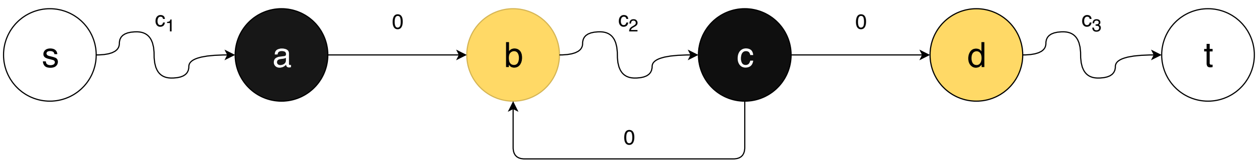
Jméno: PETR VALENTA **UČO:** 487561

0007 *list* 1 *učo* 487561 *body*

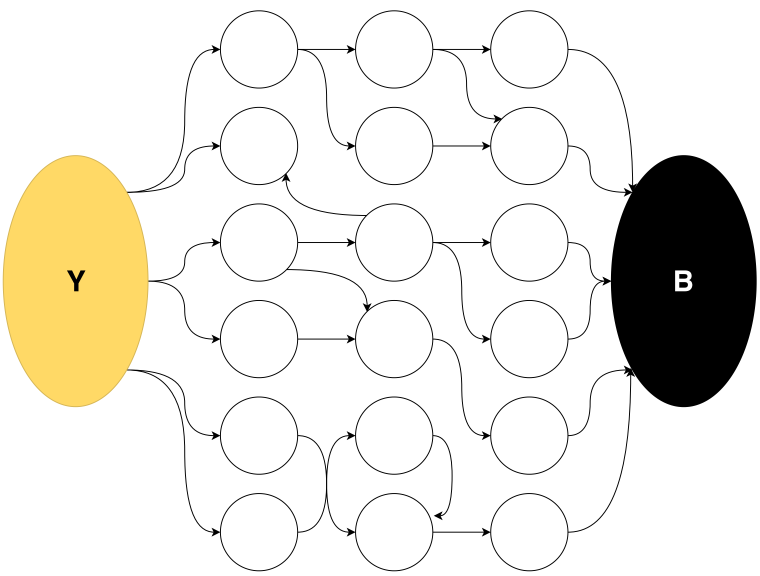
Oblast strojově snímaných informací. Své učo a číslo listu vyplňte zleva dle vzoru číslic. Jinak do této oblasti nezasahujte. 0123456789

Skákací graf je orientovaný graf, ve kterém má každý vrchol přiřazenou barvu (černou, bílou anebo žlutou) a každá hrana má přiřazenou cenu (nezáporné celé číslo). *Skákací cesta* ve skákacím grafu je posloupnost vrcholů $v_1, \ldots, v_k$ taková, že pro každé $1 \leq i < k$ buď (v_i, v_{i+1}) je hranou grafu, anebo vrchol v_i má černou barvu a vrchol v_{i+1} má žlutou barvu. Jinými slovy, skákací cesta sleduje hrany grafu anebo může přeskočit z černého do žlutého vrcholu i když mezi nimi není hrana. *Cena* skákací cesty je rovna součtu cen hran cesty (skoky jsou „zadarmo“).

Vstupem *problému Skákací cesty* je orientovaný graf $G = (V, E)$ takový, že z každého vrcholu vychází nejvýše 2019 hran. Dále je vstupem cenová funkce $c\colon E \to \mathbb{N}_0$, funkce $b\colon V \to \{\text{žlutá, bílá, černá}\}$, a dva vrcholy $s, t \in V$. Navrhněte algoritmus, který najde nejkratší skákací cestu z s do t obsahující právě tři skoky. Požadujeme, aby časová složitost algoritmu byla ve třídě $\mathcal{O}(|V| \log |V|)$.



Obrázek 1: Skákacia cesta



Obrázek 2: Upravený graf pomocou kontrakcií uzlov jednej farby

Jméno: PETR VALENTA **UČO:** 487561

0007 *list* 2 *učo* 487561 *body*

Oblast strojově snímaných informací. Své učo a číslo listu vyplňte zleva dle vzoru číslic. Jinak do této oblasti nezasahujte. 0123456789

Algoritmus

```
function Find-Path(G, s, t)
    G′ ← Optimize-Graph(G)
    if b(s) = black then
        P_{s,a} ← s
    else
        P_{s,a} ← Djikstra-To-Colour(G’,s,black)
    if b(s) = yellow then
        P_{d,t} ← t
    else
        P_{d,t} ← Djikstra-To-Colour-Backwards(G’,t,yellow)
    G″ ← Contract-Colour([yellow, black])
    // find min-cost path from contracted yellow node to contracted black
       node
    contractedYellowBlackPath ← Djikstra(start = yellow, end = black)
    // path between b and c
    P_{b,c} ← Extract-Path(contractedYellowBlackPath, G’)
    P ← (P_{s,a}, (a, b), P_{b,c}, (c, b), P_{b,c}, (c, d), P_{d,t})
    return P

function Djikstra-To-Colour(G, start, colour)
    // find shortest path tree from start using Fibonnaci heaps
    // save min-cost path to nodes of defined colour while constructing tree
    // return node with shortest path and it’s cost
    return (node, cost)

function Djikstra-To-Colour-Backwards(G, start, colour)
    // same as Djikstra-To-Colour but going in opposite direction to edges
    return (node, cost)

function Contract-Colour(G, [colours])
    // contract all nodes of same colour into one node
    return G′

function Extract-Path(P, G)
    P′ ← edge(u, v) ∈ G such that u ∈ Yand(u, v) = edge(Y, v)
    P′ ← P′ + P ∖ (Y, v)
    P′ ← P′ + edge(u, v) ∈ G such that v ∈ Band(u, v) = edge(u, B)
    P′ ← P′ + P ∖ (u, B)
    return P′
```

```
function Optimize-Graph(G)
    G' ← G
    foreach edge (u, v) ∈ G do
        if b(u) = yellow and b(v) = yellow then
            G' ∖ (u, v)
        if b(u) = black and b(v) = black then
            G' ∖ (u, v)
        if b(u) = black and b(v) = white then
            G' ∖ (u, v)
    return G'
```

