TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Programmeerimise süvendatud algkursus ITI0140

2015

Teemad

- Pinu (ingl *stack*)
- Rekursioon (ingl *recursion*)
- Sügavuti otsing (ingl *depth-first search*)

Pinu

Pinu on selline andmestruktuur, kuhu elemente lisatakse ja eemaldatakse ühest otsast, n-ö pinu tipust (ingl *top*).

Pinu operatsioonideks on meil: elemendi lisamine (*push*), elemendi eemaldamine (*pop*), oleku kontroll (kas pinu on tühi?)

Pinu võib implementeerida Pythonis näiteks järjendit kasutades.

Pinu näide

Pythonis ei ole järjendil funktsiooni push, vaid append

```
stack = []
stack.append("first") # "push" operation
print(stack)
stack.append(123) # "push" operation
print(stack)
element = stack.pop() # "pop" operation
print(element)
print(stack)
element = stack.pop() # "pop" operation
print(False if stack else True) # "empty" check
element = stack.pop() # "pop" operation
```

```
>> ['first']
['first', 123]
123
['first']
True
Traceback (most recent call last):
  File "stack.py", line 12, in <module>
    element = stack.pop() # "pop" operation
IndexError: pop from empty list
```

N-ö “mitte-pythonlik stiil” oleks

```
print(True if len(stack) == 0 else False)
```

Rekursioon

Rekursioon on viis lahendada ülesandeid muutes neid aina lihtsamaks, jõudes lõpuks baasjuhuni.

Baasjuhu ja vahepeal läbitud sammude tulemustest kombineeritakse esialgse ülesande lahendus.

Rekursioon programmeerimises tähendab funktsiooni iseenda väljakutsumist muudetud argumentidega.

Rekursioon

```
def rinse():
    print("rinsing...")

def lather():
    print("lathering...")

def repeat():
    rinse()
    lather()
    repeat()

print(repeat())
```

```
rinsing...
lathering...
rinsing...
lathering...
rinsing...
lathering...
rinsing...
lathering...
   ....
```

Kutsutakse välja iseennast!

Rekursioon

```
def rinse():
    print("rinsing...")

def lather():
    print("lathering...")

def repeat():
    rinse()
    lather()
    repeat()

print(repeat())
```

```
rinsing...
lathering...
rinsing...
lathering...
rinsing...
lathering...
rinsing...
lathering...
   ....
```

```
File "recursion.py", line 10, in repeat
  repeat()
File "recursion.py", line 8, in repeat
  rinse()
File "recursion.py.py", line 2, in rinse
  print("rinsing...")
RuntimeError: maximum recursion depth
exceeded while calling a Python object
```

Rekursiooni näide

Klassikalised rekursiooni näited on matemaatikast tuntud faktoriaal ja Fibonacci arvujada.

Faktoriaal: 6! = 1*2*3*4*5*6 = 720

Fibonacci jada: f(0) = 0, f(1) = 1, **f(x)=f(x-1) + f(x-2)**

nt. f(4) = f(3) + f(2) =
= f(2) + f(1) + f(1) + f(0) =
= f(1) + f(0) + f(1) + f(1) + f(0) =
= 1 + 0 + 1 + 1 + 0 = 3

Rekursiooni näide

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(6))

>> 720
```

Baasjuht (0! = 1)

Rekursioon

Rekursiooni näide

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(6))
```

return 6 * factorial(5)

```
>> 720
```

Rekursiooni näide

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(6))
```

```
>> 720
```

return 6 * factorial(5)
return 6 * 5 * factorial(4)

Rekursiooni näide

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(6))
```

```
>> 720
```

return 6 * factorial(5)
return 6 * 5 * factorial(4)
return 6 * 5 * 4 * factorial(3)

