Reiknirit Tarjans

Bergur Snorrason

March 4, 2024

- Skilgreinum vensl $\sim$ á milli hnúta í óstefndu neti með því að $u \sim v$ ef og aðeins ef til er vegur milli u og v.
- Auðvelt er að sýna að þetta eru jafngildisvensl (sjálfhverf, samhverf og gegnvirk).
- Við megum því skilgreina *samhengisþátt* í netinu sem jafngildisflokka þessara vensla.
- Samhengisþáttur í neti er því óstækkanlegt mengi þannig að komast má hverjum hnút í menginu til hvers annars með vegi.
- Segja má að net sé samanhangandi þá og því aðeins að það innihaldi einn samhengisþátt.

- Við getum beitt dýptarleit eða breiddarleit til að finna alla hnúta sem eru í sama samhengisþætti og tiltekinn hnútur.
- Ef við framkvæmum dýptarleit sem byrjar í einhverjum hnút x mun hún heimsækja alla hnúta í sama samhengisþætti og x.
- Við þurfum að passa að leita ekki í tilteknum samhengisþætti oftar en einu sinni.

```
vi v;
void dfs(vvi& g, int x, int c)
{
    int i;
    v[x] = c;
    for (i = 0; i < g[x].size(); i++) if (v[g[x][i]] == -1)
        dfs(g, g[x][i], c);
}

int main()
{
    int i, j, n, m, c = 0, x , y;
    cin >> n >> m;
    vvi g(n);
    for (i = 0; i < m; i++)
    {
        cin >> x >> y;
        x--, y--;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    v = vi(n, -1);
    for (i = 0; i < n; i++) if (v[i] == -1) dfs(g, i, c++);
    printf("Fjöldi samhengisþátta er %d.\n", c);
    for (i = 0; i < n; i++)
        printf("Hnútur %d er í samhengisþætti %d.\n", i + 1, v[i] + 1);
    return 0;
}
```

- Við framkvæmum sömu vinnu og í dýptarleit, svo tímaflækan er $\mathcal{O}(E + V)$.

- Það er önnur náttúruleg leið til að finna samhengisþætti.
- Við getum notað sammengisleit.
- Þá sameinum við þá hnúta sem eru nágrannar.

```c
int uf_find(int *p, int x)
{
    return p[x] < 0 ? x : (p[x] = uf_find(p, p[x]));
}

void uf_join(int *p, int x, int y)
{
    int rx = uf_find(p, x), ry = uf_find(p, y);
    if (rx == ry) return;
    if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
    else p[rx] += p[ry], p[ry] = rx;
}

void uf_init(int *p, int n)
{
    for (int i = 0; i < n; i++) p[i] = -1;
}

int main()
{
    int i, j, n, m, c = 0, x , y;
    scanf("%d%d", &n, &m);
    int p[n], v[n];
    uf_init(p, n);
    for (i = 0; i < m; i++)
    {
        scanf("%d%d", &x, &y);
        x--, y--;
        uf_join(p, x, y);
    }
    for (i = 0; i < n; i++) v[i] = 0;
    for (i = 0; i < n; i++) v[uf_find(p, i)] = 1;

    printf("Fjöldi samhengisþátta er %d.\n", c);
    for (i = 0; i < n; i++)
        printf("Hnútur %d er í samhengisþætti %d.\n", i + 1, uf_find(p, i));
    return 0;
}
```

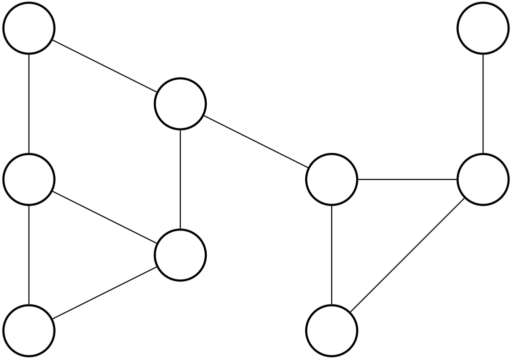
- Ef við tölum um að fjarlægja hnút úr neti þá er átt við að hnúturinn ásamt öllum leggjum til og frá honum eru fjarlægðir.
- Gerum ráð fyrir að við höfum net G og látum G_u tákna netið þar sem hnútur u hefur verið fjarlægður.
- Við segjum að hnútur u sé *liðhnútur* (e. *articulation point*) ef G hefur færri samhengisþætti en G_u.

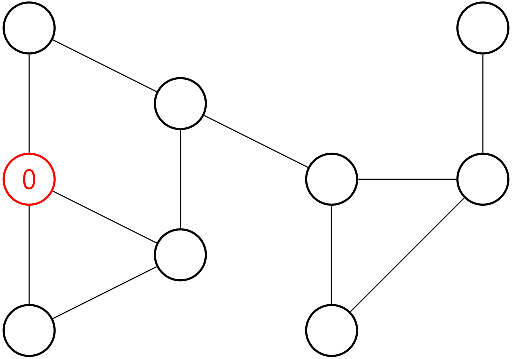
- Til að fjarlægja legg úr neti nægir að fjarlægja legginn.
- Táknum þá netið G án leggsins e með G_e.
- Leggur e eru sagður vera *brú* (e. *bridge*) ef G hefur færri samhengisþætti en G_e.
- Með öðrum orðum er hnútur u liðhnútur (leggur e brú) ef til eru hnútar v_1 og v_2 í sama samhengisþætti þannig að allir vegir frá v_1 til v_2 fari í gegnum hnútinn u (legginn e).

- Ein leið til að finna alla liðhnúta er að telja fyrst samhengisþætti netsins, fjarlægja hnút, telja samhengisþætti og endurtaka fyrir alla hnúta.
- Þar sem við þurfum að finna alla samhengisþætti $2V + 1$ neta er þessi aðferð með tímaflækju $\mathcal{O}(V^2 + VE)$.
- Samskonar aðferð til að finna brýr væri með tímaflækju $\mathcal{O}(E^2 + VE)$.
- Þetta er þó ekki æskilegt, því getum við getum fundið bæði alla liðhnúta og allar brýr með einni dýptarleit.

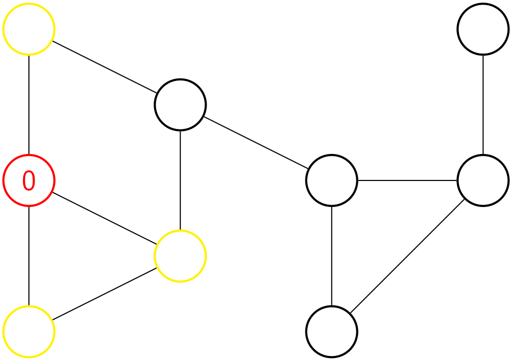
- Gerum ráð fyrir að netið okkar sé samanhangandi.
- Ef svo er ekki getum við beitt þessari aðferð á hvern samhengisþátt.
- Veljum einhvern hnút og framkvæmum dýptarleit frá honum.
- Skilgreinum svo tvær breytur fyrir hvern hnút u út frá þessari dýptarleit, u_{low} og u_{num}.
- Talan u_{num} segir hversu mörg skref í leitin við tókum til að finna hnútinn u.
- Talan u_{low} er minnsta gildið v_{num} þar sem v er hnútur sem við við getum ferðast til án þess að nota leggi sem hafa verið notaðir í leitinni.

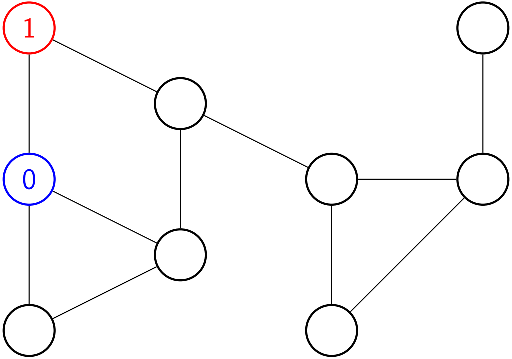
- Gerum ráð fyrir að leitin okkar fari úr hnút u í hnút v með legg e.
- Ef $v_{low} > u_{num}$ þá er e brú.
- Þetta þýðir að eina leiðin frá v til u er í gegnum legginn e.
- Ef $v_{low} \geq u_{num}$ gildir fyrir einhvern nágranna u þá er u liðhnútur.
- Þetta þýðir að eina leiðin frá v í fyrri hnúta leitarinnar er í gegnum hnútinn u.
- Við þurfum þó að afgreiða sérstaklega upphafshnútinn í leitinni.
- Ef upphafshnúturinn þarf að heimsækja fleiri en einn nágranna sinna er hann liðhnútur.

