

Reiknirit Tarjans

Bergur Snorrason

February 26, 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 *samhengispátt* í netinu sem jafngildisflokka þessara vensla.
- ▶ Samhengispáttur 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 samhengispátt.

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

```

6 vi v;
7 void dfs(vvi& g, int x, int c)
8 {
9     int i;
10    v[x] = c;
11    for (i = 0; i < g[x].size(); i++) if (v[g[x][i]] == -1)
12        dfs(g, g[x][i], c);
13 }
14
15 int main()
16 {
17     int i, j, n, m, c = 0, x, y;
18     cin >> n >> m;
19     vvi g(n);
20     for (i = 0; i < m; i++)
21     {
22         cin >> x >> y;
23         x--, y--;
24         g[x].push_back(y);
25         g[y].push_back(x);
26     }
27     v = vi(n, -1);
28     for (i = 0; i < n; i++) if (v[i] == -1) dfs(g, i, c++);
29     printf("Fjöldi samhengisþátta er %d.\n", c);
30     for (i = 0; i < n; i++)
31         printf("Hnútur %d er í samhengisþætti %d.\n", i + 1, v[i] + 1);
32     return 0;
33 }

```

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

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

```

5  int uf_find(int *p, int x)
6  {
7      return p[x] < 0 ? x : (p[x] = uf_find(p, p[x]));
8  }
9
10 void uf_join(int *p, int x, int y)
11 {
12     int rx = uf_find(p, x), ry = uf_find(p, y);
13     if (rx == ry) return;
14     if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
15     else p[rx] += p[ry], p[ry] = rx;
16 }
17
18 void uf_init(int *p, int n)
19 {
20     for (int i = 0; i < n; i++) p[i] = -1;
21 }
22
23 int main()
24 {
25     int i, j, n, m, c = 0, x, y;
26     scanf("%d%d", &n, &m);
27     int p[n], v[n];
28     uf_init(p, n);
29     for (i = 0; i < m; i++)
30     {
31         scanf("%d%d", &x, &y);
32         x--, y--;
33         uf_join(p, x, y);
34     }
35     for (i = 0; i < n; i++) v[i] = 0;
36     for (i = 0; i < n; i++) v[uf_find(p, i)] = 1;
37
38     printf("Fjöldi samhengisþátta er %d.\n", c);
39     for (i = 0; i < n; i++)
40         printf("Hnútur %d er í samhengisþætti %d.\n", i + 1, uf_find(p, i));
41     return 0;
42 }

```

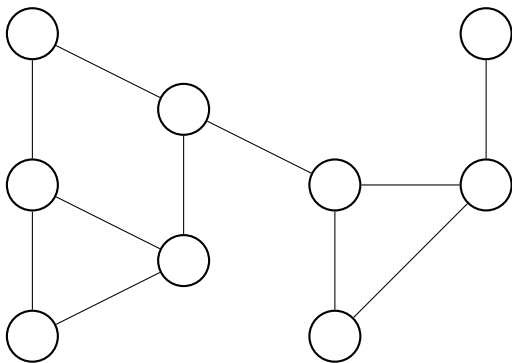
- ▶ Ef við tölum um að fjarlægja hnút úr neti þá er átt við að hnúturinn ásamt öllum leggjum til og frá honum eru fjarlægðir.
- ▶ Gerum ráð fyrir að við höfum net G og látum G_u tákna netið þar sem hnútur u hefur verið fjarlægður.
- ▶ Við segjum að hnútur u sé *liðhnútur* (e. *articulation point*) ef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

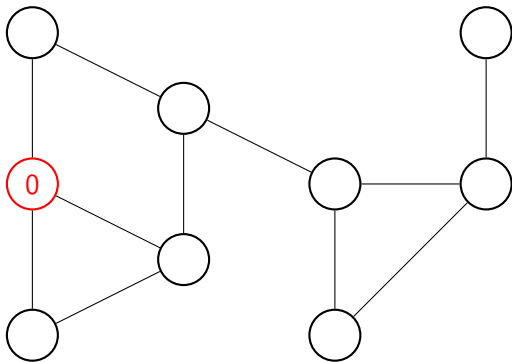
- ▶ Til að fjarlægja legg úr neti nægir að fjarlægja legginn.
- ▶ Táknum þá netið G án leggsins e með G_e .
- ▶ Leggur e eru sagður vera *brú* (e. *bridge*) ef G hefur færri samhengispætti en G_e .
- ▶ Með öðrum orðum er hnútur u liðhnútur (leggur e brú) ef til eru hnútar v_1 og v_2 í sama samhengispætti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

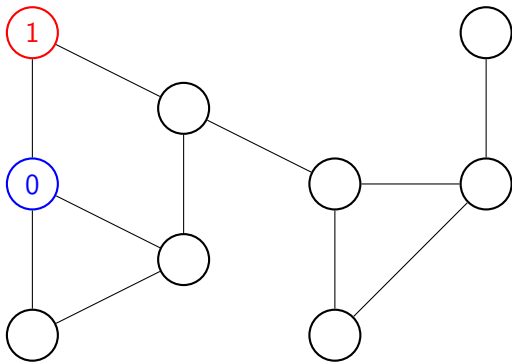
- ▶ Ein leið til að finna alla liðhnúta er að telja fyrst samhengisþætti netsins, fjarlægja hnút, telja samhengisþætti og endurtaka fyrir alla hnúta.
- ▶ Þar sem við þurfum að finna alla samhengisþætti $2V + 1$ neta er þessi aðferð með tímaflækju $\mathcal{O}(V^2 + VE)$.
- ▶ Samskonar aðferð til að finna 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.

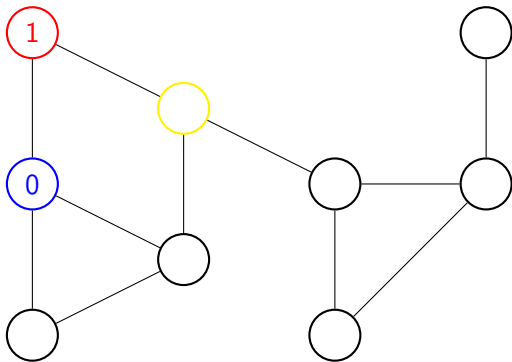
- ▶ Gerum ráð fyrir að netið okkar sé samanhangandi.
- ▶ Ef svo er ekki getum við beitt þessari aðferð á hvern samhengisþátt.
- ▶ Veljum einhvern hnút og framkvæmum dýptarleit frá honum.
- ▶ Skilgreinum svo tvær breytur fyrir hvern hnút u út frá þessari dýptarleit, u_{low} og u_{num} .
- ▶ Talan u_{num} segir hversu mörg skref í leitinni 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.

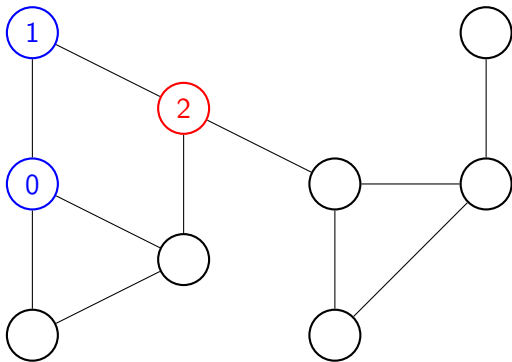
- ▶ Gerum ráð fyrir að leitin okkar fari úr hnút u í hnút v með legg e .
- ▶ Ef $v_{low} > u_{num}$ þá er e brú.
- ▶ Þetta þýðir að eina leiðin frá v til u er í gegnum leggin e .
- ▶ Ef $v_{low} \geq u_{num}$ gildir fyrir einhvern nágranna u þá er u liðhnútur.
- ▶ Þetta þýðir að eina leiðin frá v í fyrri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