Rekursiooni näide

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(6))
```

```
>> 720
```

return 6 * factorial(5)
return 6 * 5 * factorial(4)
return 6 * 5 * 4 * factorial(3)
return 6 * 5 * 4 * 3 * factorial(2)

Rekursiooni näide

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(6))
```

```
>> 720
```

return 6 * factorial(5)
return 6 * 5 * factorial(4)
return 6 * 5 * 4 * factorial(3)
return 6 * 5 * 4 * 3 * factorial(2)
return 6 * 5 * 4 * 3 * 2 * factorial(1)

Rekursiooni näide

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(6))
```

```
>> 720
```

return 6 * factorial(5)
return 6 * 5 * factorial(4)
return 6 * 5 * 4 * factorial(3)
return 6 * 5 * 4 * 3 * factorial(2)
return 6 * 5 * 4 * 3 * 2 * factorial(1)
return 6 * 5 * 4 * 3 * 2 * 1 * factorial(0)

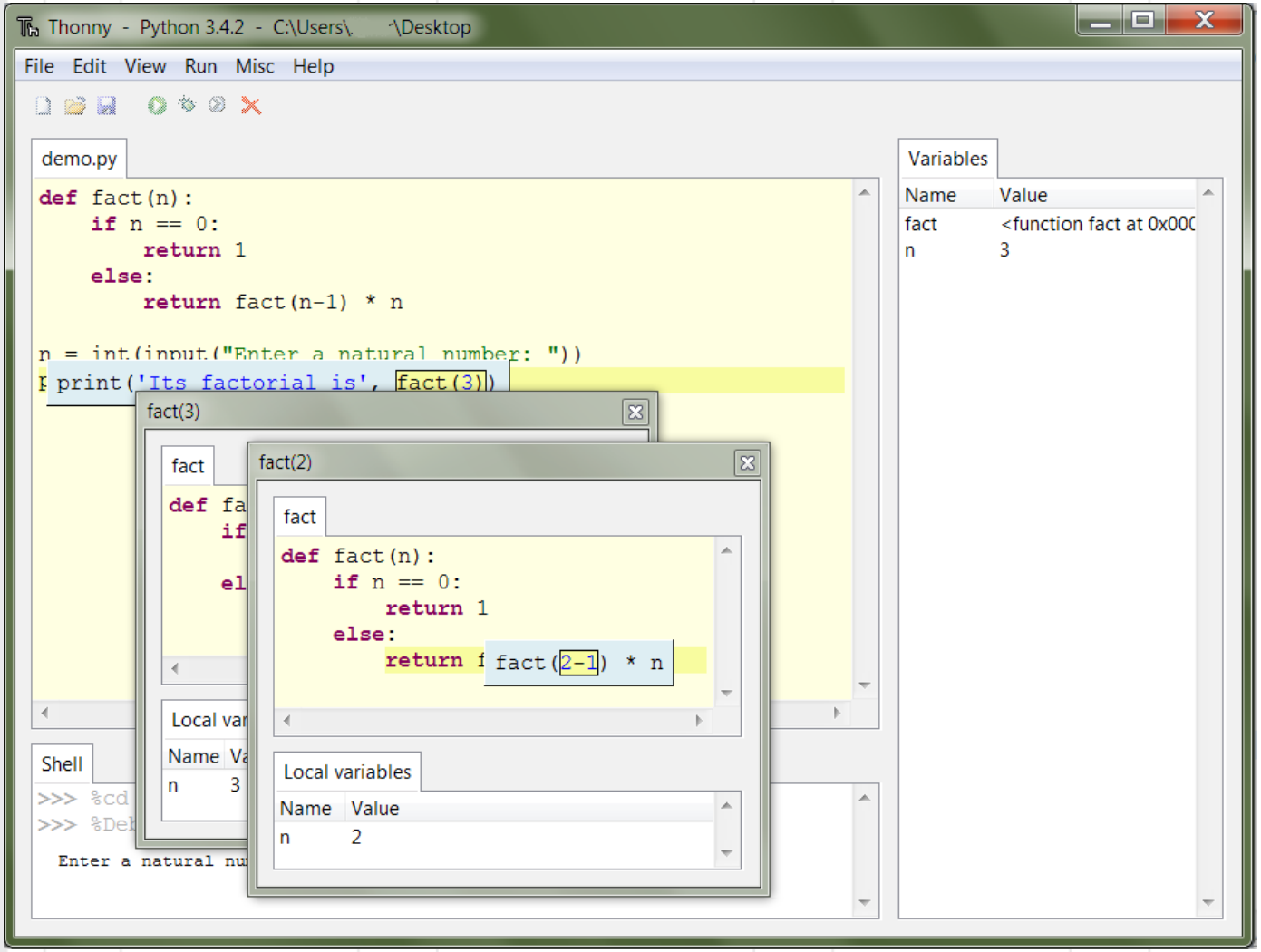
Rekursiooni näide

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(6))
```