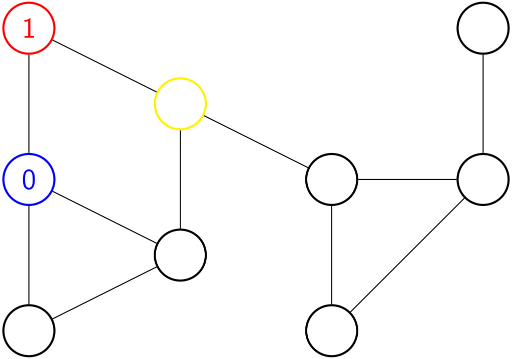


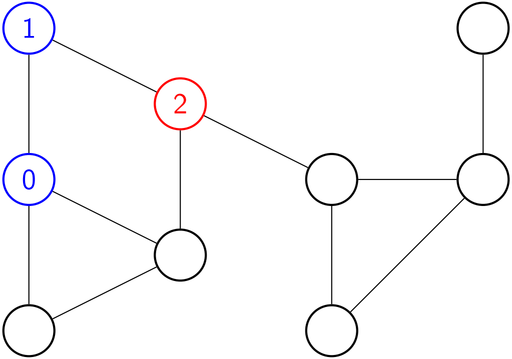


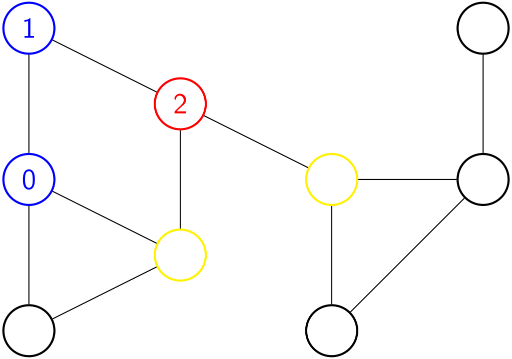
0

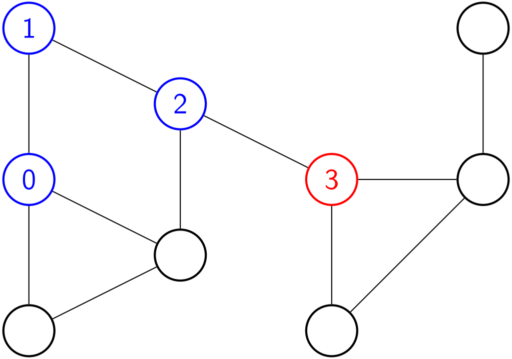

0

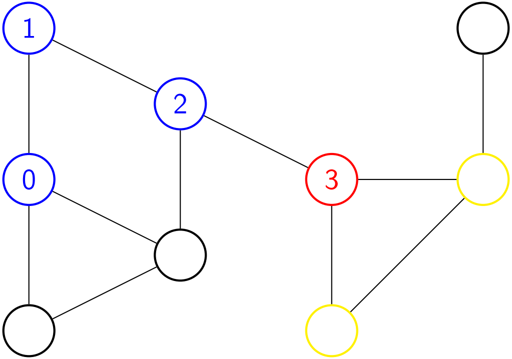

1
0

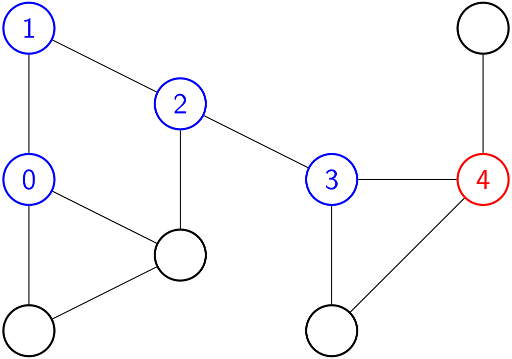

1
0

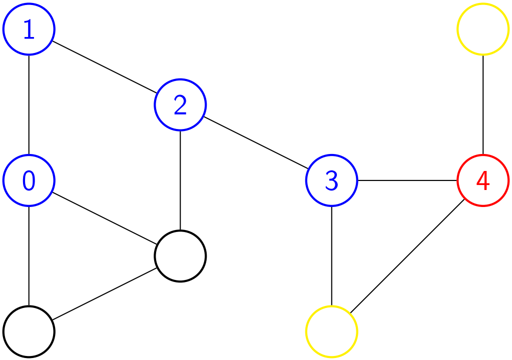

1
2
0

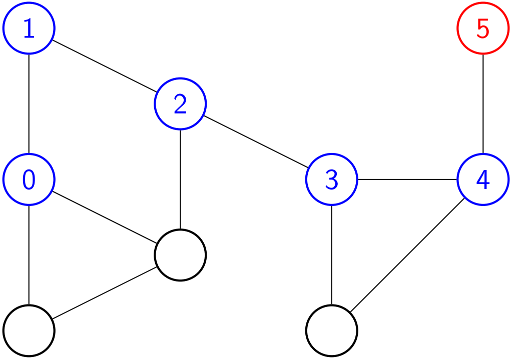

1
2
0

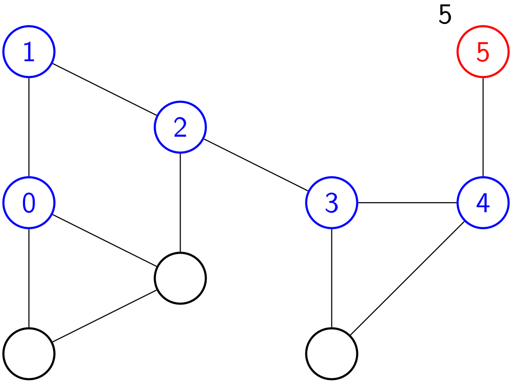

1
2
0
3

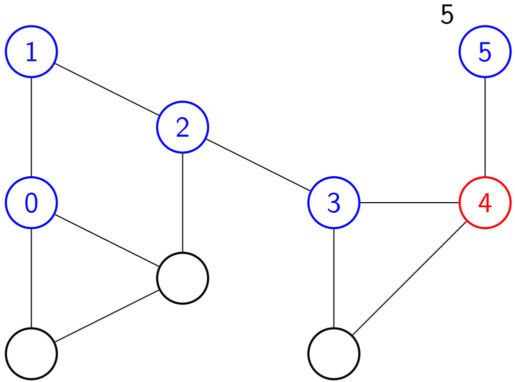

1
2
0
3


1
2
0
3
4

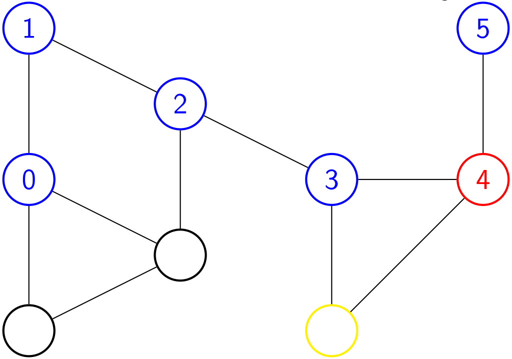

1
2
0
3
4


1
5
2
0
3
4

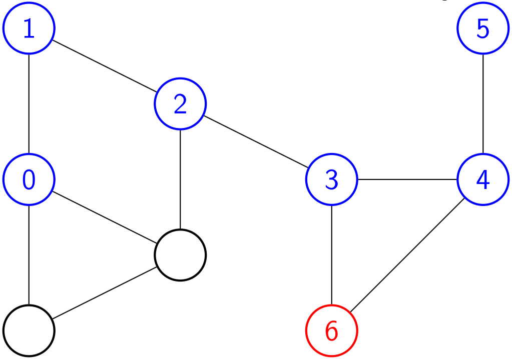

5
1
5
2
0
3
4

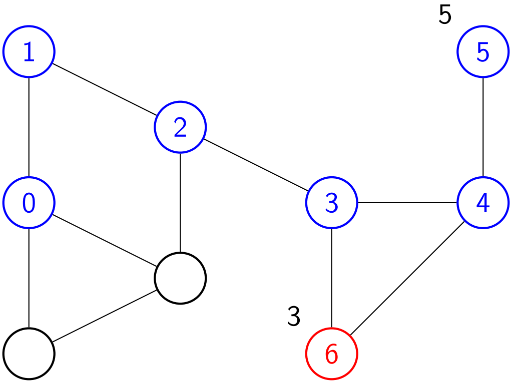

5
1
5
2
0
3
4

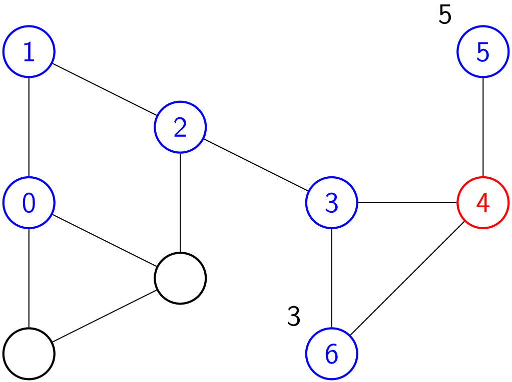
5

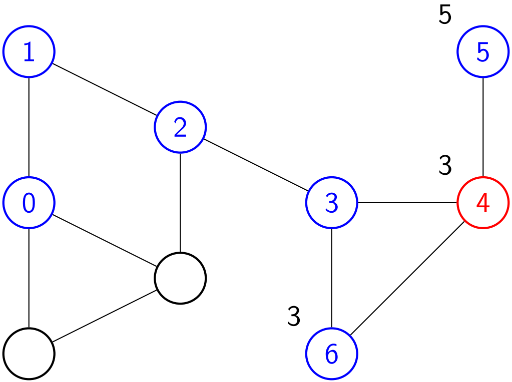


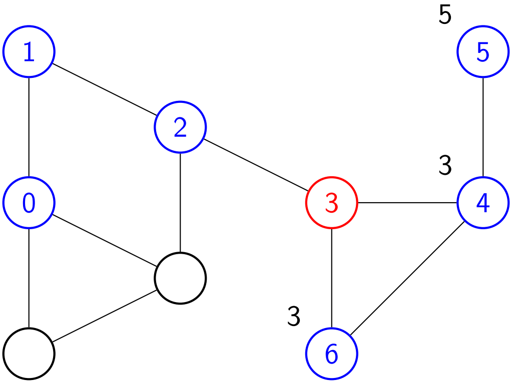
5

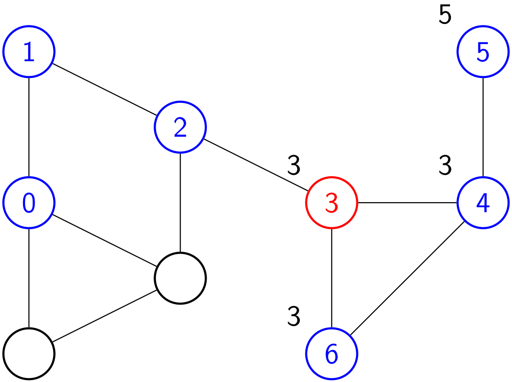


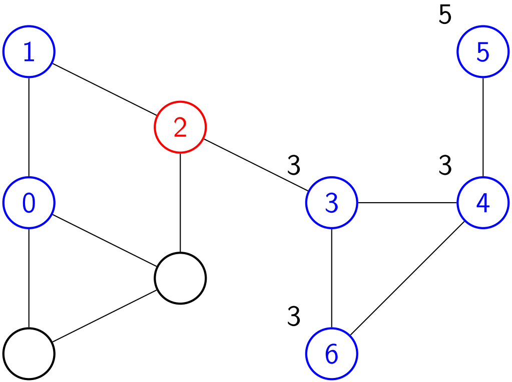

5
1
5
2
0
3
4
3
6

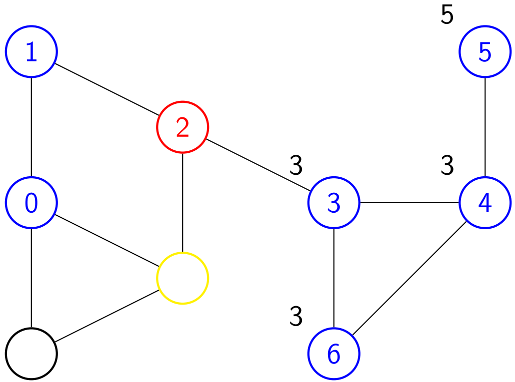

5
1
5
2
0
3
4
3
6

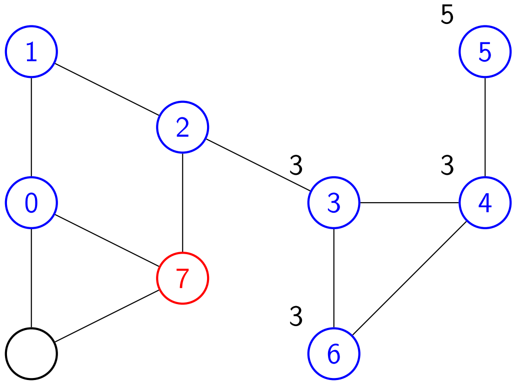

5
1
5
2
3
0
3
4
3
6

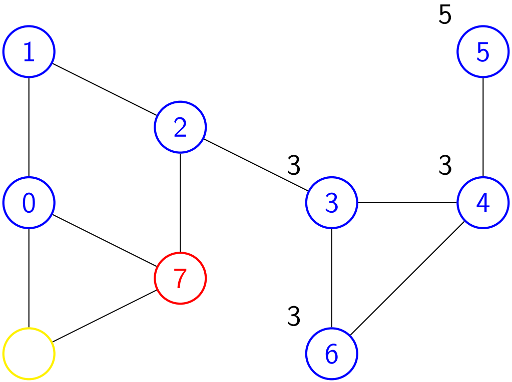

5
1
5
2
3
0
3
4
3
6

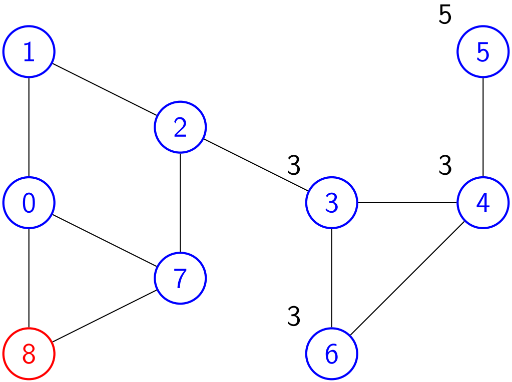

5
1
5
2
3
3
0
3
4
3
6

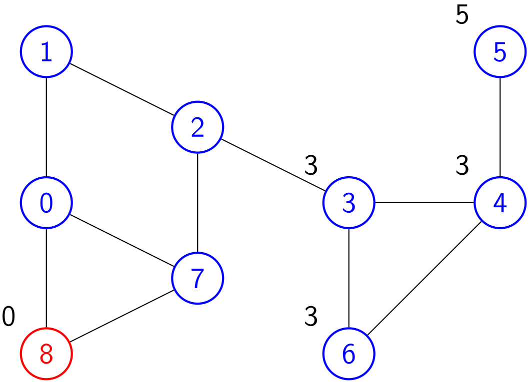

5
1
5
2
3
3
0
3
4
3
6

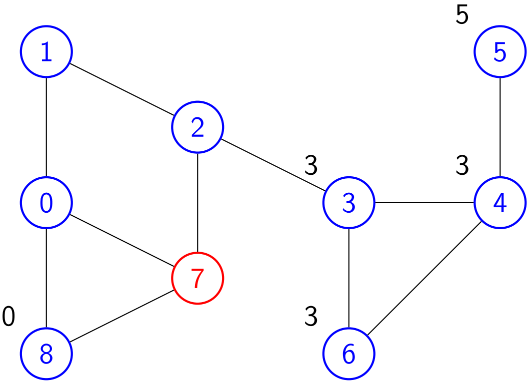

5
1
5
2
3
3
0
3
4
3
6

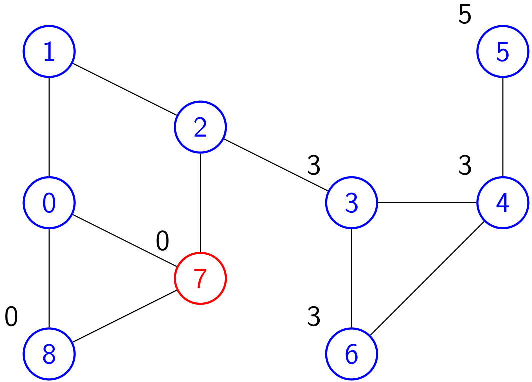

5
1
5
2
3
3
0
3
4
7
3
6

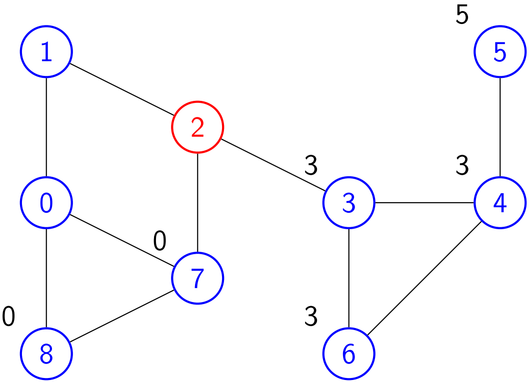

5
1
5
2
3
3
0
3
4
7
3
6

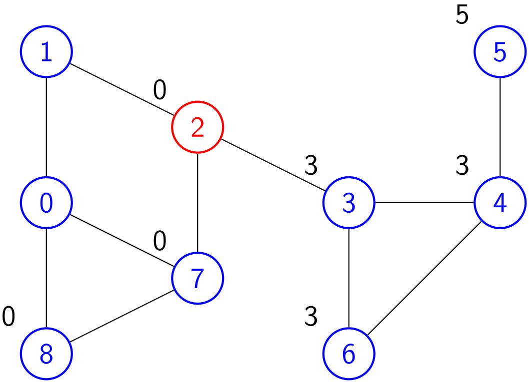

5
1
5
2
3
3
0
3
4
7
3
8
6

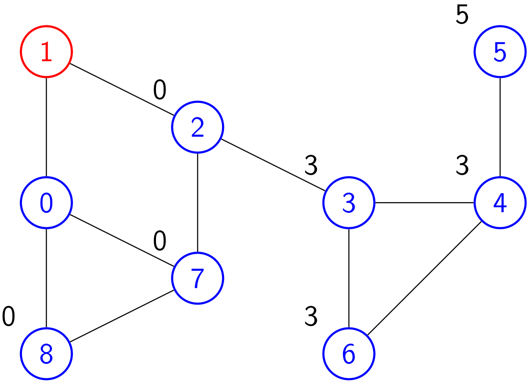

5
1
5
2
3
3
0
3
4
7
0
3
8
6

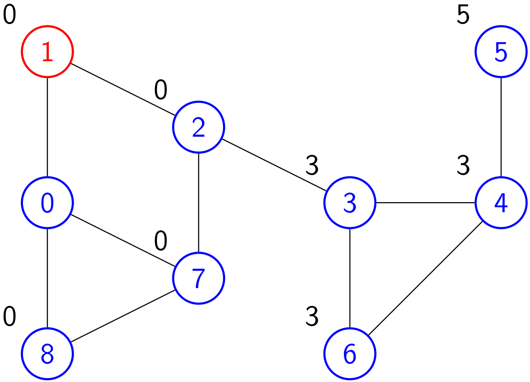

5
1
5
2
3
3
0
3
4
7
0
3
8
6

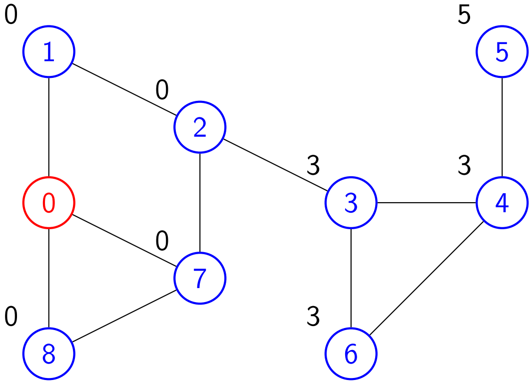

5
1
5
2
3
3
0
3
4
0
7
0
3
8
6

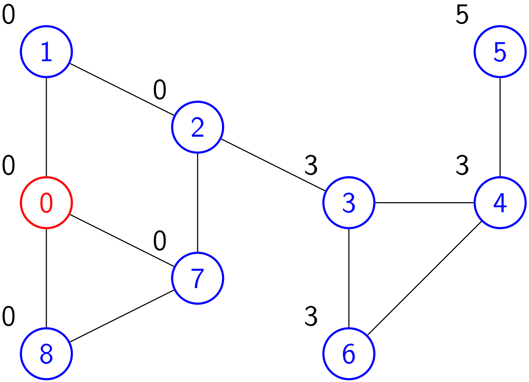

5
1
5
2
3
3
0
3
4
0
7
0
3
8
6

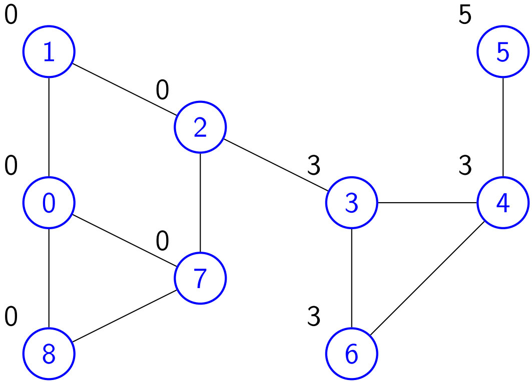

5
1
5
0
2
3
3
0
3
4
0
7
0
3
8
6

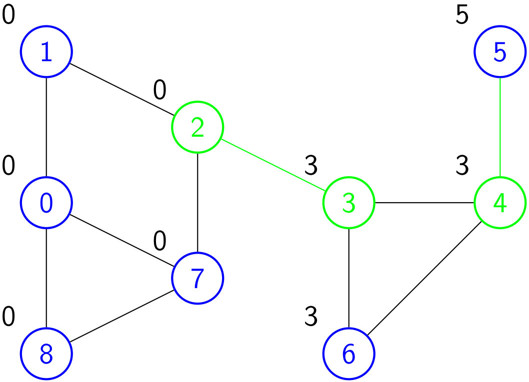

5
5
1
0
2
3
3
0
3
4
0
7
0
3
8
6


0
1
0
2
3
3
0
3
4
5
5
0
7
0
8
3
6


0
1
5
5
0
2
3
3
3
0
4
0
7
0
3
8
6


0
1
5
5
0
2
0
3
3
0
3
4
0
7
0
3
8
6


0
1
0
2
0
0
3
3
5
5
4
0
7
0
8
3
6



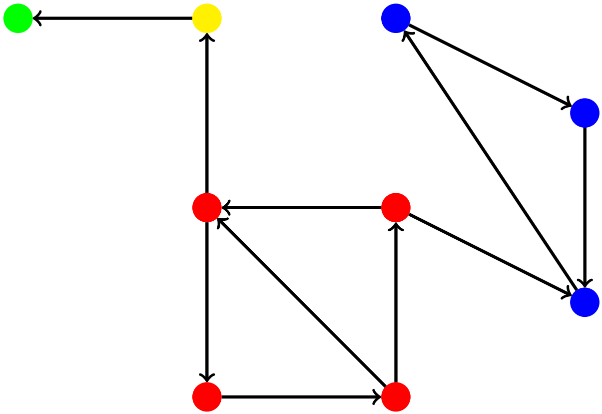
0
1
0
