Grafové algoritmy 2

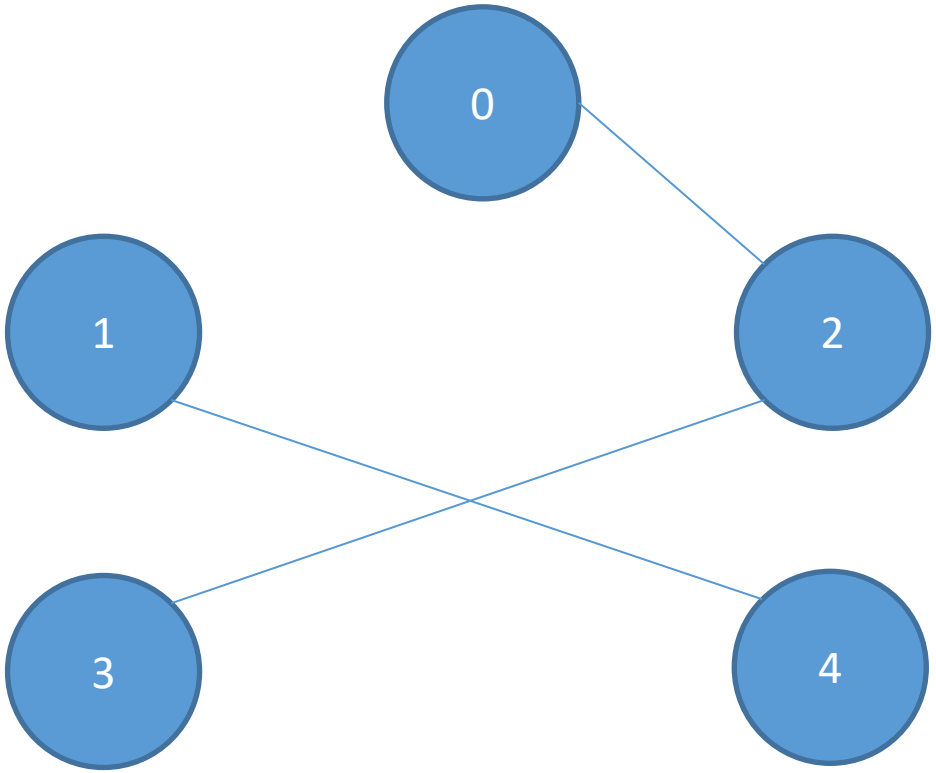
Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZADS - Základní algoritmy a datové struktury