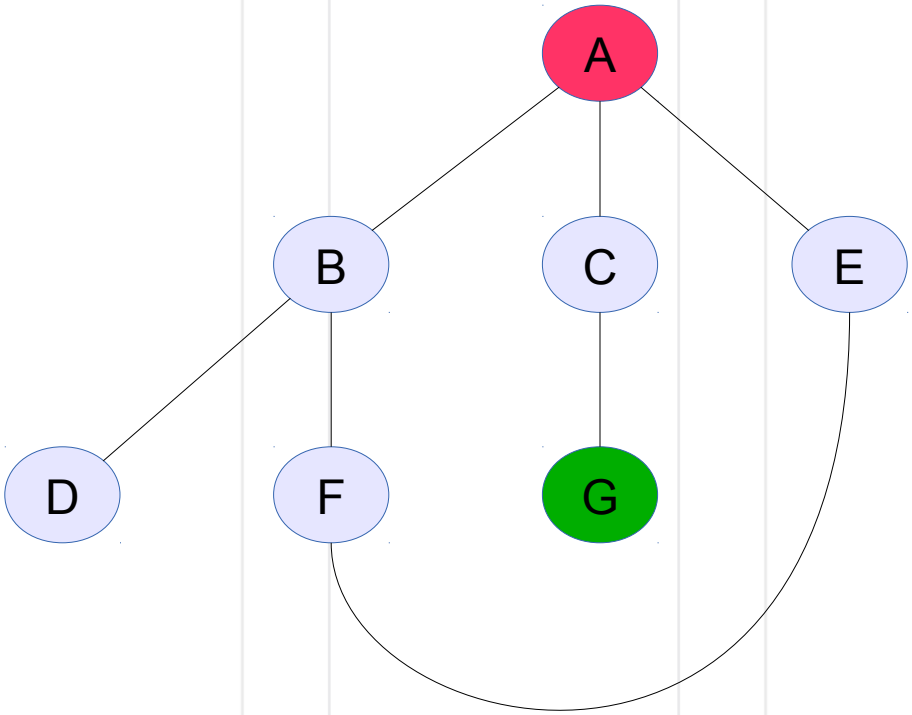
```
>> 720
```

return 6 * factorial(5)
return 6 * 5 * factorial(4)
return 6 * 5 * 4 * factorial(3)
return 6 * 5 * 4 * 3 * factorial(2)
return 6 * 5 * 4 * 3 * 2 * factorial(1)
return 6 * 5 * 4 * 3 * 2 * 1 * factorial(0)
return 6 * 5 * 4 * 3 * 2 * 1 * 1