2
0
0
3
0
7
0
8
5
5
3
3
3
4
3
6

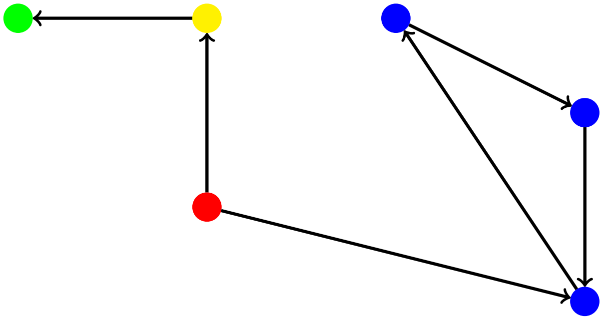
```
int dfs(vvi &g, int u, int p, int d)
{
    int i, x, c = 0, z = d, y; v[u] = d;
    for (i = 0; i < g[u].size(); i++) if (g[u][i] != p)
    {
        x = g[u][i];
        if (v[x] == -1)
        {
            y = dfs(g, x, u, d + 1), c++;
            z = min(z, y);
            if (y > v[u]) bri.push_back(ii(u, x));
            if (p != -1 && y >= v[u]) a[u] = 1;
        }
        else z = min(z, v[x]);
    }
    if (p == -1 && c > 1) a[u] = 1;
    return z;
}

void cpb(vvi &g)
{
    cp.clear(), bri.clear();
    for (int i = 0; i < g.size(); i++) v[i] = -1, a[i] = 0;
    for (int i = 0; i < g.size(); i++) if (v[i] == -1) dfs(g, i, -1, 0);
    for (int i = 0; i < g.size(); i++) if (a[i]) cp.push_back(i);
}
```

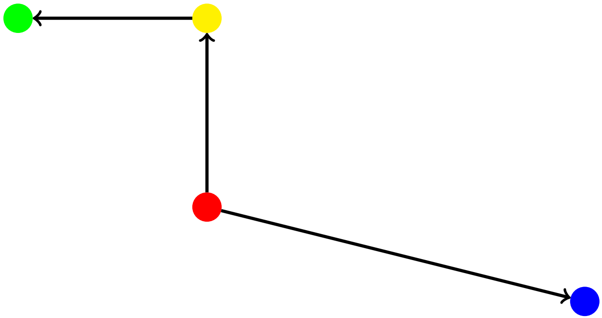
- Tímaflækjan er $\mathcal{O}(E + V)$ því það er tímaflækja dýptarleitar.

- Gerum ráð fyrir að við séum með stefnt net.
- Þá eru venslin sem við skilgreindum áðan ekki lengur jafngildisvensl því þau eru ekki samhverf.
- Við getum þó gert þau samhverf með því að krefjast að það sé til vegur í báðar áttir.
- Með öðrum orðum er $x \sim y$ ef og aðeins ef til er vegur frá u til v og vegur frá v til u.
- Jafngildisflokkar þessara vensla eru kallaðir *strangir samhengisþættir* (e. *strong connected components*).
- Ég mun þó leyfa mér að kalla þetta *samhengisþætti* þegar ljóst er að við séum að ræða um stenft net.

- Takið eftir að ef netið inniheldur rás þá eru allir hnútar í rásinni í sama samhengisþætti.
- Einnig gildir að ef netið inniheldur enga rás er hver hnútur sinn eigin samhengisþáttur.
- Slík net kallast *stefnd órásuð net* (e. *directed acycle graphs (DAG)*).
- Þau hafa ýmsa þæginlega eiginleik, til dæmis má beyta kvikri bestun á þau.
- Við getum breytt stefndu neti í órásað stefnt net með því að deila út jafngildisvenslunum.
- Nánar, þá lítum við svo á að hnútar í sama samhengisþætti séu í raun sami hnúturinn og verður leggur milli samhengisþátta ef vegur liggur milli einhverja hnúta í samhengisþáttunum sem fer ekki í annan samhengisþátt.