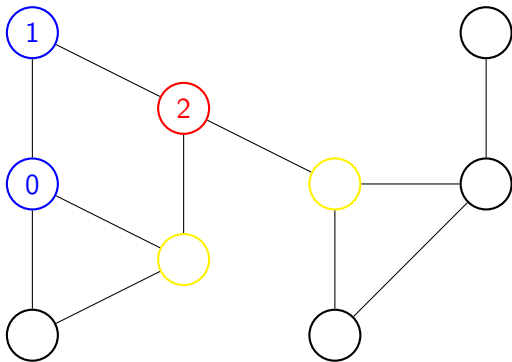


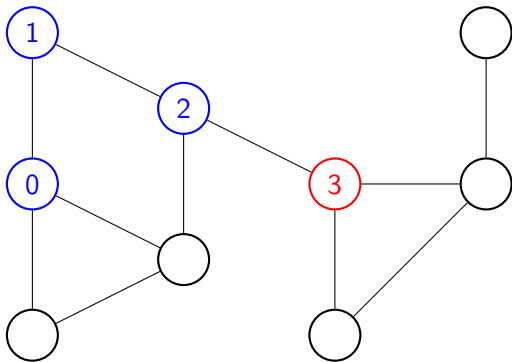


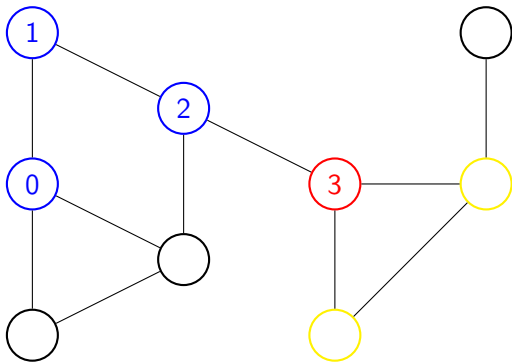


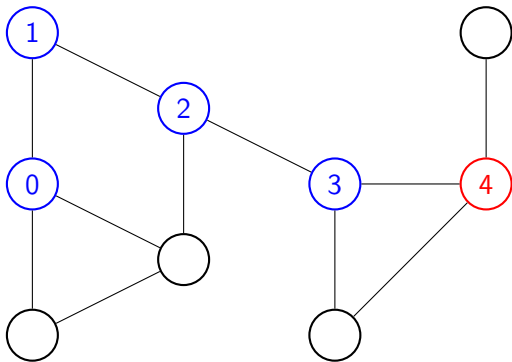


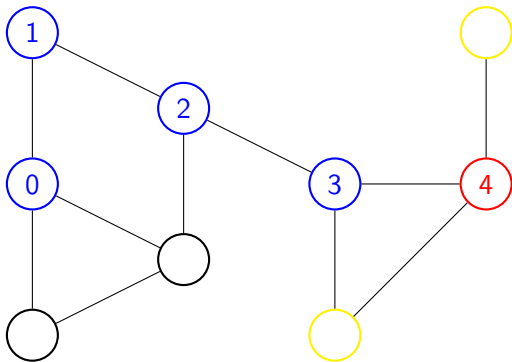


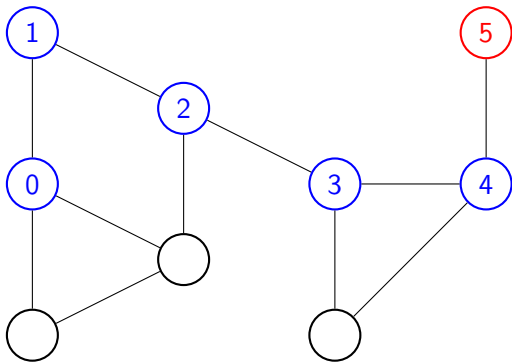


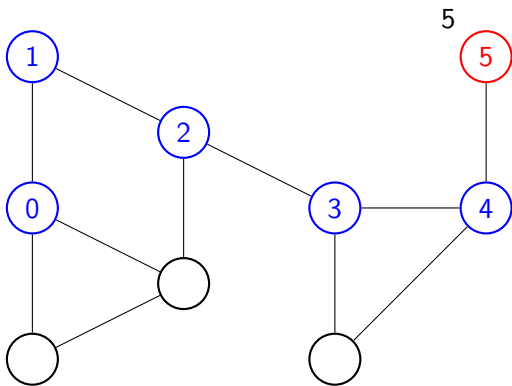


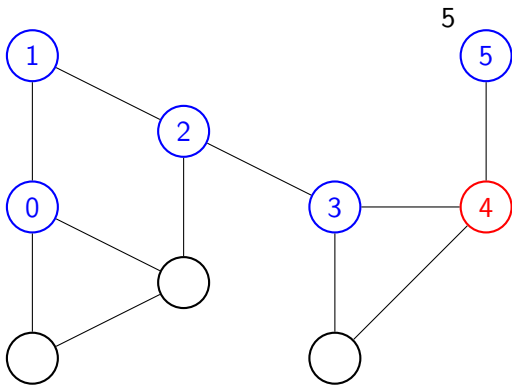


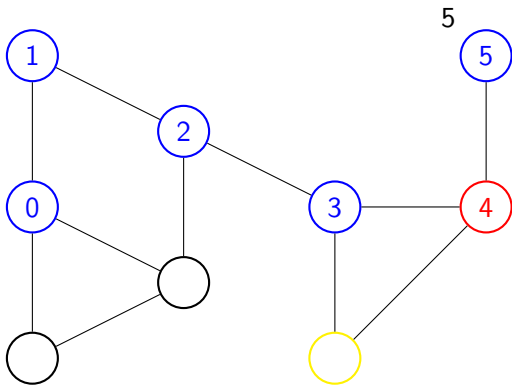


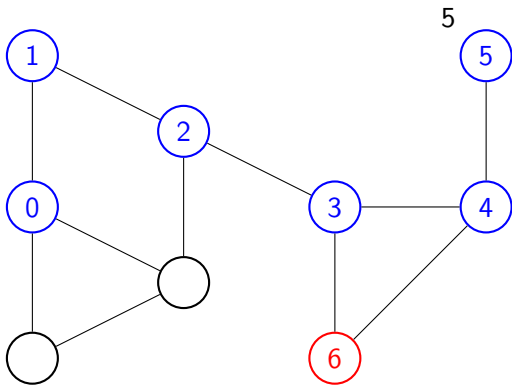


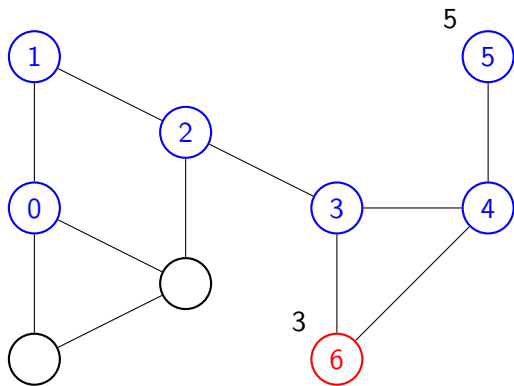


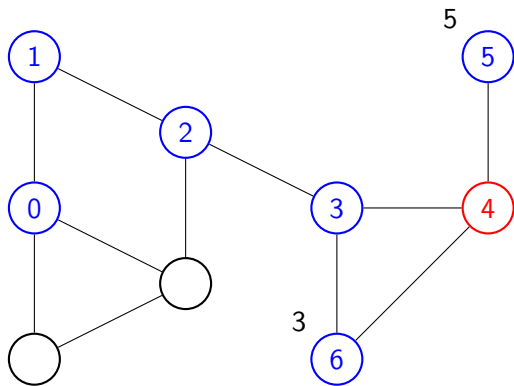


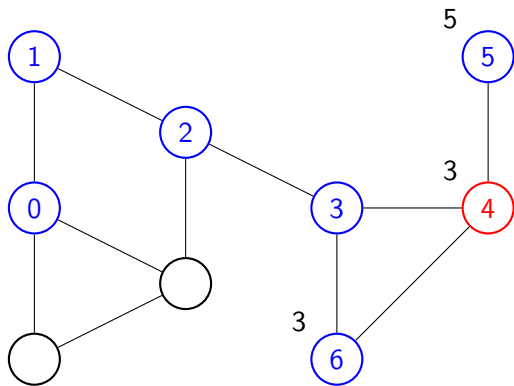


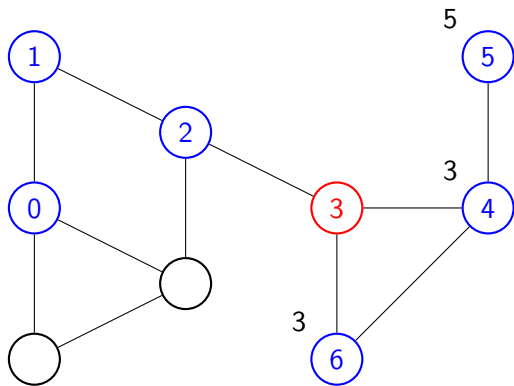


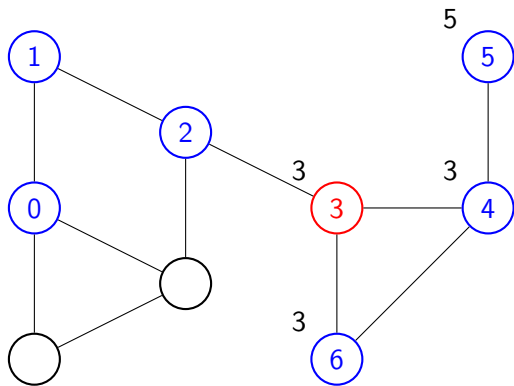


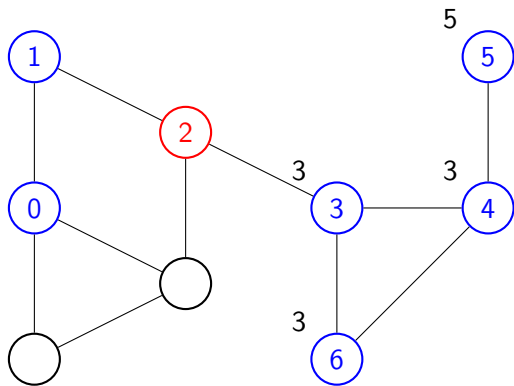


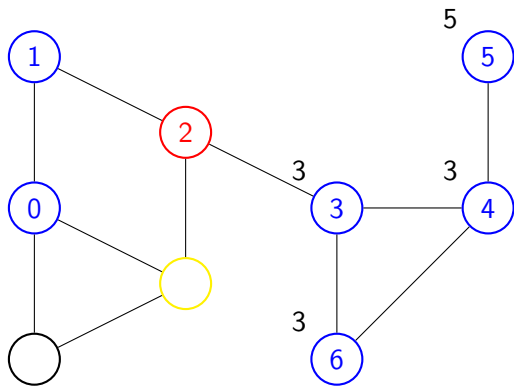


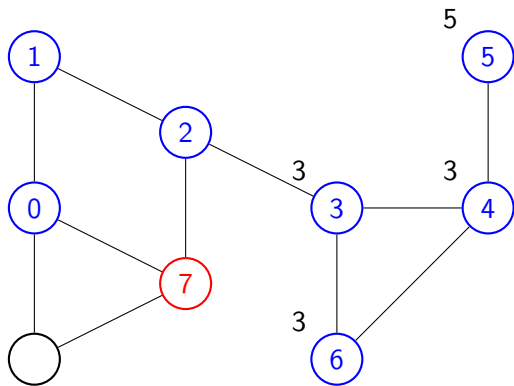


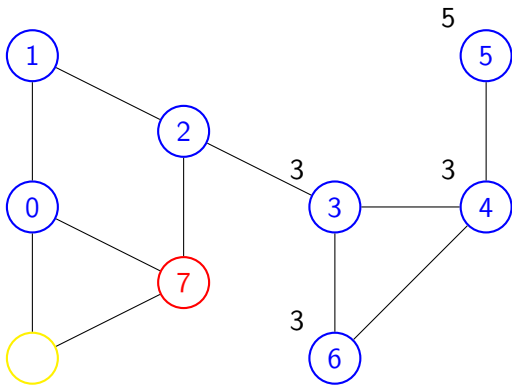


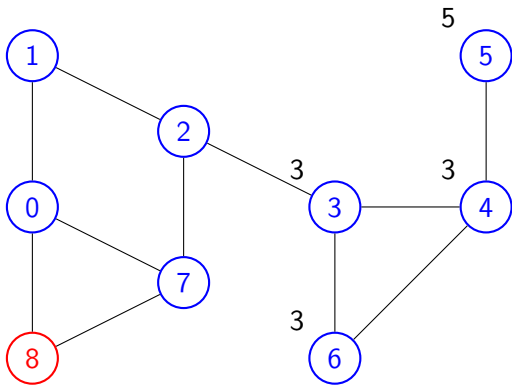


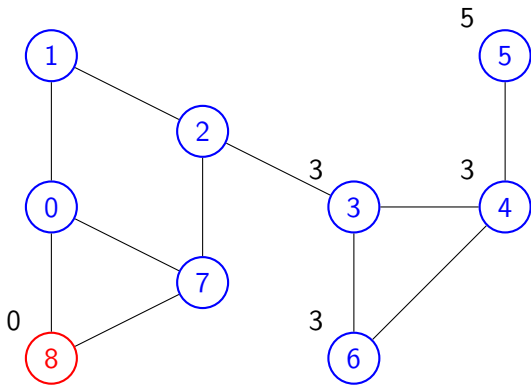


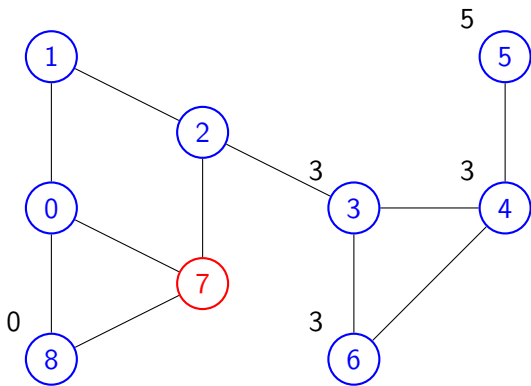


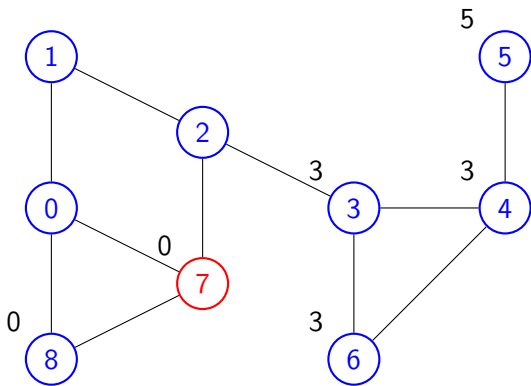


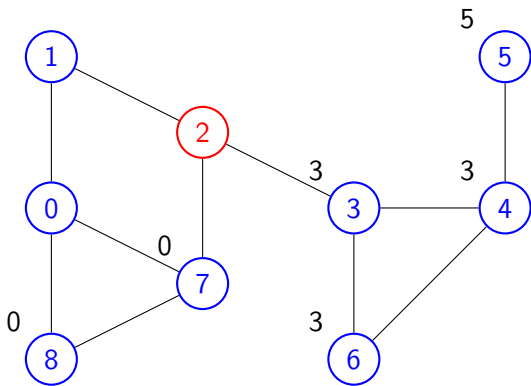


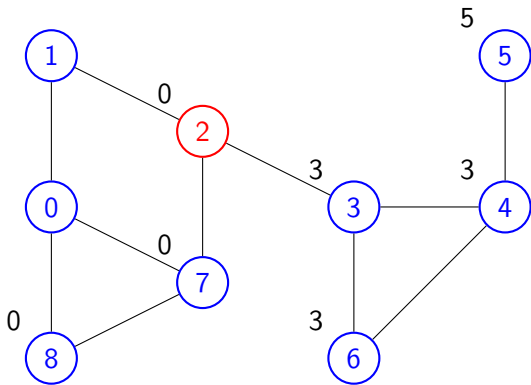


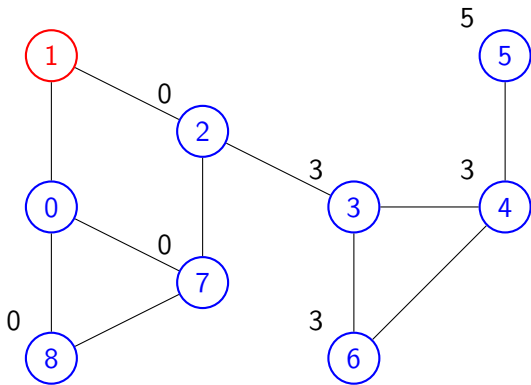