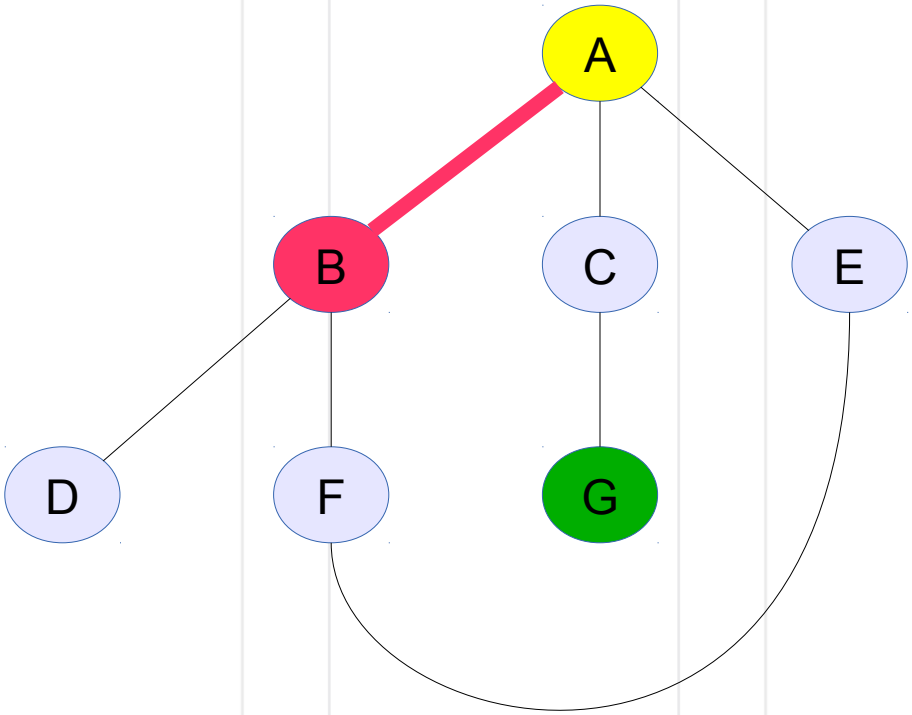
http://thonny.cs.ut.ee/



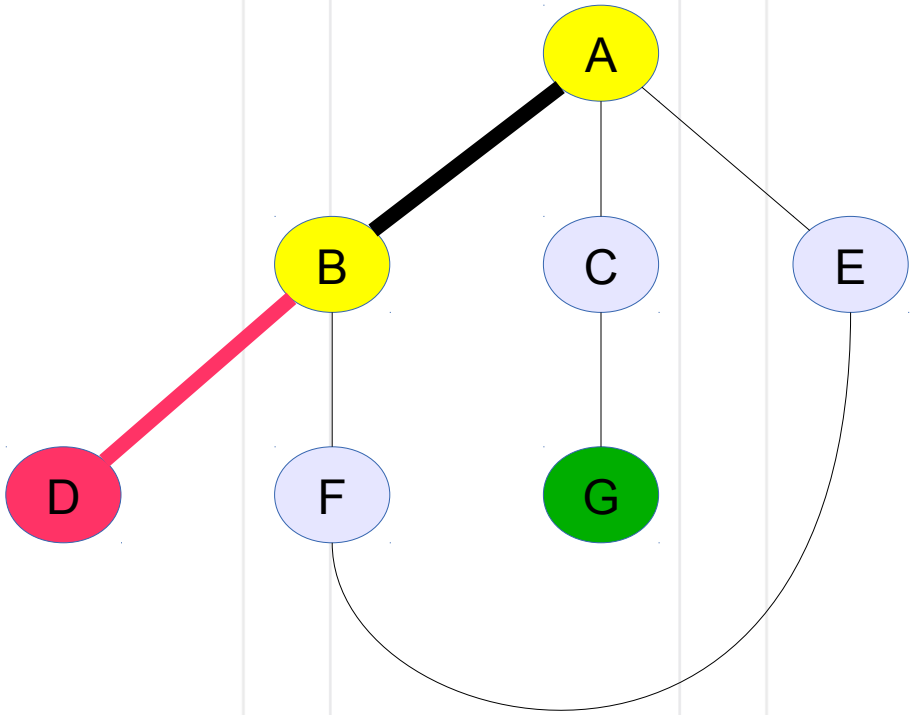
Sügavuti otsing