- Við köllum þetta net *herpingu* (e. *contraction*) upprunalega netsins.

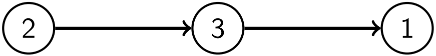






- Til að finna herpinguna þurfum við fyrst að finna samhengisþættina.
- Við getum breytt lítilega forritinu sem við vorum með áðan til að finna samhengisþætti stefnds nets.
- Við getum skoðað hvort $u_{low} = u_{num}$ á leiðinni upp úr endurkvæmninni.
- Ef svo er þá er u fyrsti hnúturinn sem við sáum í samhengisþættinum sem u tilheyrir.
- Við geymum því hnútana sem við heimsækjum á hlaða.
- Þegar við finnum umrætt u (á leiðinni upp úr endurkvæmninni) tínum við af hlaðanum þangað til við sjáum u og setjum alla þá hnúta saman í samhengisþátt.

- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.



num

low

- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.

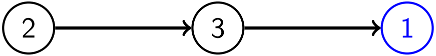
$$2 \rightarrow 3 \rightarrow 1$$

num

low

- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

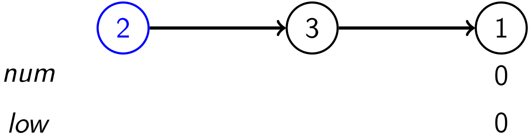
- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.



num	0
low	0

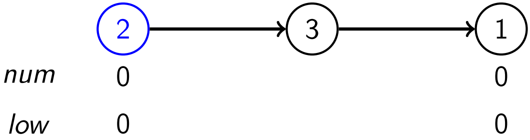
- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.



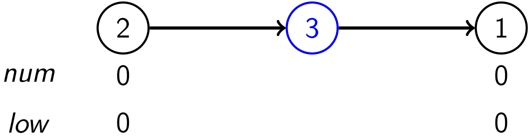
- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.



- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.



- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

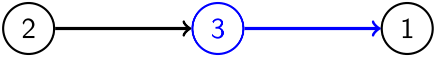
- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.

$$2 \rightarrow 3 \rightarrow 1$$

num	0	1	0
low	0	1	0

- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

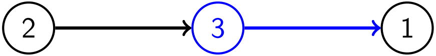
- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.



num	0	1	0
low	0	1	0

- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

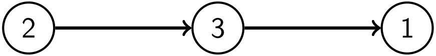
- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.



num	0	1	0
low	0	0	0

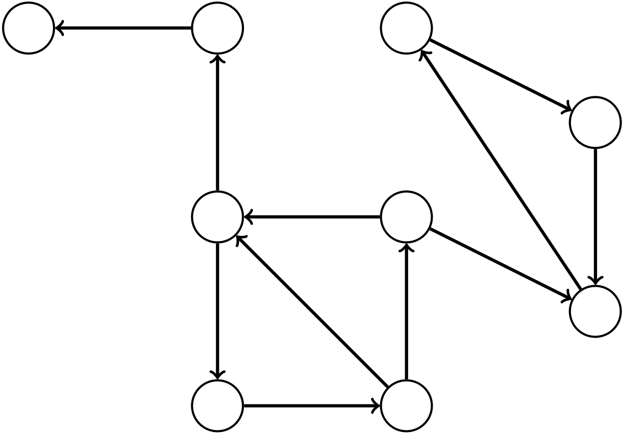
- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

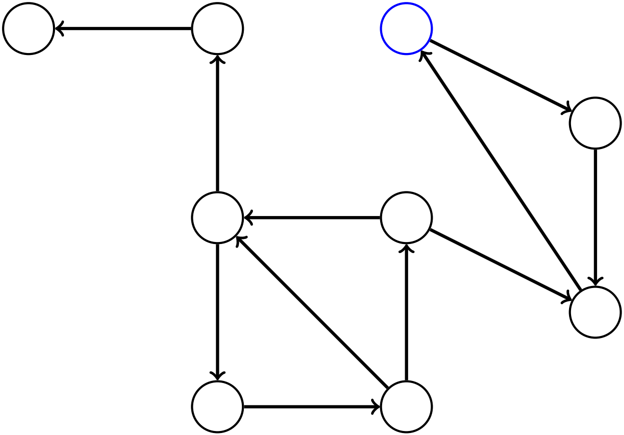
- Við þurfum að passa okkur þegar við uppfærum u_{low}.
- Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.