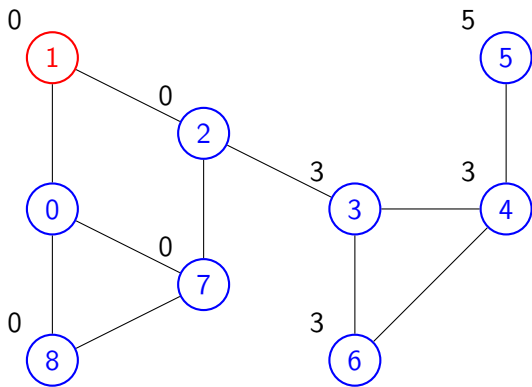


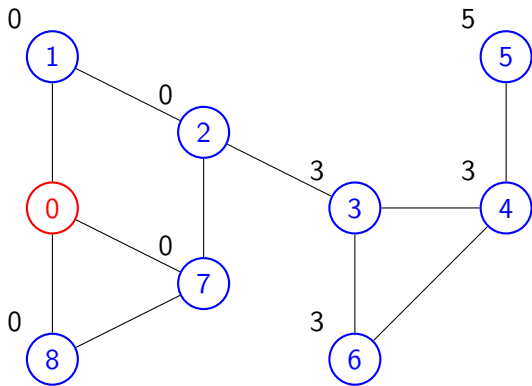


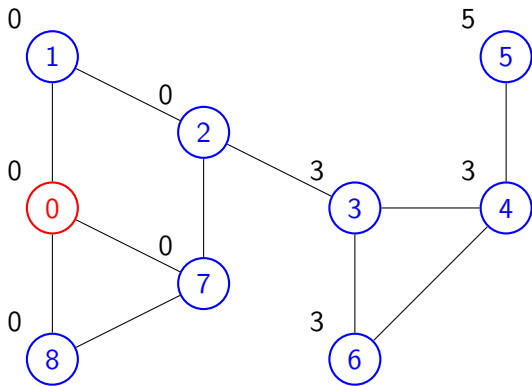


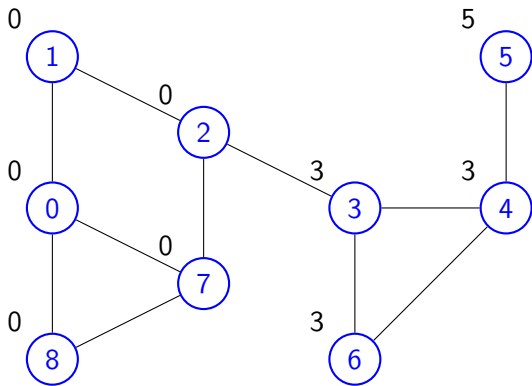


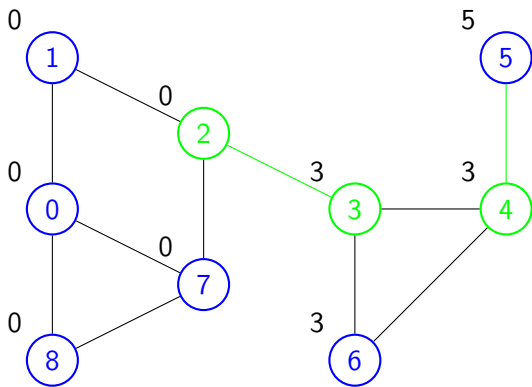












```

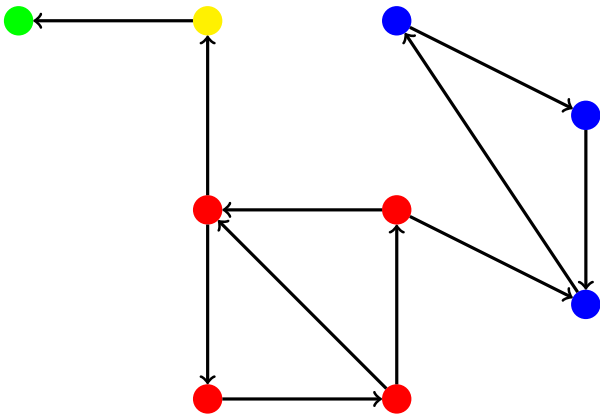
11 int dfs(const vvi &g, int u, int p, int d)
12 {
13     int i, x, c = 0, z = d, y; v[u] = d;
14     for (i = 0; i < g[u].size(); i++) if (g[u][i] != p)
15     {
16         x = g[u][i];
17         if (v[x] == -1)
18         {
19             y = dfs(g, x, u, d + 1), c++;
20             z = min(z, y);
21             if (y > v[u]) bri.push_back(ii(u, x));
22             if (p != -1 && y >= v[u]) a[u] = 1;
23         }
24         else z = min(z, v[x]);
25     }
26     if (p == -1 && c > 1) a[u] = 1;
27     return z;
28 }
29
30 void cpb(const vvi &g)
31 {
32     cp.clear(), bri.clear();
33     for (int i = 0; i < g.size(); i++) v[i] = -1, a[i] = 0;
34     for (int i = 0; i < g.size(); i++) if (v[i] == -1) dfs(g, i, -1, 0);
35     for (int i = 0; i < g.size(); i++) if (a[i]) cp.push_back(i);
36 }

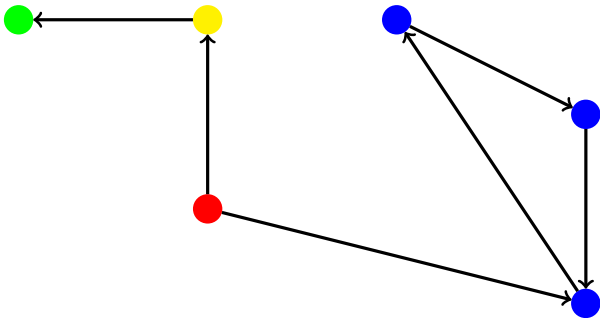
```

