10. domácí úkol | Vilém Zouhar

1

1.1 Popis

Stačí z každého vrcholu spustit DFS a dívat se, zdali náhodou nemáme součet K, např. kdykoliv se vnořujeme dál. Pokud ano, tak odpovíme obsahem zásobníku. Při zanořování tedy přičteme cenu cesty z akumulátoru, při vynořování naopak odečteme zpět.

1.2 Korektnost

Pro každý vrchol dfs projde všechny cesty, které tímto vrcholem začínají.

1.3 Pseudokód

```
def DFS(root):
        ...
        while s.not_empty():
                u, acc = s.pop()
                if acc == K:
                        output(s)
                forearch v in s.neighbours:
                        s.push(v, acc+d(u,v))
        ...
foreach v in G:
        DFS(v)
```

1.4 Složitost

$n \cdot DFS \Rightarrow O(n^2)$ časově a $O(n)$ paměťově. Děláme jen konstantně mnoho operací navíc a v každý moment běží právě jednou DFS. Jelikož může existovat klidně řádově n^2 cest, které bychom museli vypsat, tak je časová složitost optimální.

2

2.1 Popis

Intervaly si zesortíme podle jejich začátku a pak je postupně projdeme. Budeme si pamatovat kolik jich je otevřených a vždy, když se budeme zanořovat, tak přičteme počet otevřených. Musíme si ale taky pamatovat, kdy uzavíráme interval. Proto třízení budou dvě.

2.2 Korektnost

Z matematického hlediska opravdu můžeme postupně přičítat počet aktuálně otevřených a dostaneme tak celkový počet dvojic.

2.3 Pseudokód

```
beg = sorted(intervals, => beg)
end = sorted(intervals, => end) # queues
open = 0
total = 0
while end.not_empty():
        if beg.top() < end.top():
                beg.pop()
                total += ++open
        else:
                a = end.top()
                open--
output(total)
```

2.4 Složitost

Dvojité sortění trvá $O(n \cdot \log(n))$, paměťově nepotřebujeme nic navíc, tedy $O(n)$.