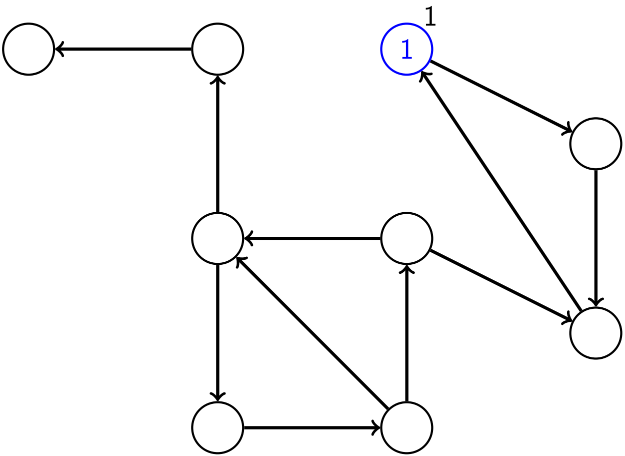


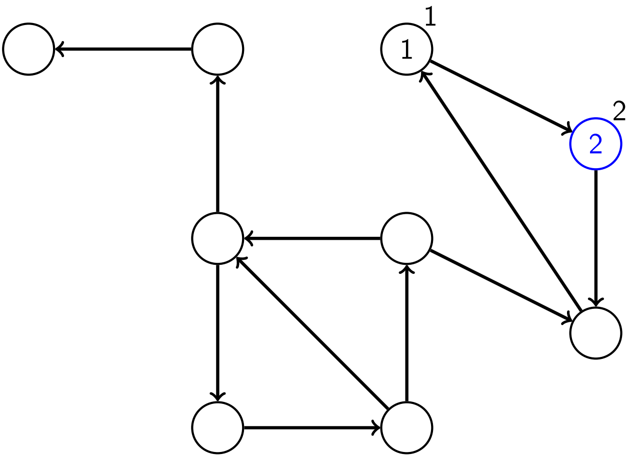
num	0	1	0
low	0	0	0

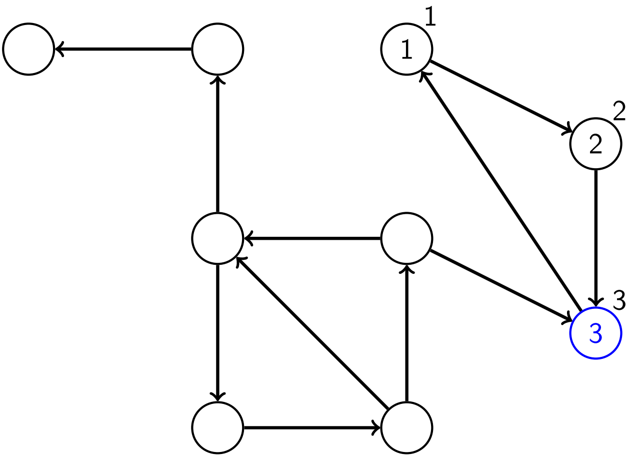
- Hverjir eru ströngu samhengisþættir netsins?
- Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið stranga samhángisáttinn fyrir.

