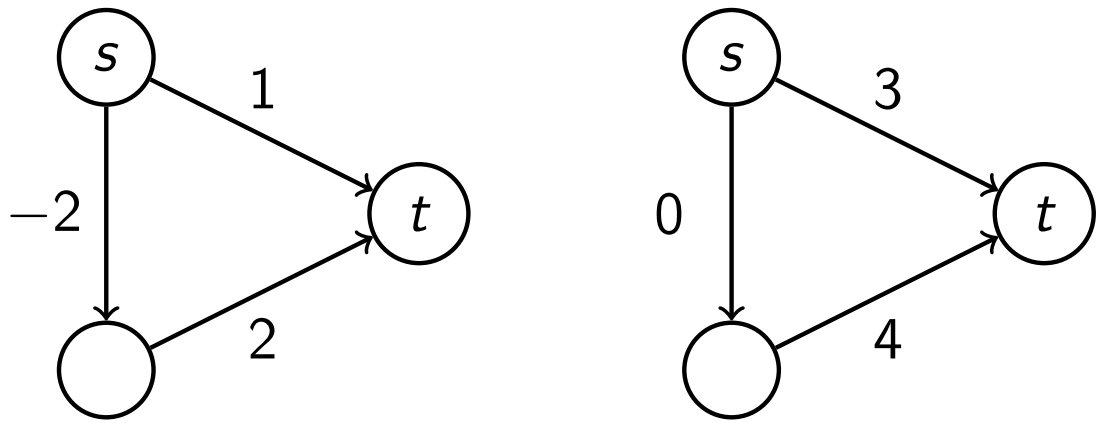
Reiknirit Johnsons (1977)

Bergur Snorrason

30. mars 2023

- Við höfum séð hvernig það má finna fjarlægðir milli alla hnúta í neti í $\mathcal{O}(V^3)$ tíma með reikniriti Floyds og Warshalls.
- Það má þá bæta þetta lítilega fyrir net sem hafa ekki marga leggi.
- Ef netið okkar er með jákvæða leggi getum við nota Dijkstra úr hverjum hnút og fengið tímaflækjuna $\mathcal{O}(V(E+V)\log E)$.
- En hvað gerum við ef við erum með neikvæða leggi?

- Hugmyndin er að breyta vigtinni á leggjum þannig að vigtirnar verði jákvæðar, en án þess að breyta hvaða vegir eru stystir.
- Eðlilegt er að spyrja sig hvort það nægi ekki að bæta nógu stórri tölu við alla leggina þannig að allir leggir sé jákvæðir.
- Þetta virkar ekki, því þetta lengir vegi með fleiri leggi meira en vegi með fáa leggi.



- Reiknirit Johnsons breytir vigtinni á leggjunum þannig að stystu vegirnir breytast ekki en allir leggirnir verða jákvæðir.

- Látum $G = (E, V, w)$ vera vegið stefnt net og $f : V \to \mathbb{Z}$.
- Við getum þá búið til nýtt net $G' = (E, V, w')$ þannig að $w'(e) = w(e) + f(u) - f(v)$.
- Ef $v_1, v_2, \ldots, v_k$ er vegur þá er

$$\begin{aligned}\sum_{j=1}^{k-1} w'(v_j, v_{j+1}) &= \sum_{j=1}^{k-1} w(v_j, v_{j+1}) + f(v_j) - f(v_{j+1}) \\ &= \sum_{j=1}^{k-1} w(v_j, v_{j+1}) + f(v_1) - f(v_k)\end{aligned}$$

- Svo stysti vegurinn milli hnúta u og v í G er líka stysti vegurinn milli u og v í G'.
- Spurningin er þá: Er hægt að velja f þannig að $w' \geq 0$?

- Gerum ráð fyrir að það sé engin neikvæð rás í G.
- Við getum þá notað reinkirit Bellmans og Fords til að finna stysta veginn sem endar í u, fyrir öll u.
- Táknum lengdina á þessum vegi með $h(u)$.
- Tökum nú eftir að ef e er leggur úr u í v þá er $w(e) + h(u) \geq h(v)$.
- Svo ef f, á glærunni á undan, er látið vera h þá er G' ekki með neinn neikvæðann legg.
- Við getum því notað reiknirit Dijkstras fyrir hvern hnút til að finna stystu fjarlægðina milli u og v í G'.
- Við höfum svo að ef fjarlægðin milli u og v í G' er μ, þá er fjarlægðin milli u og v í G gefin með $\mu - h(u) + h(v)$.