Průchod do hloubky rekurzivně

Graf



Funkce dfs_all

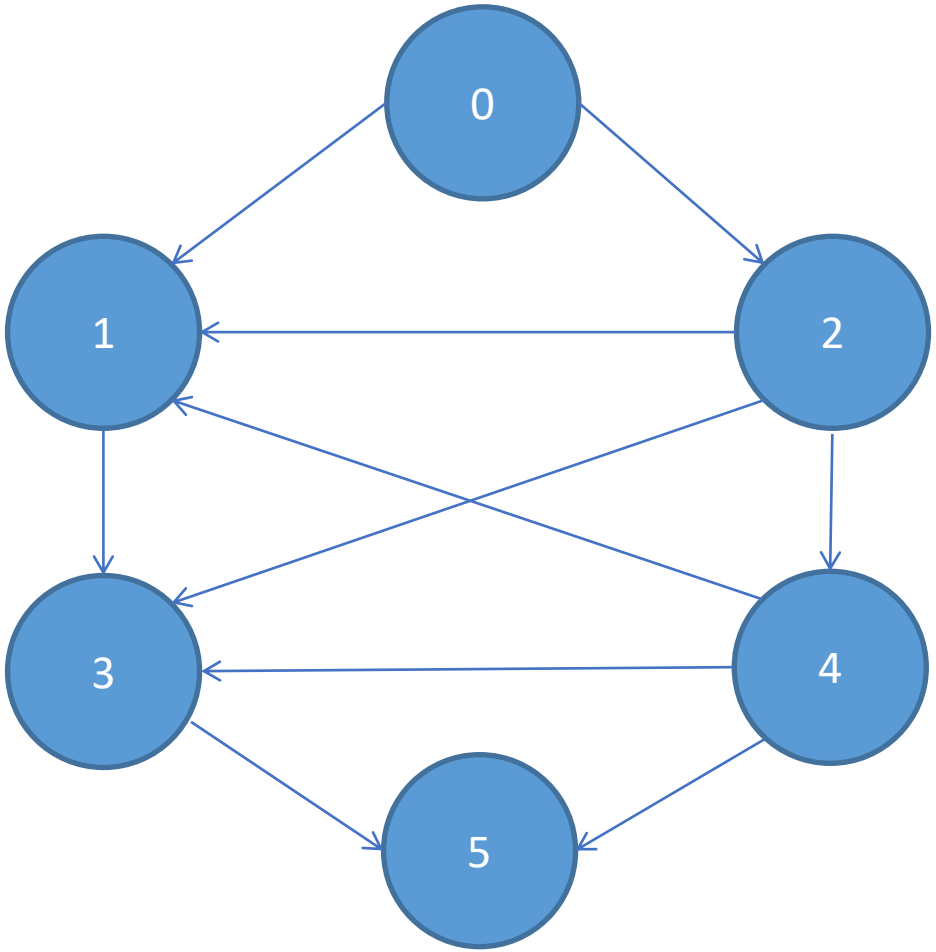
```
def dfs_all(G):
    result=[{"X":False,"D":0,"F":0,"P":None} for i in G["V"]]
    for u in G["V"]:
        if result[u]["X"] == False:
            dfs_visit(G,u,result)
    return result
```

Funkce dfs_visit

```
def dfs_visit(G, u,result):
    global time
    time=time + 1
    result[u]["D"]=time
    result[u]["X"]=True
    for v in G["adj"][u]:
        if result[v]["X"] == False:
            result[v]["P"]=u
            dfs_visit(G,v,result)
    time=time + 1
    result[u]["F"]=time
```

Topologické uspořádání

Graf



Funkce topol

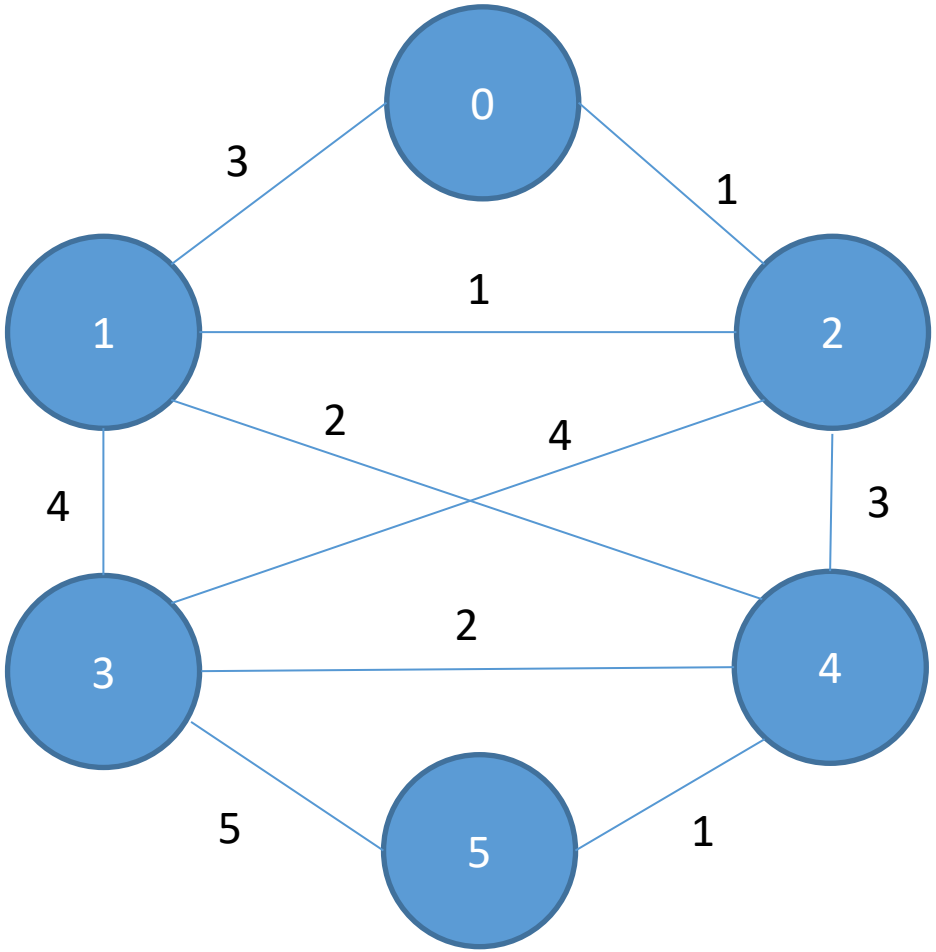
```
def topol(G):
    result=[{"X":False,"D":0,"F":0,"P":None} for i in G["V"]]
    top=[]
    for u in G["V"]:
        if result[u]["X"] == False:
            dfs_visit(G,u,result,top)
    return top
```