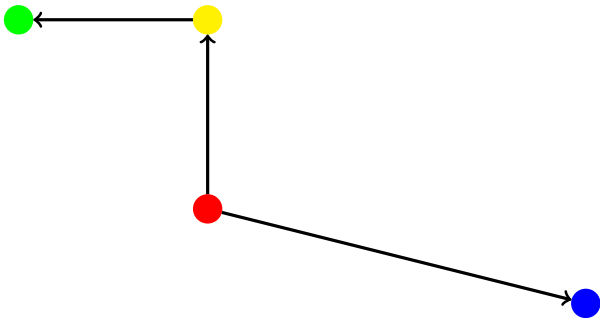
- ▶ Tímaflækjan er $\mathcal{O}(E + V)$ því það er tímaflækja dýptarleitar.

- ▶ 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 stefnt net.

- ▶ Takið eftir að ef netið inniheldur rás þá eru allir hnútar í rásinni í sama samhengispætti.
- ▶ Einnig gildir að ef netið inniheldur enga rás er hver hnútur sinn eigin samhengispáttur.
- ▶ 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 samhengispætti séu í raun sami hnúturinn og verður leggur milli samhengispátta ef vegur liggur milli einhverja hnúta í samhengispáttunum sem fer ekki í annan samhengispátt.
- ▶ Við köllum þetta net *herpingu* (e. *contraction*) upprunalega netsins.

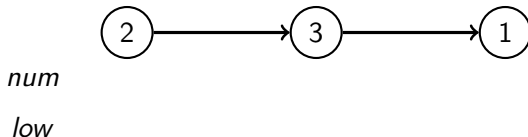






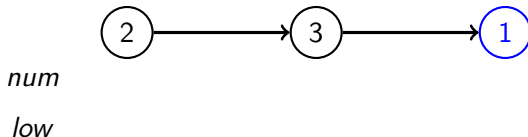
- ▶ Til að finna herpinguna þurfum við fyrst að finna samhengisþættina.
- ▶ Við getum breytt lítilega forritinu sem við vorum með áðan til að finna samhengisþætti stefnds nets.
- ▶ Við getum skoðað hvort $u_{low} = u_{num}$ á leiðinni upp úr endurkvæmninni.
- ▶ Ef svo er þá er u fyrsti hnúturinn sem 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.