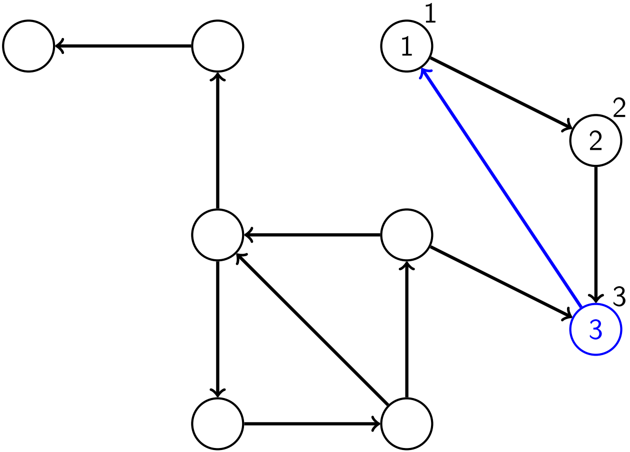


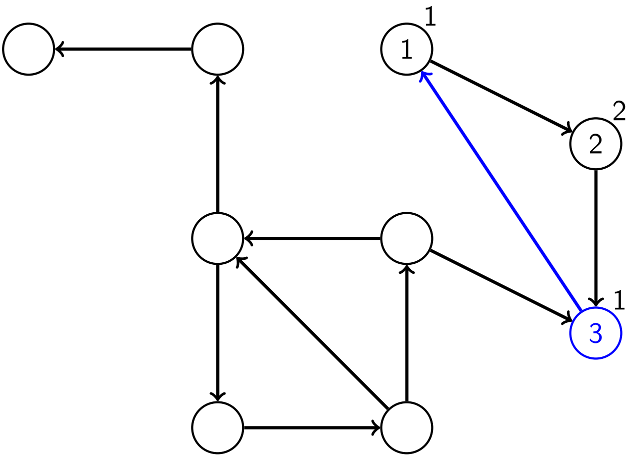


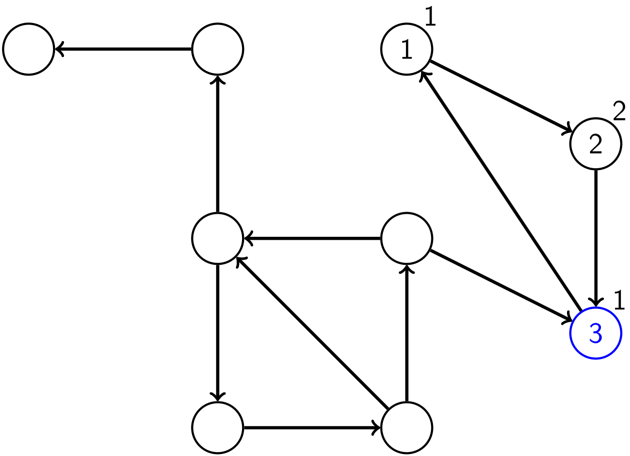

1
1

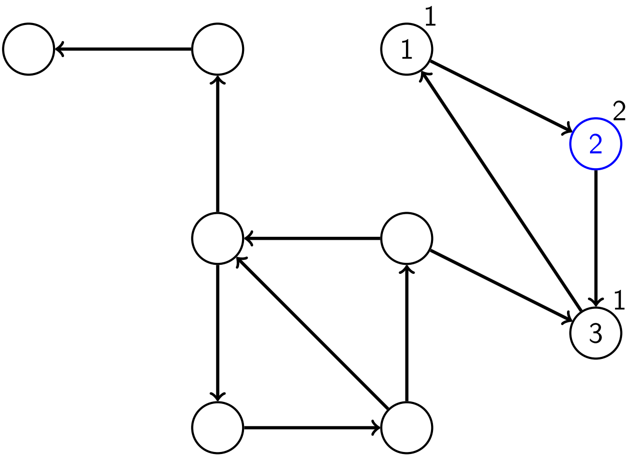

1
1
2
2

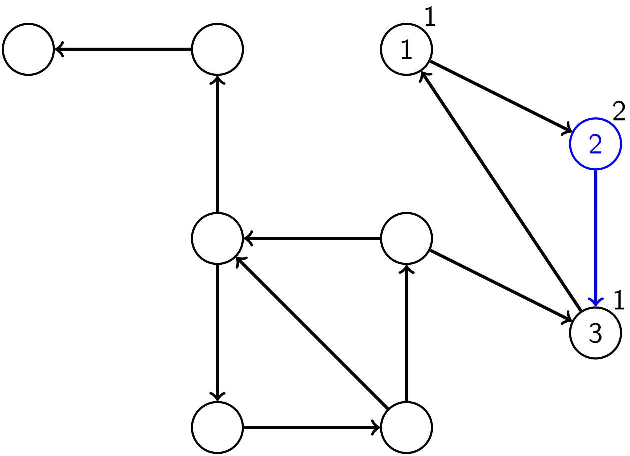

1
1
2
2
3
3

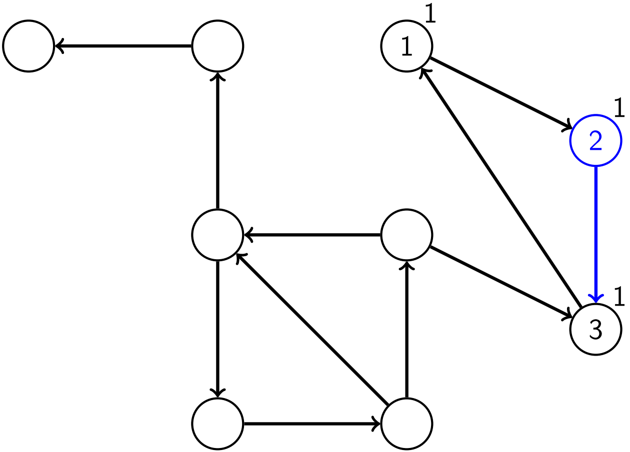

1
1
2
2
3
3

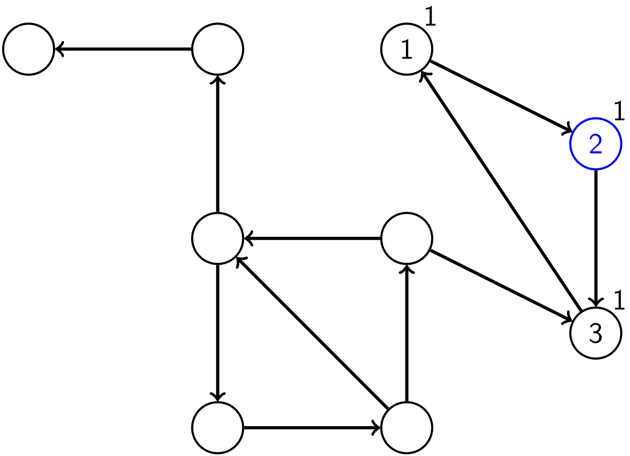

1
1
2
2
1
3

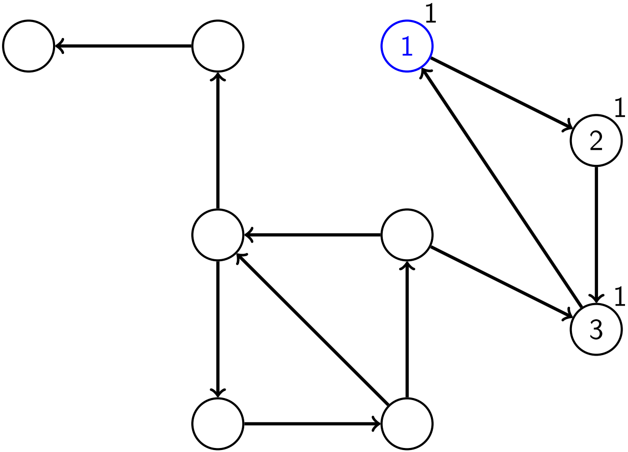

1
1
2
2
1
3

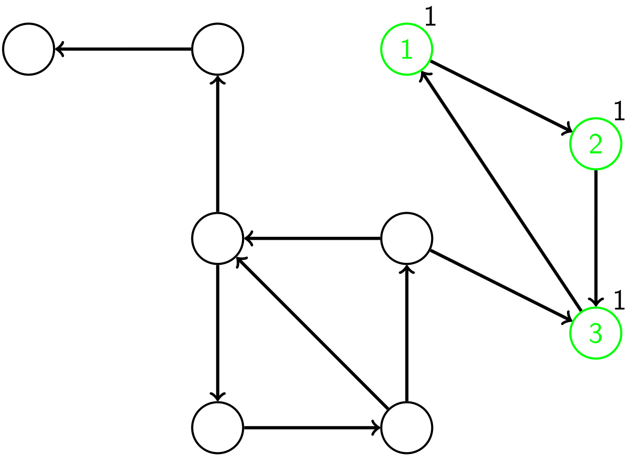

1
1
2
2
1
3

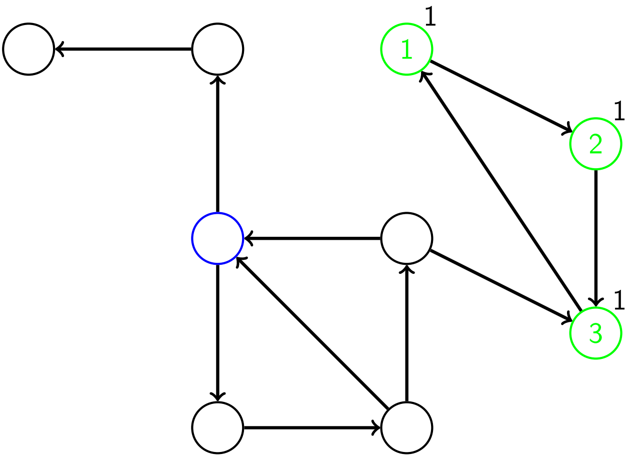

1
1
2
2
1
3

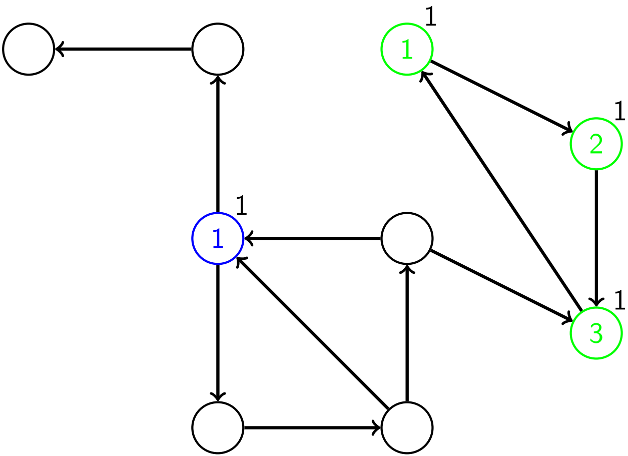

1
1
1
2
1
3
1

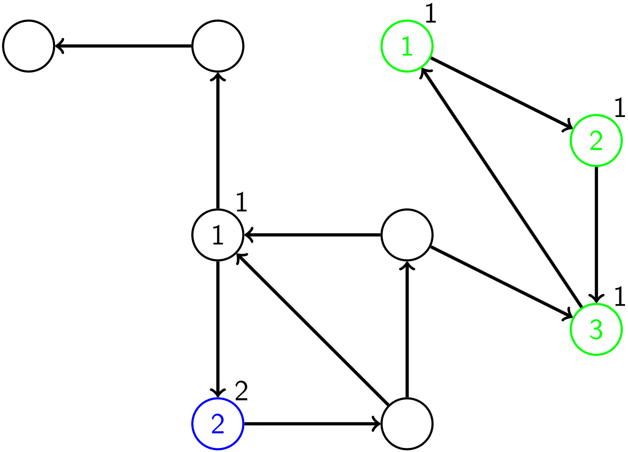

1
1
1
2
1
3

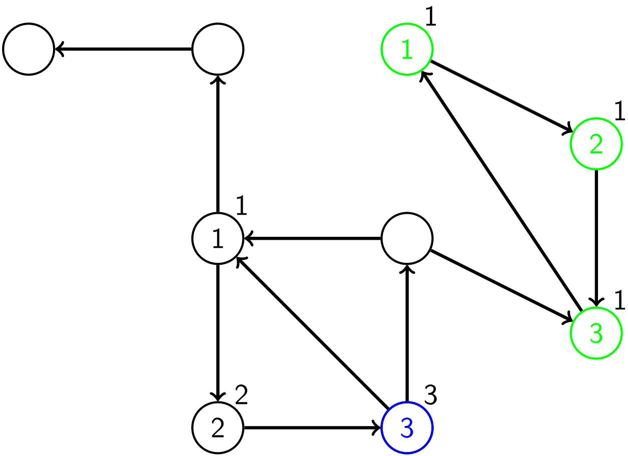

1
1
1
2
1
3
1

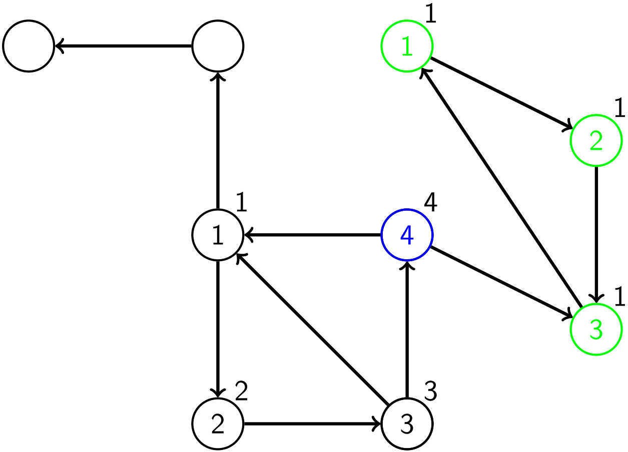

1
1
1
2
1
3
1

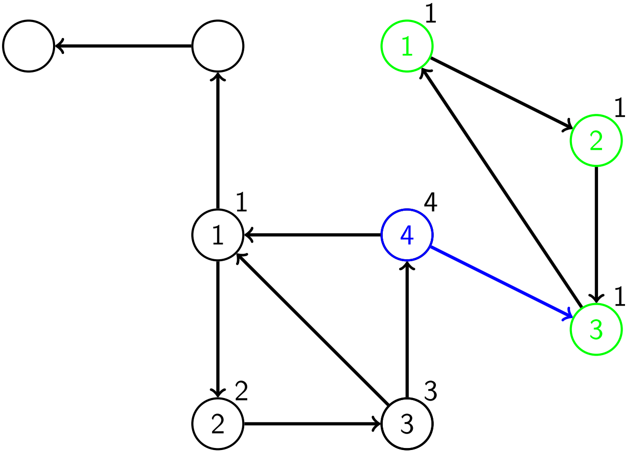

1
1
1
2
1
3
1

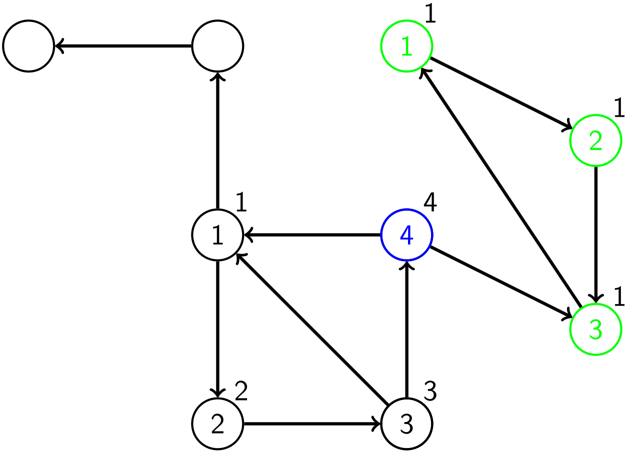

1
1
1
2
1
1
1
3

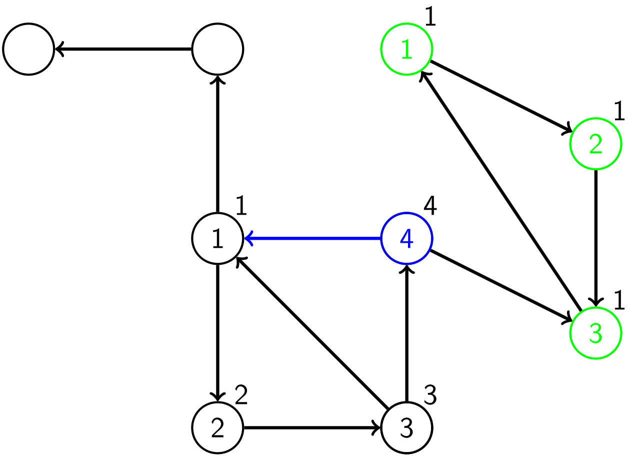

1
1
1
2
1
1
2
1
2
3
1

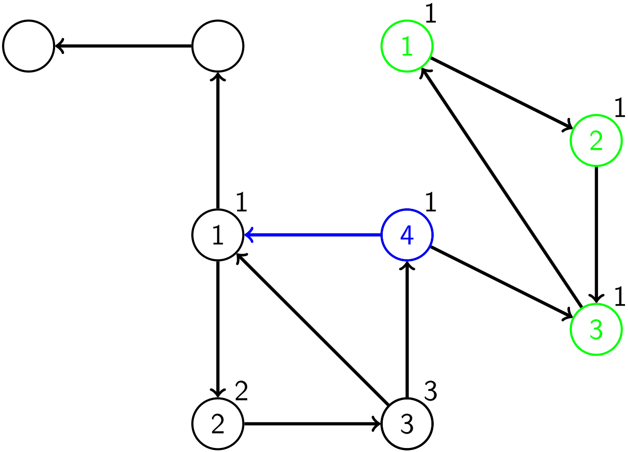

1
1
1
2
1
1
2
3
3
1
3


1
1
1
2
1
4
1
4
1
2
2
3
3
3
1