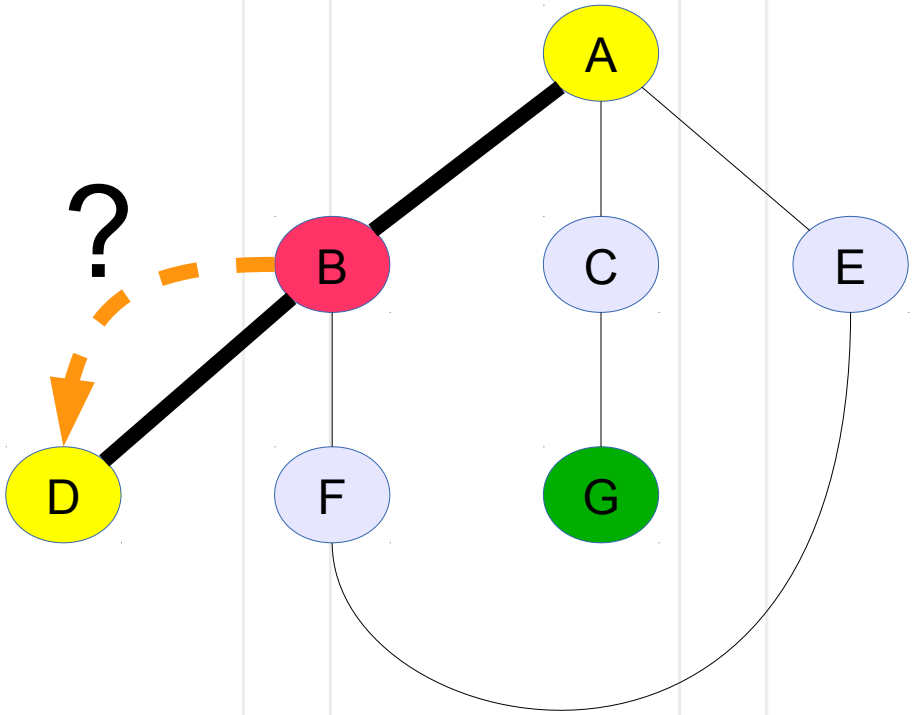
Sügavuti otsing



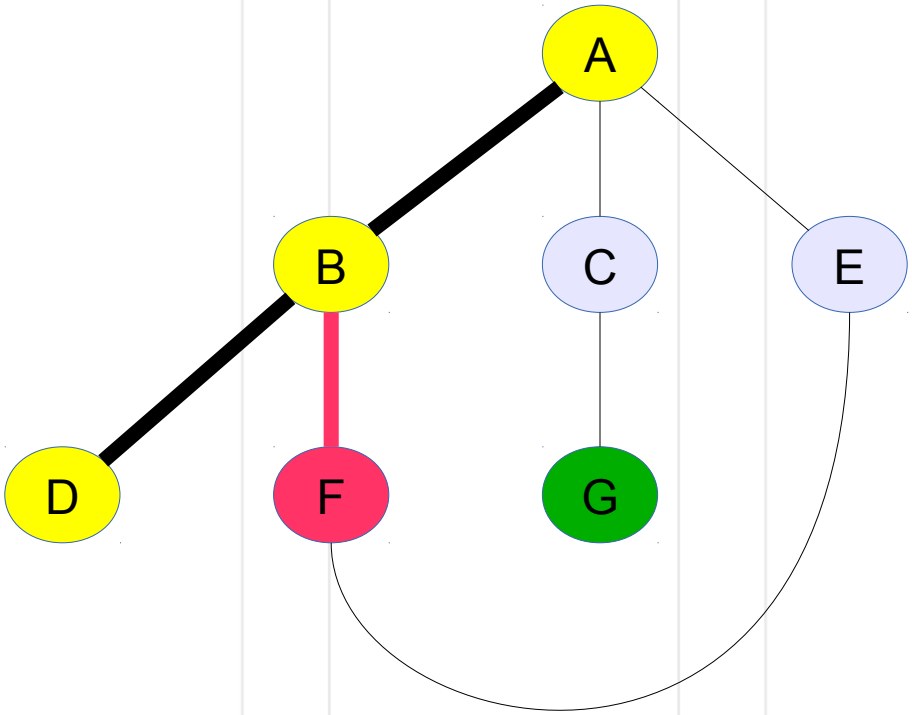
Sügavuti otsing



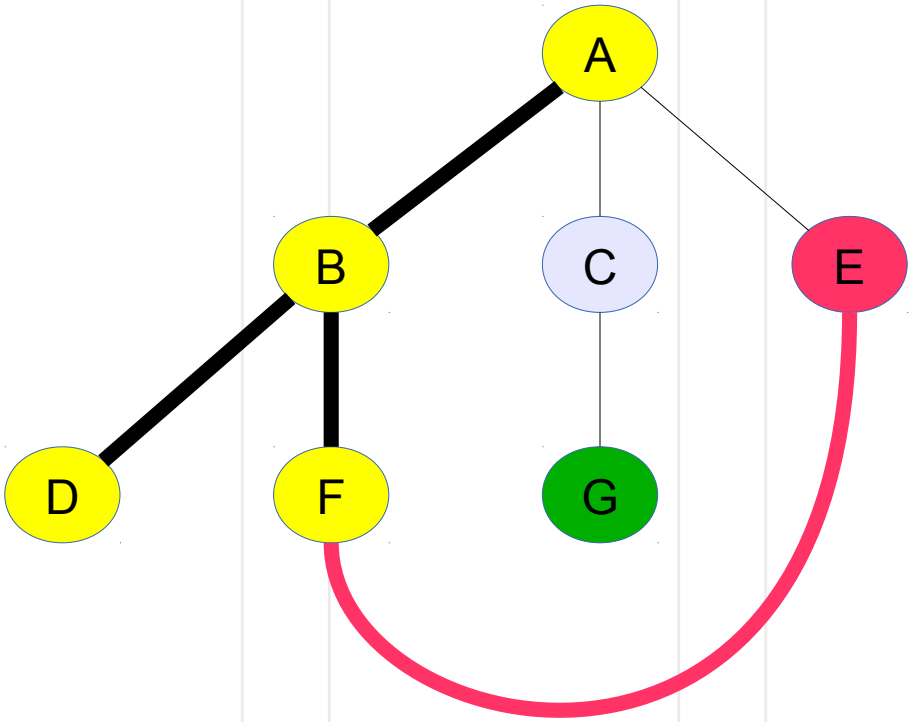