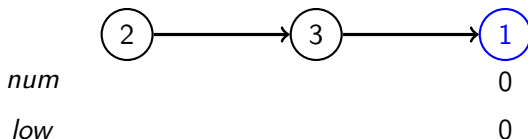
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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



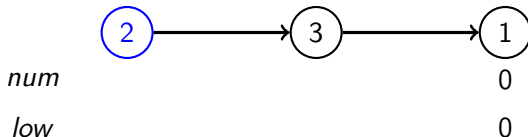
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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



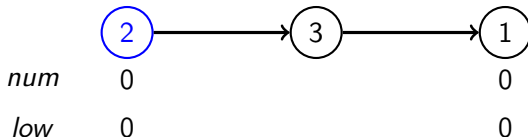
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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



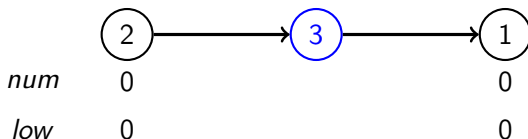
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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



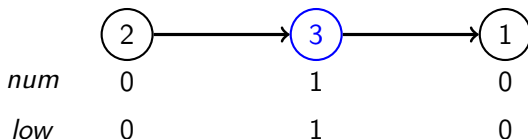
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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



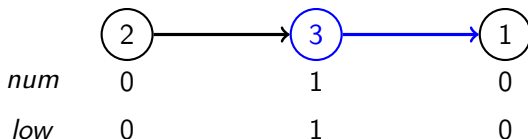
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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



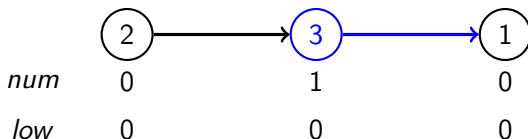
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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



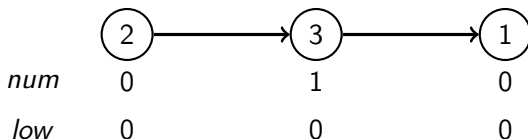
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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

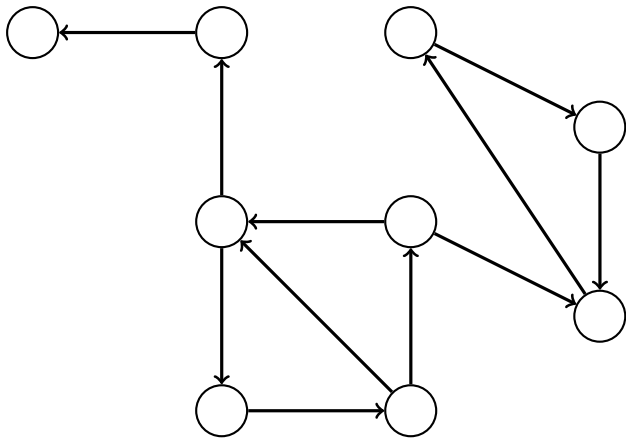


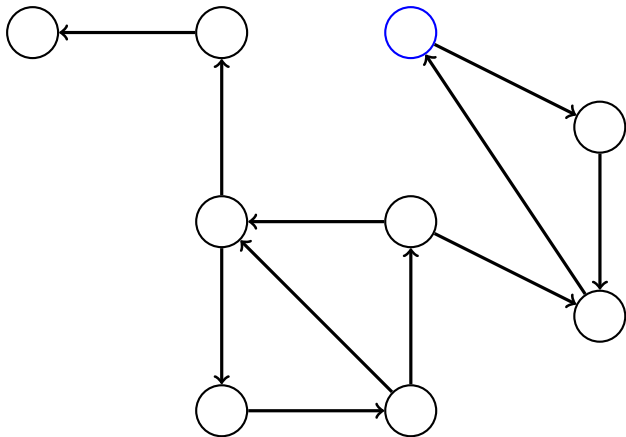
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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

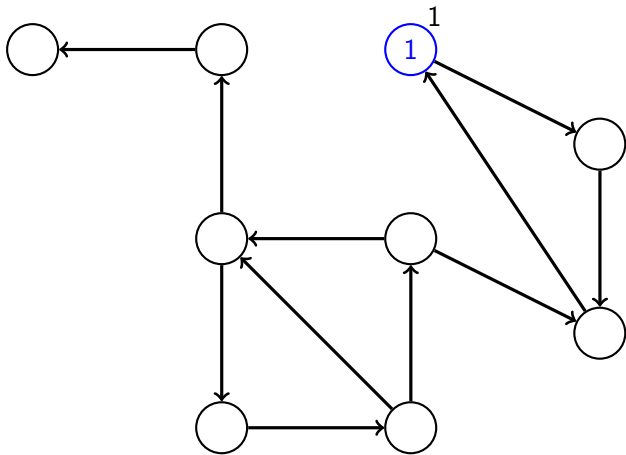
- ▶ Við þurfum að passa okkur þegar við uppfærum u_{low} .
- ▶ Ef við íhugum ekki röðina sem við veljum upphafshnúta getum við fengið rangar niðurstöður.

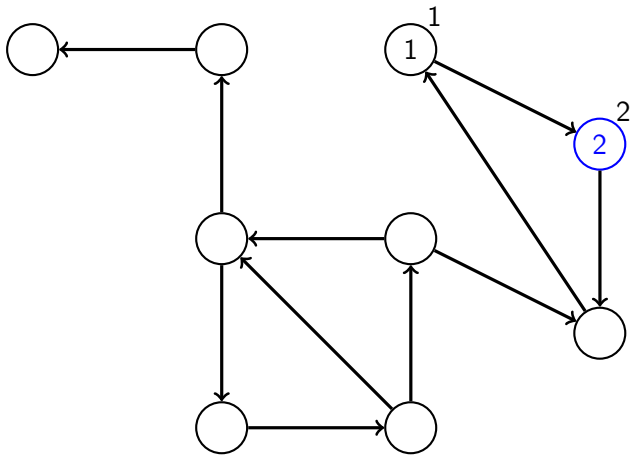


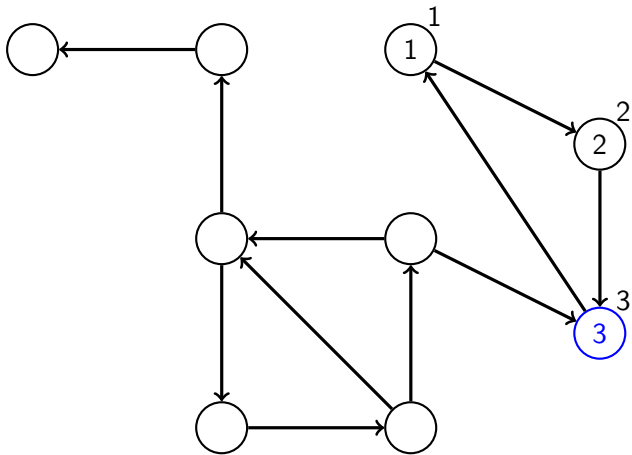
- ▶ Hverjir eru ströngu samhengisþættir netsins?
- ▶ Hverjir verða ströngu samhengisþættir netsins samkvæmt glærunni á undan?
- ▶ Afhverju gerðist þetta ekki þegar við vorum að finna liðhnúta og brýr?
- ▶ Einföld leið til að laga þetta er að uppfæra bara u_{low} með nágrönnum sem við höfum ekki fundið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