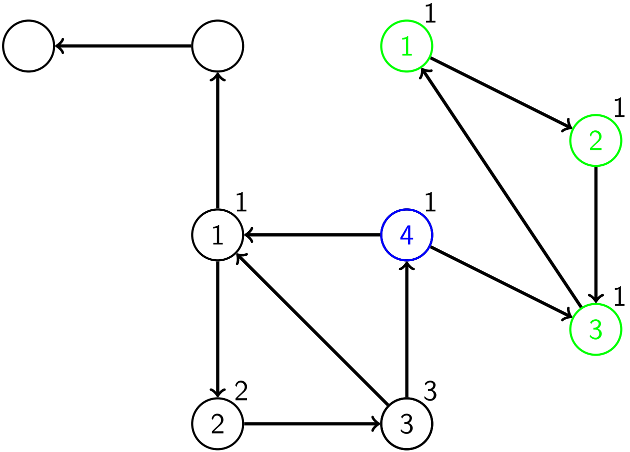

1
1
1
1
2
1
4
1
4
3
1
2
2
3
3
3

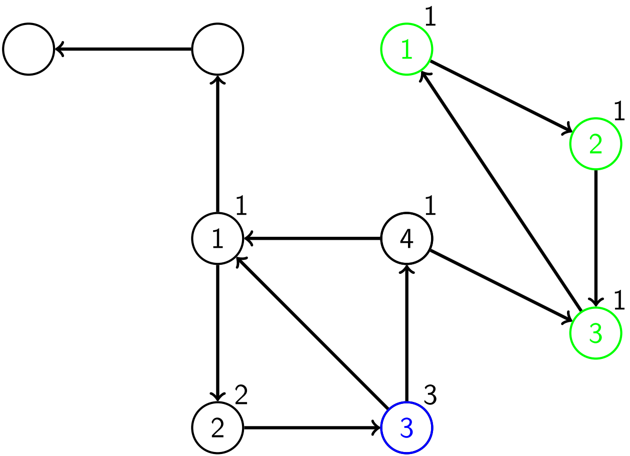

1
1
1
1
1
2
4
4
1
2
2
3
3
3

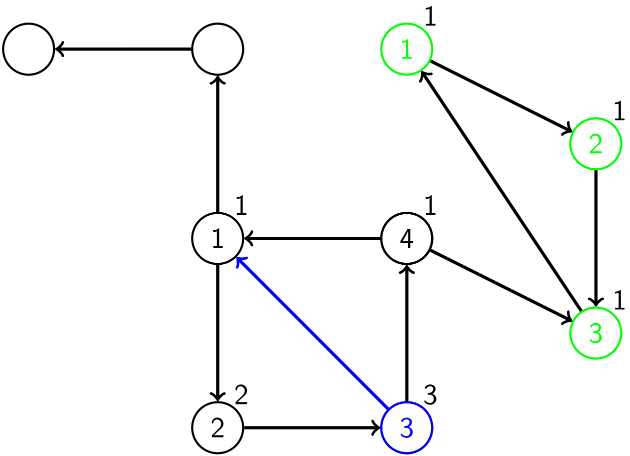

1
1
1
1
2
1
4
4
1
3
2
2
3
3



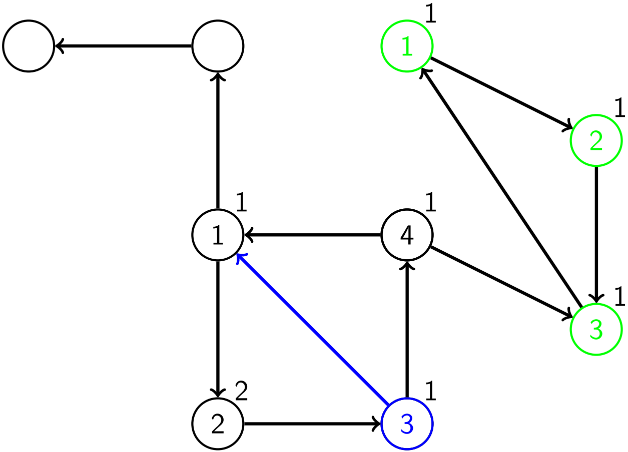
1
1
1
2
1
1
1
4
1
1
2
2
3
3
3

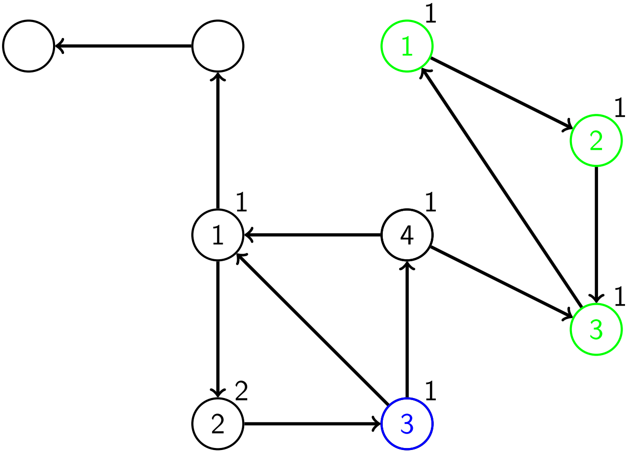

1
1
1
2
1
1
1
4
1
2
2
3
3
3
1
2
3

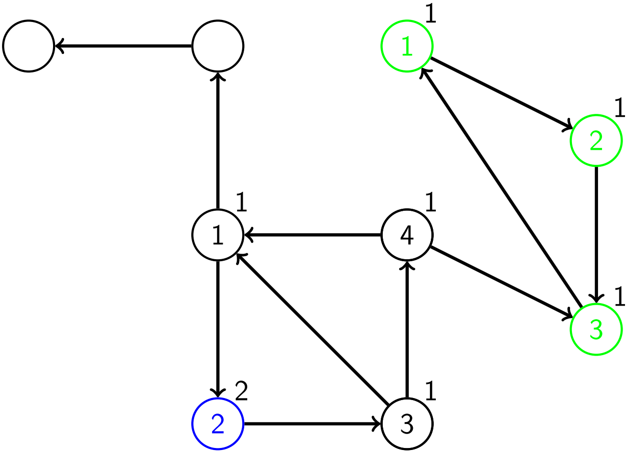

1
1
1
2
1
3
1
1
1
4
2
2
3
3



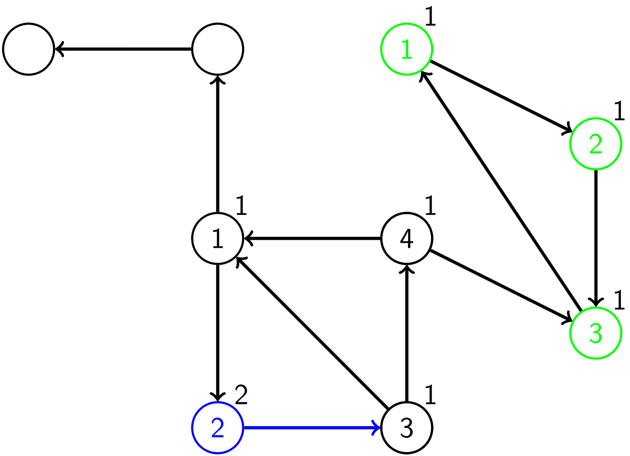
1
1
1
2
1
1
4
1
3
2
2
3
3

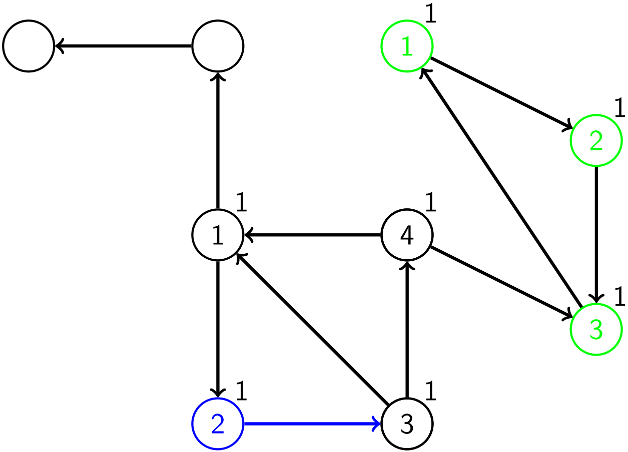

1
1
1
2
1
1
1
1
4
1
2
3
1
2
3
1

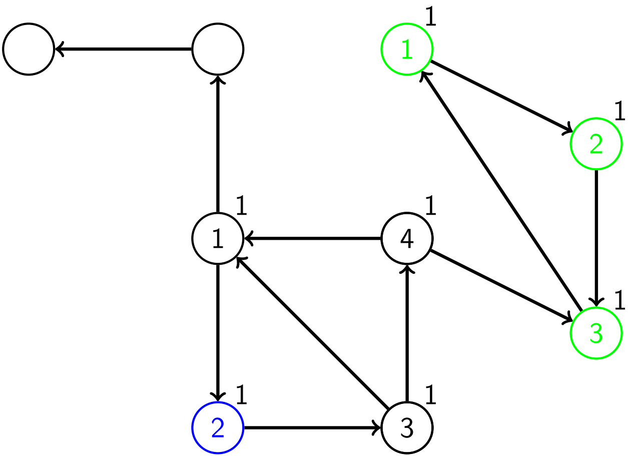

1
1
1
2
1
3
1
1
1
4
1
2
2
3
1
3

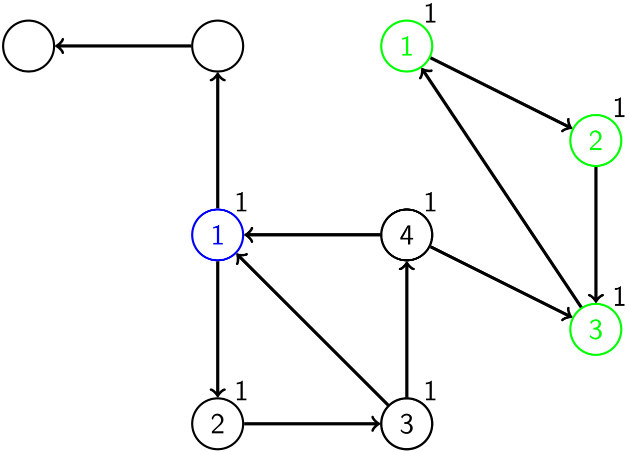


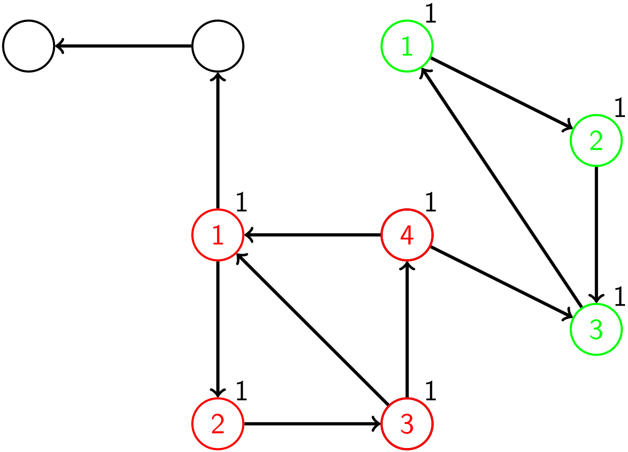
1
1
1
2
1
2
3
1
1
1
4
1
2
2
3
1


1
1
1
2
1
1
4
1
1
3
1
2
2
3
1

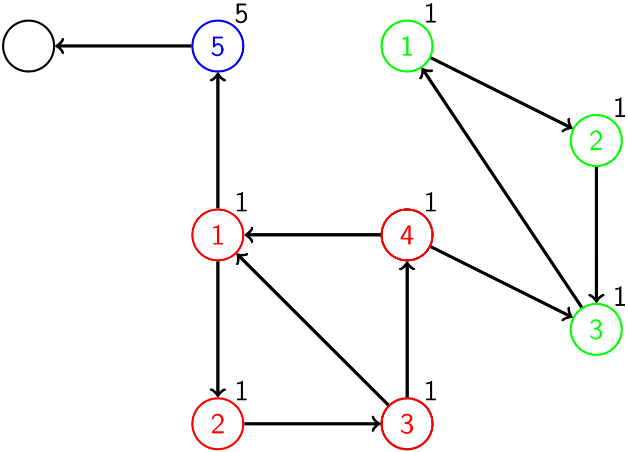

1
1
1
1
2
1
3
1
4
1
1
2
1
3
1
2
3

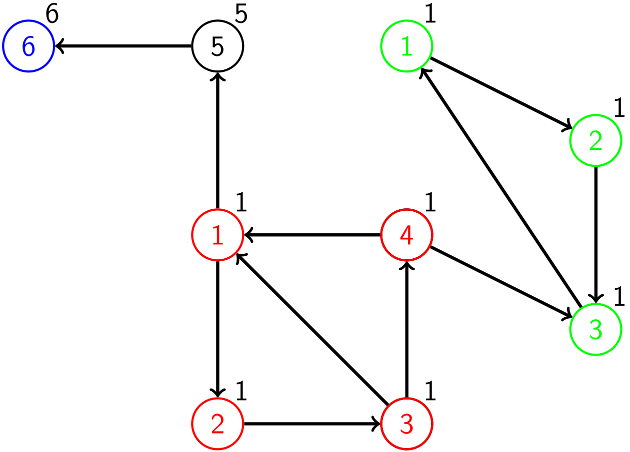

1
1
1
2
1
3
1
1
1
4
1
2
1
3

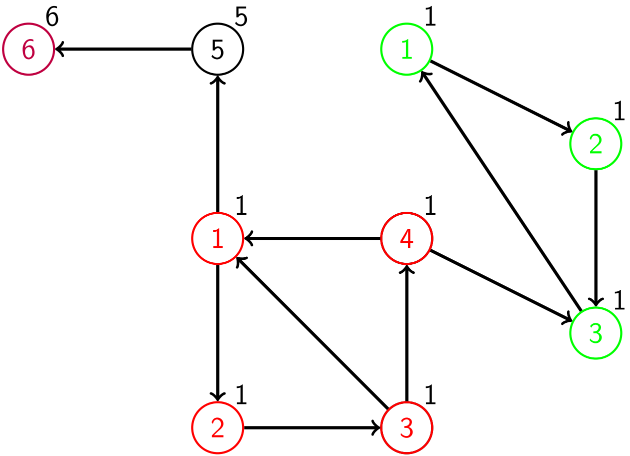

1
1
1
2
1
1
1
4
1
3
1
2
1
3