Sügavuti otsing



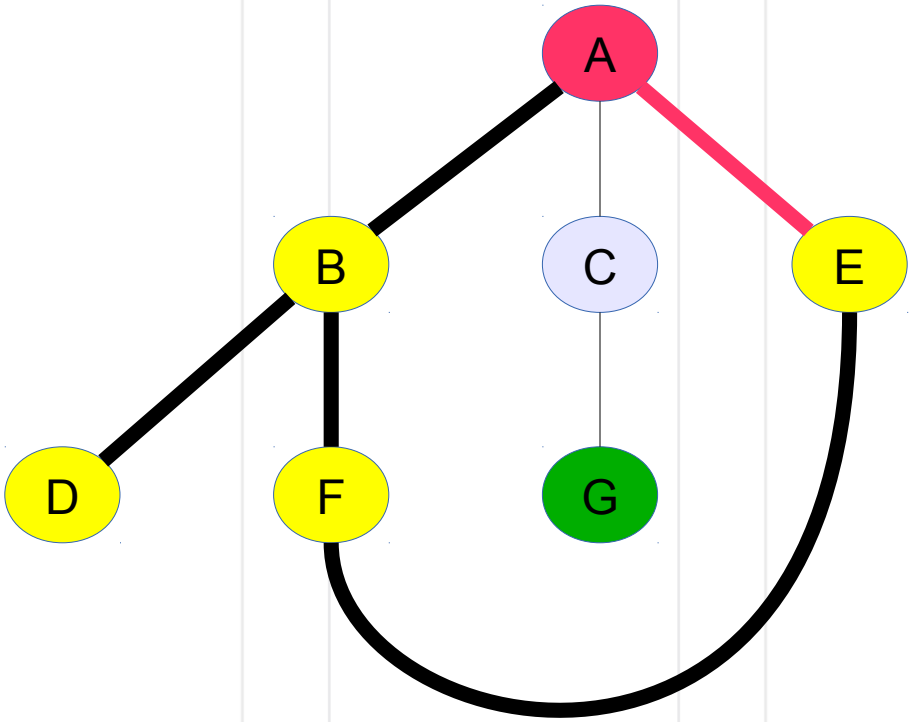
Sügavuti otsing



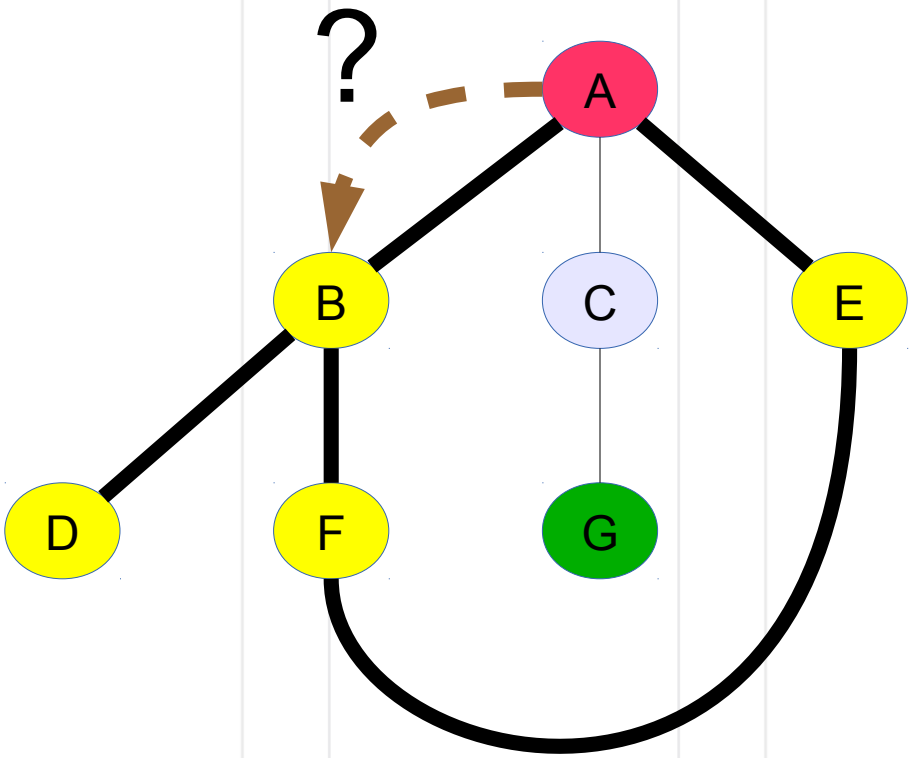