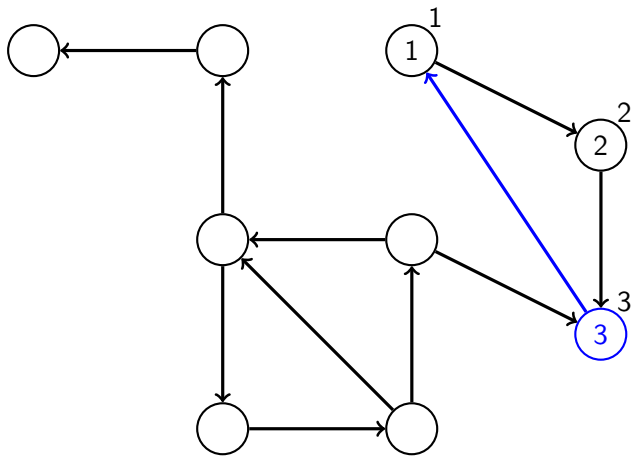


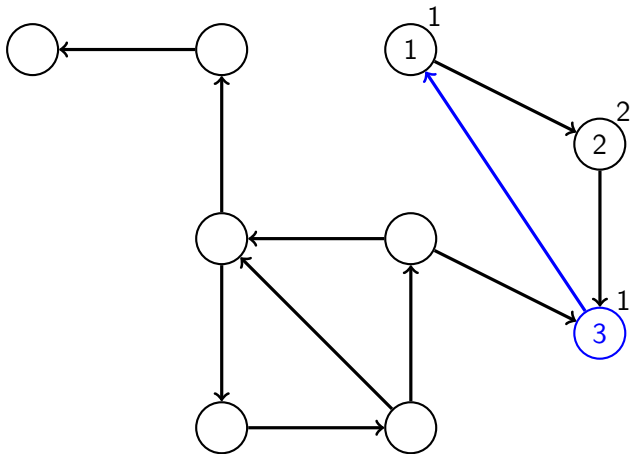


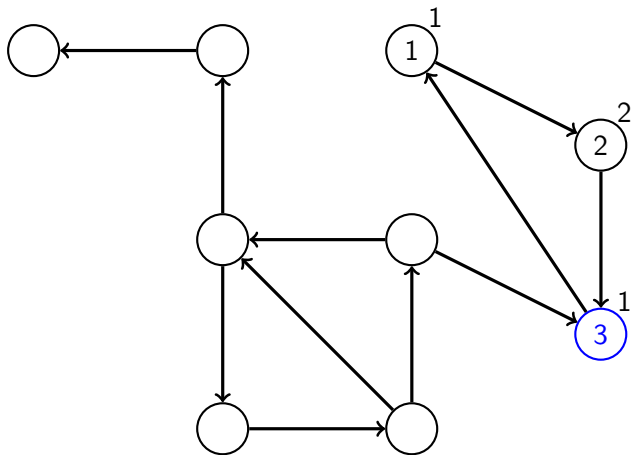


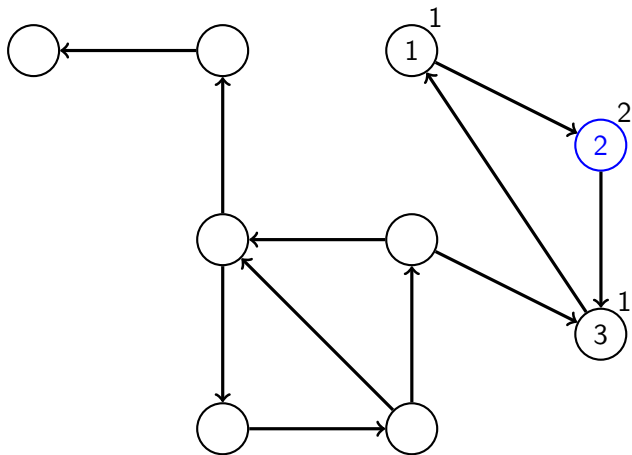


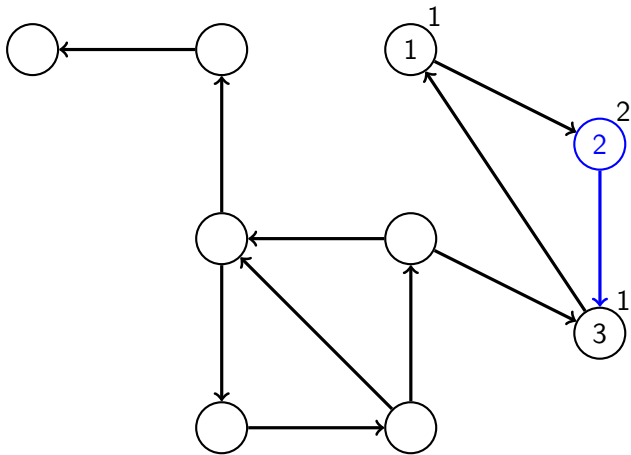


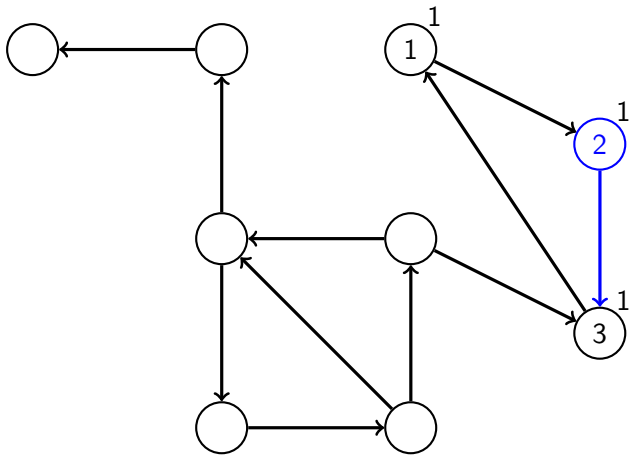


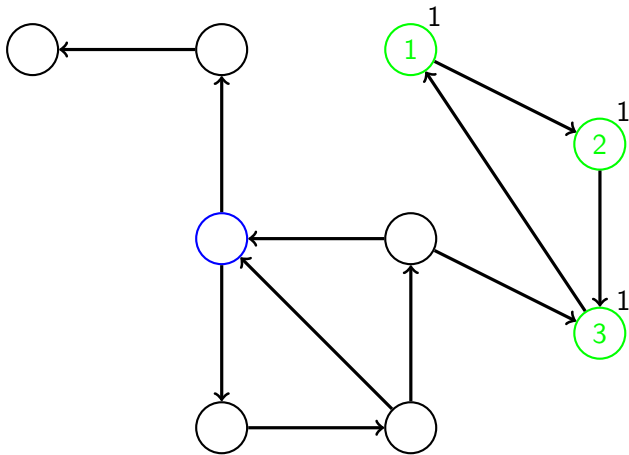


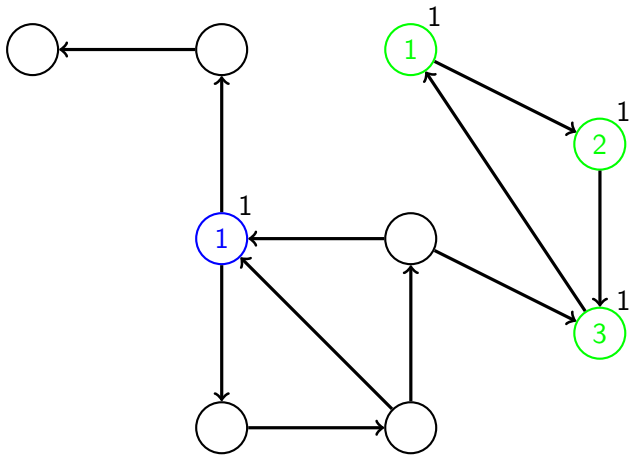


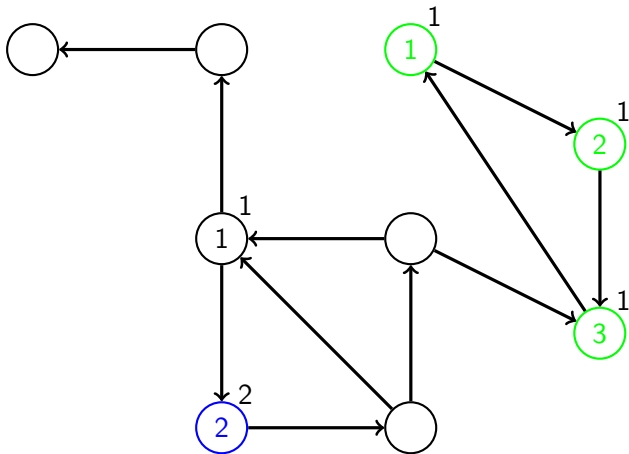