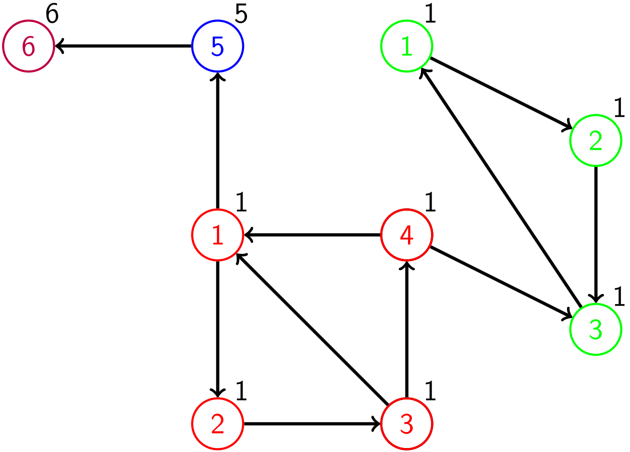


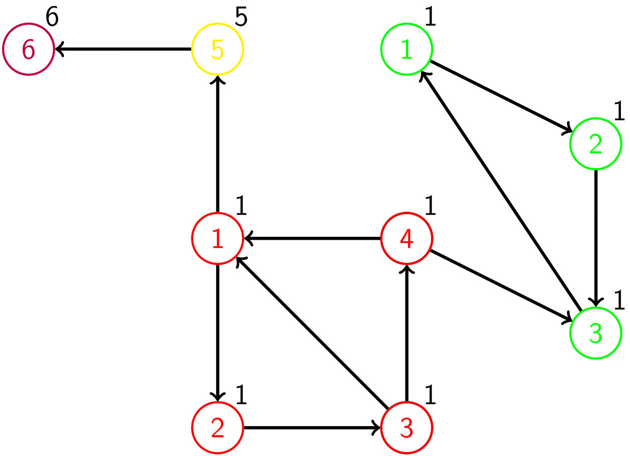
1
1
1
2
1
3
1
1
1
4
1
2
1
3


5
5
1
1
1
2
1
1
4
1
1
3
1
2
1
3


6
6
5
5
1
1
1
2
1
4
1
1
3
1
2
1
3
1


6
6
5
5
1
1
1
2
1
3
1
1
1
4
1
2
1
3


6
5
1
6
5
1
1
2
1
1
4
1
3
1
2
1
3


6
6
5
5
1
1
1
2
1
3
1
1
1
4
1
2
1
3
1

```
int dfs(vvi &g, int u, int d, int *s, int *a, int *l)
{
    l[u] = d, s[++s[0]] = u;
    int i, x, z = d;
    for (i = 0; i < g[u].size(); i++)
    {
        x = g[u][i];
        if (l[x] == -1) d = dfs(g, x, d + 1, s, a, l);
        if (a[x] == -1) l[u] = min(l[u], l[x]);
    }
    if (l[u] == z) while (a[u] == -1) a[s[s[0]--]] = u;
    return d;
}

void scc(vvi &g, int *a)
{
    int i, n = g.size(), s[n + 1], l[n];
    for (i = 0, s[0] = 0; i < n; i++) l[i] = a[i] = -1;
    for (i = 0; i < n; i++) if (l[i] == -1) dfs(g, i, 0, s, a, l);
}
```

- Þar sem við leitum bara einu sinni í netinu með dýptarleit fæst að þetta reiknirit er $\mathcal{O}(E + V)$.
- Við köllum þetta reiknrit, ásamt því sem finnur liðhnúta og brýr, reiknrit Tarjans.