Sügavuti otsing



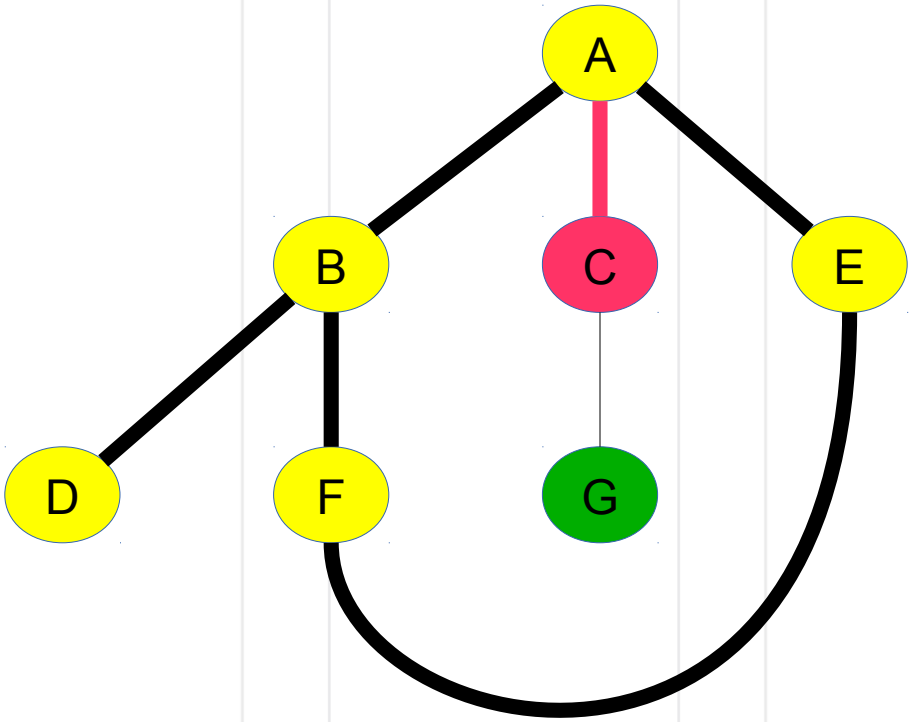
Sügavuti otsing



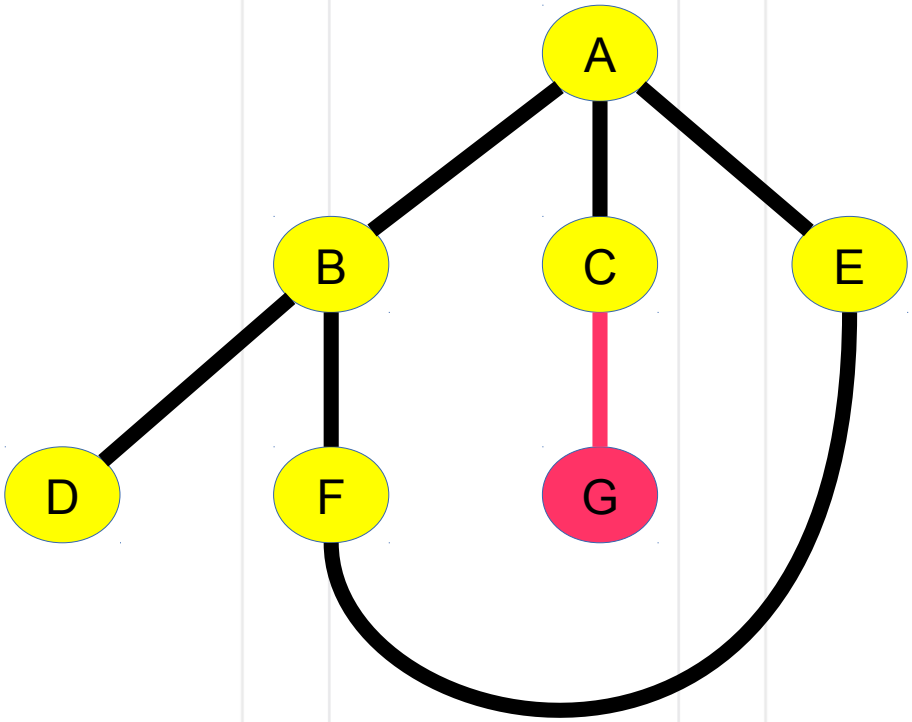
Sügavuti otsing



Sügavuti otsing



Sügavuti otsing



Ülesanne

Ülesanne on nähtaval

https://ained.ttu.ee
https://courses.cs.ttu.ee/pages/ITI0140