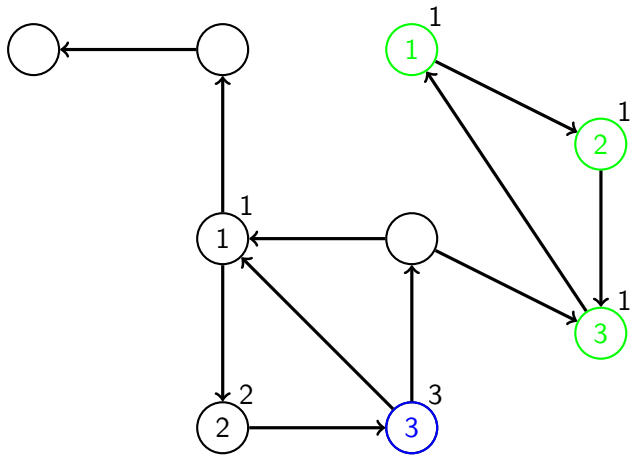


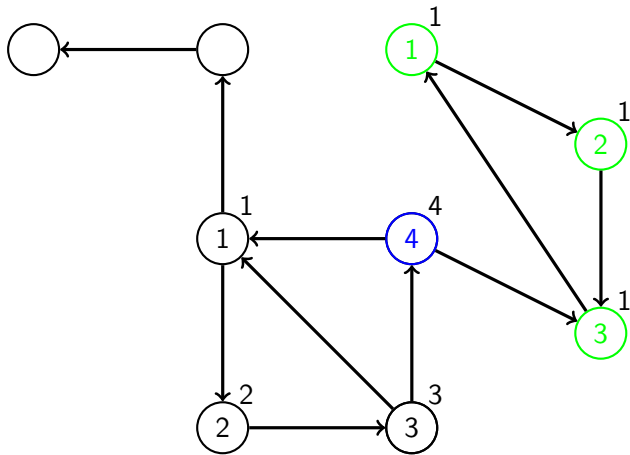


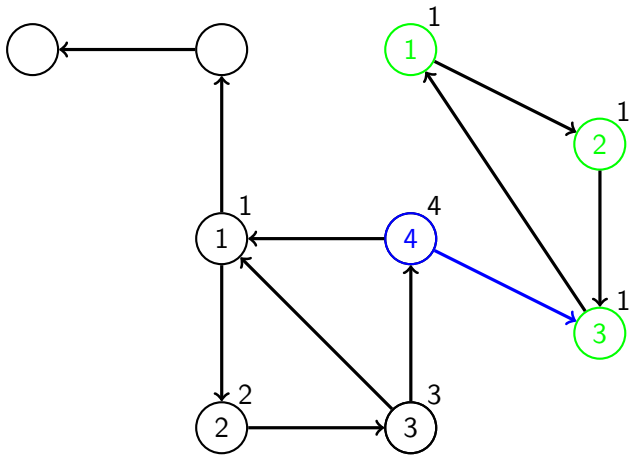


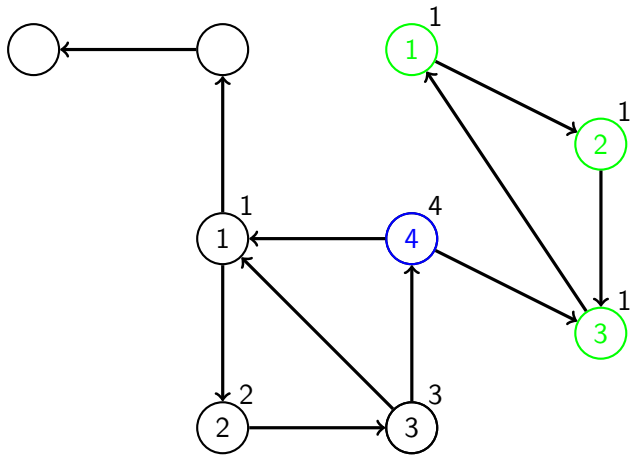


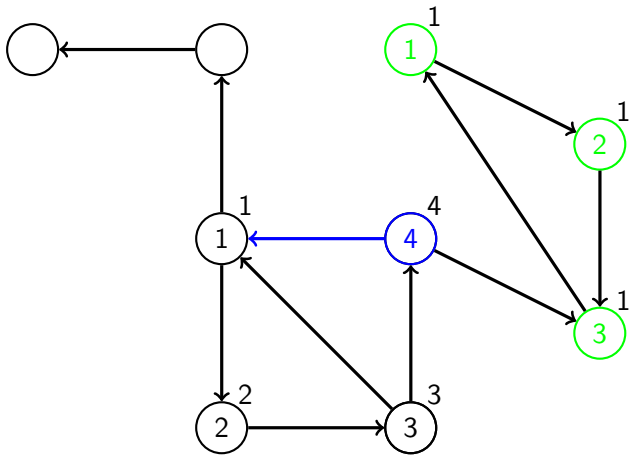


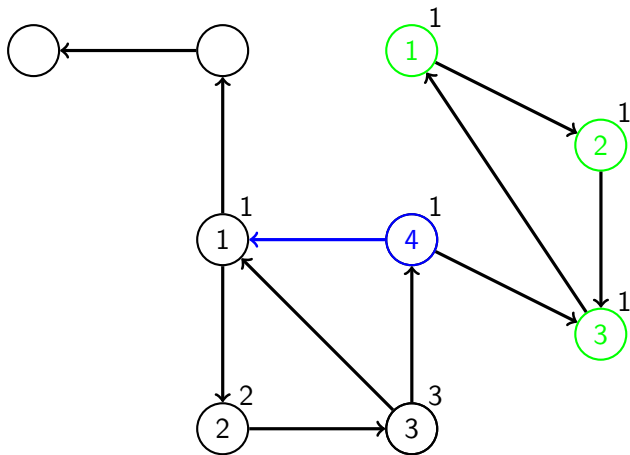


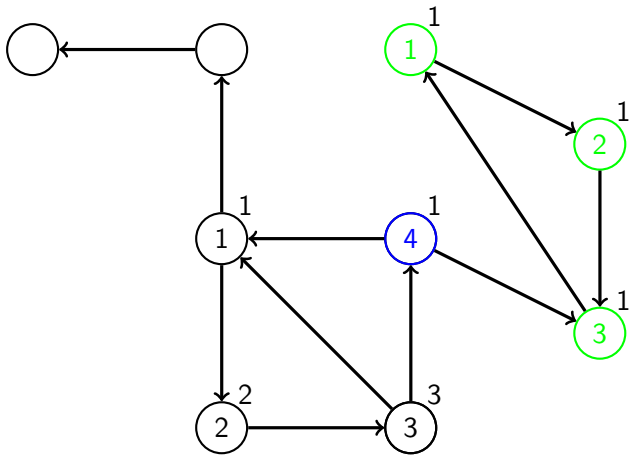


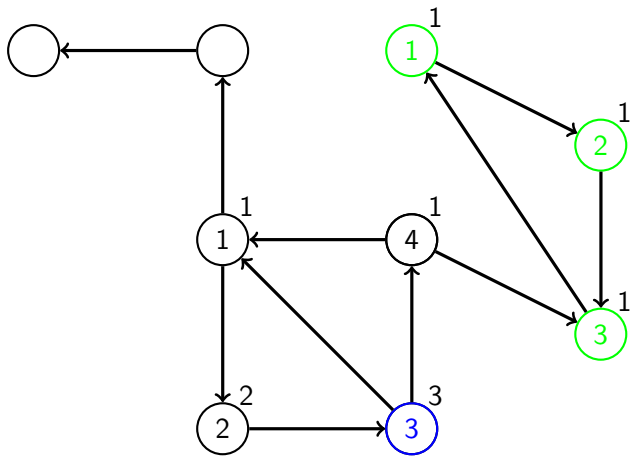


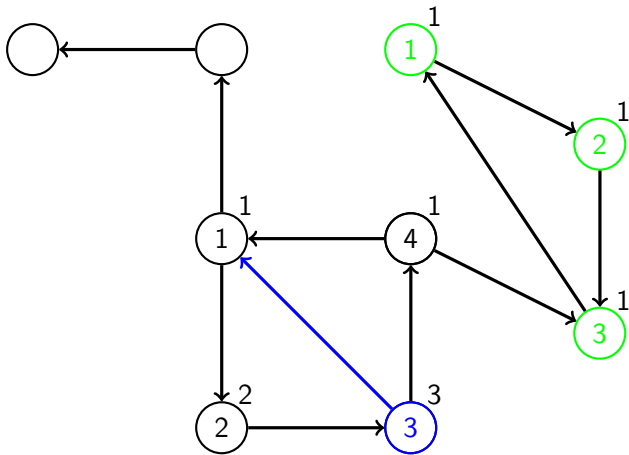