- Ef það er neikvæð rás í netinu fáum við að $h(u) = -\infty$ fyrir einhverja hnúta u.
- Við leysum þetta með því finna all hnúta sem eru hluti af einhverri rás, fjarlægja þá úr netinu og reikna svo h fyrir hnútana sem eru ekki hluti af neikvæðri rás.
- Við getum notað þetta net til að finna allar fjarlægðir sem eiga ekki að vera $-\infty$.
- En hvernig finnum við alla hnúta sem eru hluti af neikvæðri rás?

- Rifjum upp að net er stranglega samanhangandi ef fyrir alla hnúta u og v er til vegur frá u til v og vegur frá v til u.
- Ef netið okkar er stranglega samanhangandi og inniheldur neikvæða rás þá eru allir hnútar hluti af neikvæðri rás.
- Svo við getum notað reiknirit Bellmans og Fords til að finna stysta veg sem endar í hverjum hnút, og þá vitum við hvort það sé neikvæð rás í stranga samhengisþættinum.
- Fyrir almennt net eru þó allar rásir alfarið í einum ströngum samhengisþætti.
- Svo okkur nægir að skoða hvern stranga samhengisþátt fyrir sig.

```
31 ll dfs(vvii& g, ll u, ll d, ll *s, ll *a, ll *l)
32 {
33     l[u] = d, s[++s[0]] = u;
34     ll i, x, z = d;
35     for (i = 0; i < g[u].size(); i++)
36     {
37         x = g[u][i].first;
38         if (l[x] == -1) d = dfs(g, x, d + 1, s, a, l);
39         if (a[x] == -1) l[u] = min(l[u], l[x]);
40     }
41     if (l[u] == z) while (a[u] == -1) a[s[s[0]--]] = u;
42     return d;
43 }
44
45 void scc(vvii& g, ll *a)
46 {
47     ll i, n = g.size(), s[n + 1], l[n];
48     for (i = 0, s[0] = 0; i < n; i++) l[i] = a[i] = -1;
49     for (i = 0; i < n; i++) if (l[i] == -1) dfs(g, i, 0, s, a, l);
50 }
```

```
52 void bellman_ford(vvii& g, vi& s, vi& d, vi& a)
53 {
54     ll i, j, k, m = s.size(), x, w, q[3 + 2*m*m], p[3 + 2*m*m];
55     q[0] = q[1] = p[0] = p[1] = 2;
56     for (i = 0; i < m; i++)
57         d[s[i]] = 0, a[s[i]] = 1, q[q[1]++] = s[i], p[p[1]++] = 0;
58     while (q[0] != q[1])
59     {
60         i = q[q[0]++], j = p[p[0]++], a[i] = 0;
61         for (k = 0; k < g[i].size(); k++) if (a[g[i][k].first] != -1)
62         {
63             x = g[i][k].first, w = g[i][k].second;
64             if (d[x] != -INF && d[i] + w < d[x])
65             {
66                 d[x] = j < m ? d[i] + w : -INF;
67                 if (!a[x]) a[x] = 1, q[q[1]++] = x, p[p[1]++] = j + 1;
68             }
69         }
70     }
71 }
```

```
73 vi all_nodes_on_negative_cycle(vvii& g)
74 {
75     ll i, j, n = g.size(), a[n];
76     vi r(n), d(n), v(n);
77     vvi s(n);
78     scc(g, a);
79     for (i = 0; i < n; i++) s[a[i]].push_back(i), v[i] = -1;
80     for (i = 0; i < n; i++)
81     {
82         bellman_ford(g, s[i], d, v);
83         for (j = 0; j < s[i].size(); j++) r[s[i][j]] = (d[s[i][j]] != -INF);
84         for (j = 0; j < s[i].size(); j++) v[s[i][j]] = -1;
85     }
86     return r;
87 }
```

