Отпечатване

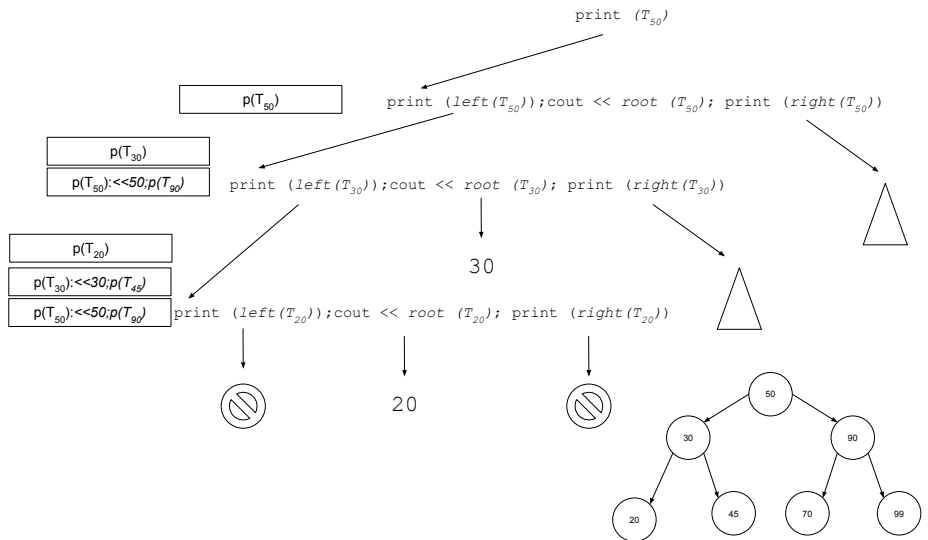
- Отпечатване с рекурсия

```
void print (t){  
    if (empty (t))  
        return;  
    print (left (t));  
    cout << root (t);  
    print (right (t));  
}
```

# Процес на отпечатване



# Процес на отпечатване



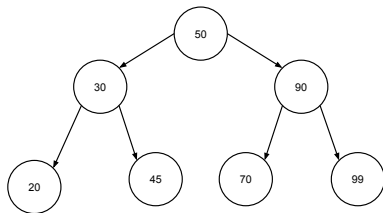
# Отпечатване чрез стек

- Отпечатване с рекурсия

```
void print (t){  
    if (empty (t))  
        return;  
    print (left (t));  
    cout << root (t);  
    print (right (t));  
}
```

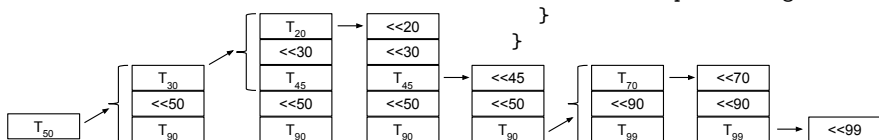
```
void print (t){  
    stack.push (t);  
    while (!empty (stack)){  
        x = stack.pop();  
        if (x is a number)  
            cout << x;  
        else {  
            stack.push (left(x));  
            stack.push (root(x));  
            stack.push (right(x));}  
    }  
}
```

## Отпечатване чрез стек



```

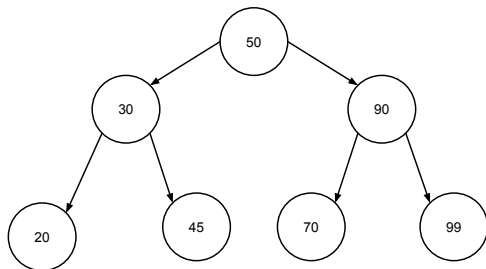
void print (t){
    stack.push (t);
    while (!empty (stack)){
        x = stack.pop();
        if (x is a number)
            cout << x;
        else {
            stack.push (left(x));
            stack.push (root(x));
            stack.push (right(x));}
    }
}
  
```





Обходане с с опашка

## Обхождане с опашка



Въпроси?