Upravené dfs_visit

```
def dfs_visit(G, u,result,t):
    global time
    time=time + 1
    result[u]["D"]=time
    result[u]["X"]=True
    for v in G["adj"][u]:
        if result[v]["X"] == False:
            result[v]["P"]=u
            dfs_visit(G,v,result,t)
    time=time + 1
    result[u]["F"]=time
    t.insert(0,u)
```

Dijkstrův algoritmus

Graf



Reprezentace grafu

```
V=[i for i in range(6)]

adj=[[{"delta":3,"vertex":1},{"delta":1,"vertex":2}],
[{"delta":3,"vertex":0},{"delta":1,"vertex":2},{"delta":4,"vertex":3},{"delta":2,
"vertex":4}],
[{"delta":1,"vertex":0},{"delta":1,"vertex":1},{"delta":3,"vertex":4}],
[{"delta":4,"vertex":1},{"delta":4,"vertex":4},{"delta":5,"vertex":5}],
[{"delta":2,"vertex":1},{"delta":3,"vertex":2},{"delta":4,"vertex":4},{"delta":1,
"vertex":5}],
[{"delta":5,"vertex":3},{"delta":1,"vertex":4}]]

G={"V":V,"adj":adj}
```

Výpočet omegy

```
def compute_omega(G):
    omega=1
    for u  in range(len(G["V"])):
        for node in G["adj"][u]:
            omega=omega + node["delta"]
    return omega
```

Dijksrt

```
def dijkstra(G,s):
    omega=compute_omega(G)
    nodes=[{"key":omega,"data":i,"color":"white","parent":None} for i in
range(len(G["V"]))]
    nodes[s]["key"]=0
    nodes[s]["color"]="gray"
    Q=make_priority_queue()
    insert(Q, nodes[s])
    while not empty(Q):
        m=extract_min(Q)["data"]
        for v in G["adj"][m]:
            id=v["vertex"]
            if nodes[id]["color"] != "black":
```

Dijksrt

```
                x=nodes[m]["key"] + v["delta"]
                if x < nodes[id]["key"]:
                    nodes[id]["parent"]=m
                    if nodes[id]["color"] == "gray":
                        decrease_key(Q, nodes[id], x)
                    else:
                        nodes[id]["key"]=x
                        nodes[id]["color"]="gray"
                        insert(Q, nodes[id])
    nodes[m]["color"]="black"
return nodes
```

Úkol

1. Reprezentujte tuto mapu pomocí grafu.
2. Pro vybrané město vytiskněte nejkratší vzdálenost do ostatních měst (použijte Dijksrtův algoritmus).

- Příklad:
  - Pro Ostravu program vypíše

Od města Ostrava jsou ostatní města vzdálená:

Praha 336 km.

Liberec 340 km.

Plzeň 456 km.

České Budějovice 413 km.

Brno 179 km.

Olomouc 84 km.