- Búum nú til nýtt net, sem er ekki með neina neikvæða leggi.
- Táknum með $R \subset V$ mengi þeirra hnúta sem eru hluti af einhverri neikvæðri rás.
- Við látum $G' = (\{1, 2\} \times V, E', w')$ þannig að:
  - leggur úr $(1, u)$ í $(1, v)$ er í E' ef (u, v) er í E og $u \notin R$ og $v \notin R$.
  - leggur úr $(1, u)$ í $(2, v)$ er í E' ef (u, v) er í E og $v \in R$.
  - leggur úr $(2, u)$ í $(2, v)$ er í E' ef (u, v) er í E,
  - leggur úr $(1, u)$ í $(1, v)$ hefur vigt $w(u, v) + h(u) - h(v)$,
  - leggur úr (j, u) í $(2, v)$ hefur vigt 0, fyrir bæði $j = 1$ og $j = 2$.
- Ein leið til að ímynda sér netið G' er að við séum með tvö eintök af netinu G, annað ofan á hinu.
- Við byrjum í neðra netinu og alltaf þegar við lendum á hnút í R (hnút sem er í neikvæðri rás) þá förum við upp í efra netið.
- Fjarlægðin úr u til v í G er þá $-\infty$ ef það er vegur úr $(1, u)$ í $(2, v)$ í G', og $\mu - h(u) + h(v)$ annars, þar sem μ er fjarlægðin milli $(1, u)$ og $(1, v)$ í G'.

```
89  vvi johnson(vvii &g)
90  {
91      ll i, j, n = g.size(), x, w;
92      vi s, b = all_nodes_on_negative_cycle(g), c(n, -1), h(n, -INF);
93      for (i = 0; i < n; i++) if (b[i]) s.push_back(i);
94      bellman_ford(g, s, h, c);
95      vvi d(n);
96      vvii q(2*n);
97      for (i = 0; i < n; i++) for (j = 0; j < g[i].size(); j++)
98      {
99          x = g[i][j].first, w = g[i][j].second;
100         if (h[x] == -INF) q[i].push_back(ii(x + n, 0));
101         else if (h[i] != -INF) q[i].push_back(ii(x, w + h[i] - h[x]));
102         q[i + n].push_back(ii(x + n, 0));
103     }
104     for (i = 0; i < n; i++) d[i] = dijkstra(q, i);
105     for (i = 0; i < n; i++) for (j = 0; j < n; j++)
106         if (d[i][j] != INF) d[i][j] = d[i][j] - h[i] + h[j];
107     for (i = 0; i < n; i++) for (j = 0; j < n; j++)
108         if (d[i][j + n] != INF) d[i][j] = -INF;
109     for (i = 0; i < n; i++) d[i].resize(n);
110     return d;
111 }
```

- Við framkvæmum reiknirit Bellmans og Fords tvisvar og reiknirit Dijkstras V sinnum.
- Tímaflækja reiknirit Bellmans og Fords er tímaflækuna $\mathcal{O}(E \cdot V)$.
- Tímaflækja reiknirit Dijkstras er tímaflækuna $\mathcal{O}((E + V) \log E)$.
- Svo tímaflækjan er $\mathcal{O}(E \cdot V + V \cdot (E + V) \log E)$.
- Þetta bætir bara reiknirit Floyds og Warshalls ef við erum með fáa leggi í netinu.

Reiknirit	Útskýring	Skilyrði	Tímaflækja	Tímaflækja á þétt net
BFS	Einn í alla	Óvegið	$\mathcal{O}(E+V)$	$\mathcal{O}(V^2)$
Dijkstra	Einn í alla	Jákvæðir leggir	$\mathcal{O}((E+V)\log E)$	$\mathcal{O}(V^2 \log V)$
Bellman-Ford	Einn í alla		$\mathcal{O}(E \cdot V)$	$\mathcal{O}(V^3)$
Dijkstra	Allir í alla	Jákvæðir leggir	$\mathcal{O}(V(E+V)\log E)$	$\mathcal{O}(V^3 \log V)$
Bellman-Ford	Allir í alla		$\mathcal{O}(V^2 \cdot E)$	$\mathcal{O}(V^4)$
Floyd-Warshall	Allir í alla		$\mathcal{O}(V^3)$	$\mathcal{O}(V^3)$
Johnson	Allir í alla		$\mathcal{O}(E \cdot V + V(E+V)\log E)$	$\mathcal{O}(V^3 \log V)$