Časová složitost algoritmu

Djikstrov algortmus pre hľadanie najlepšej cesty má pri použii Fibonacciho haldy časovú zložitosť $\mathcal{O}(|V| \times log(|V|))$. Funkcia Optimize-Graph odstráni hrany, ktoré nepomáhajú k riešeniu v čase $\mathcal{O}(|E|)$. Z vlastností vstupného problmému vieme, že z každého uzlu vedie maximálne 2019 hrán. Preto môžme počet hrán vyjadriť pomocou počtu uzlov ako $\mathcal{O}(2019 \times |V|)$. Následne spustíme dvakrát Djisktrov algoritmus nad upraveným grafom v čase $\mathcal{O}(|V| \times log(|V|))$. Táto fáza má časovú zložitosť $\mathcal{O}(2019 \times |V| + 2 \times (|V| \times log(|V|)))$.

Pre nájdenie najlepšej cesty zo žltého uzlu do čierneho kontraktujeme všetky žlté uzly do jedneho super žlteho uzlu a rovnako aj všetky čierne do super čierneho uzlu. Na to je potrebné prejsť všetky uzly v grafe čo n=am dáva časovú zložitosť $\mathcal{O}(|V|)$. Nad grafom s kontraktovanými uzlami sputíme Djikstrov algoritmus. Celkovo má táto fáza časovú zložitosť $\mathcal{O}(|V| + (|V| \times log(|V|)))$.

Obe fázy dohromady nám davajú výslednú časovú zložitosť $\mathcal{O}(2019 \times |V| + 2 \times (|V| \times log(|V|)) + (|V| + |V| \times log(|V|)))$. Čo môžme zjednodušiť na $\mathcal{O}(2 \times (|V| + (|V| \times log(|V|))))$ ďalej $\mathcal{O}(|V| \times (1 + log(|V|)))$ až $\mathcal{O}(|V| \times log(|V|))$

Korektnost algoritmu

Najlepšia skákacia cesta je zobrazená na obrázku č. 1. Označíme ju SC s uzlami s, a, b, c, d, t kde $s, a, b, c, d, t \in V$. Požiadavka na tri skoky znamená, že musíme navštíviť aspoň jeden čierny uzol. Preto nájdeme najbližší, rozumej aj najlacnejší čierny uzol od začiatočného uzlu s, označíme ho a. Pre skok je popri čiernom uzle potrebný aj žltý uzol, na ktorý sa z čierneho skáče. Nájdeme preto najbližší žltý uzol od cieľového uzlu t a označíme ho d. Po prvom koku na žltý uzol musíme nájsť jeho najbližší čierny uzol aby sme mohli vykonať nasledujúci druhý skok. To isté potom platí aj pre tretí skok.

Najmenšia skákacia cesta musí obsahovať najmenšiu cestu zo žltého do čierneho uzlu, označíme ju $b \rightsquigarrow c$, na ktorú sa napojíme skokom z uzlu a. Keďže ide o najlepšiu žlto-čiernu cestu, zopakuje sa v cykle skokom z uzlu c na uzol b. Posledným tretím skokom potom bude skok z uzlu c na uzol d.

Najkratšia skákacia cesta SC je potom postupnosť vrcholov: $s \rightsquigarrow a, a \to b, b \rightsquigarrow c, c \to b, b \rightsquigarrow c, c \to d, d \rightsquigarrow t$. Táto ceste je určite najkratšia, pretože cesta $b \rightsquigarrow c$ je najkratšou žlto-čiernou cestou a pridanie akejkoľvek ďalšej hrany cenu cesty len zvýši. Preto pre vykonenie práve troch skokov je najlepšou cestou cyklus $b \rightsquigarrow c, c \to b$, ktorý označíme C. Začatím v uzle s s cieľom v uzle t znamená, že sa musíme dostať do aj z cyklu C. Pridaním najbližšieho (najlacnejšieho) čierneho uzlu a od uzlu s zabezpečíme skok zadarmo do C. Rovnako najbližší žltý uzol d od uzlu t, na ktorý možno skočiť zadarmo z C. Cesta SC preto stojí $c(s \rightsquigarrow a) + c(b \rightsquigarrow c) + c(d \rightsquigarrow t)$, nakoľko všetky skoky majú cenu 0.

O každej z týchto ciest vieme, že je najlepšou možnou s požadovaným začiatkom a cieľom, preto môžme povedať, že nájdená cesta SC je najkratšou cestou z s do t.

Korektnosť optimalizácie hrán

Vynechanť všetky hrany *žtlá* → *žltá* môžme, pretože na tú vzdialenejšiu skočíme z čiernej s cenou $0 \ll$ *žltá* → *žltá*. Hrany *čierna* → *čierna* vynecháme tiež, pretože z čierneho uzlu skáčeme rovno na žltú rovnako s cenou 0. Pridaním hrany *čierna* → *čierna* by sme zvýšili cenu cesty SC. Rovnaký dôvod má aj odstránenie hrán *čierna* → *biela*. Tieto tvrdenia platia, nakoľko všetky hrany majú nezápornú cenu.