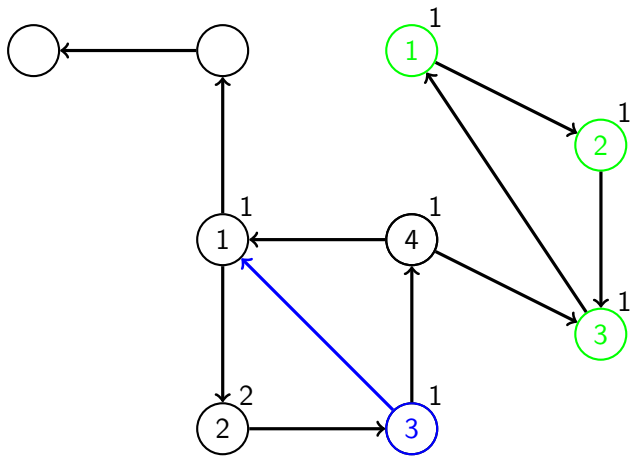


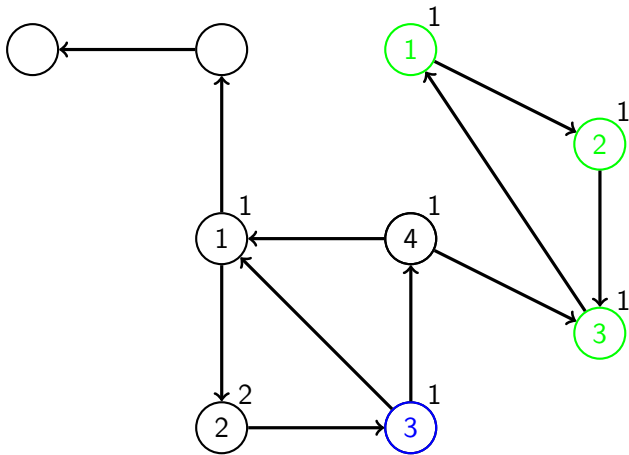


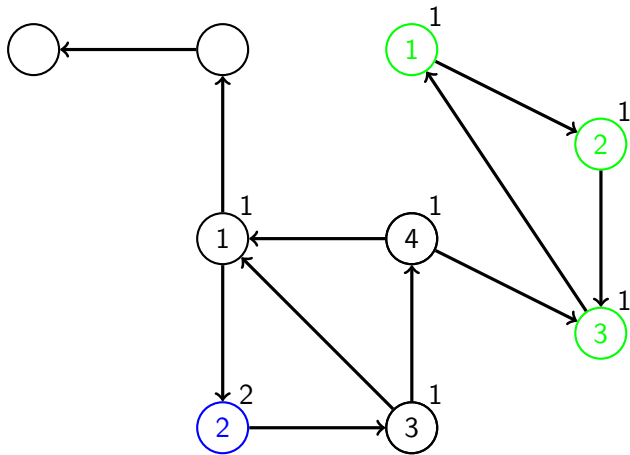


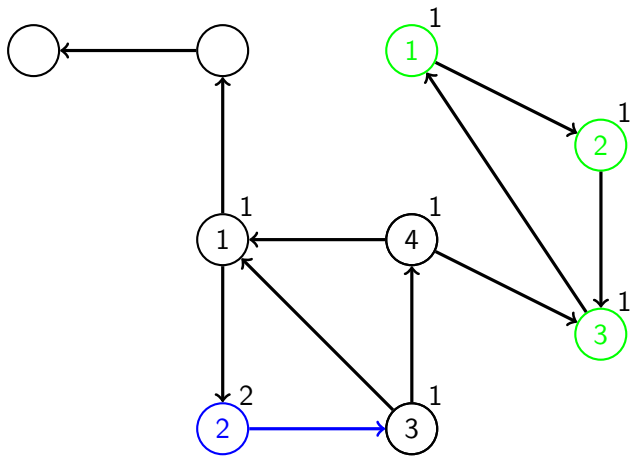


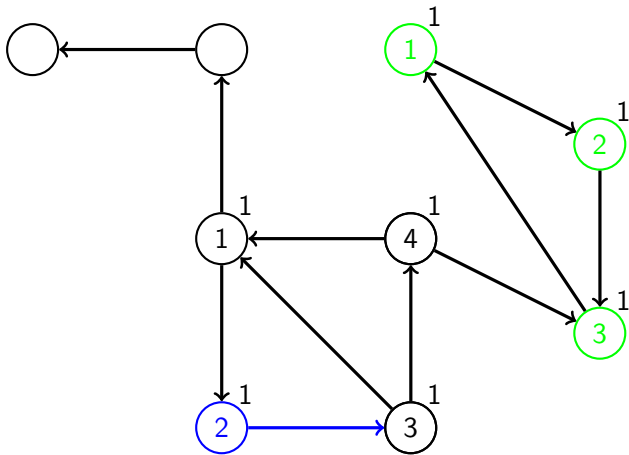


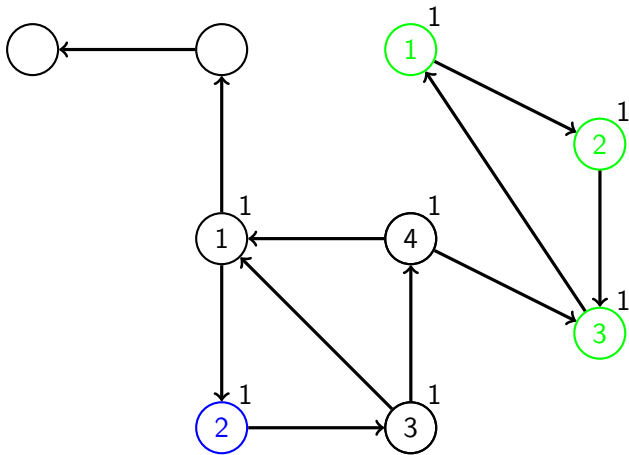


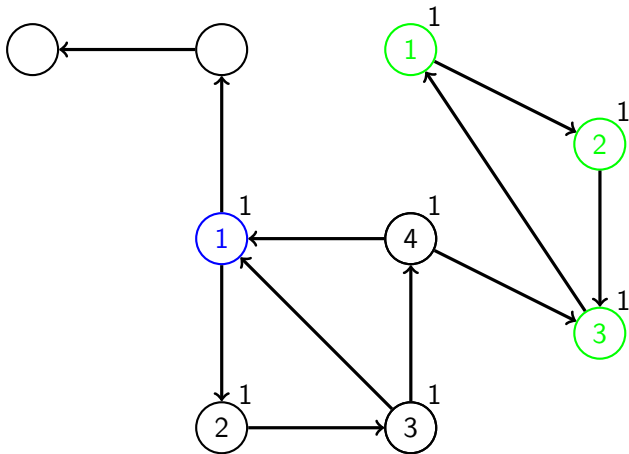


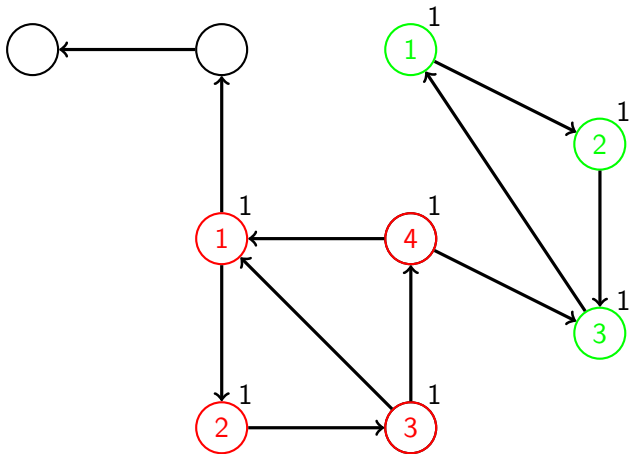


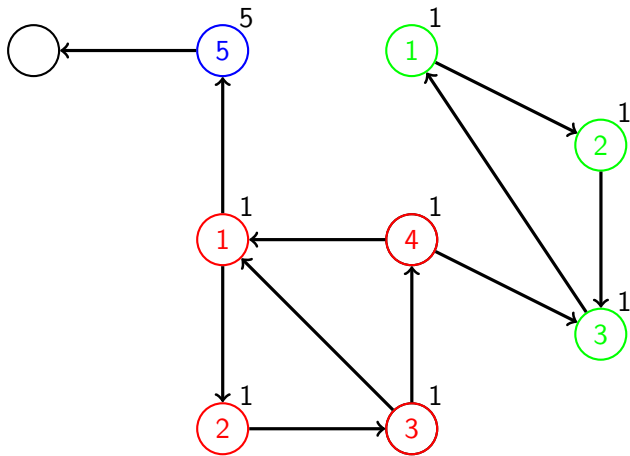


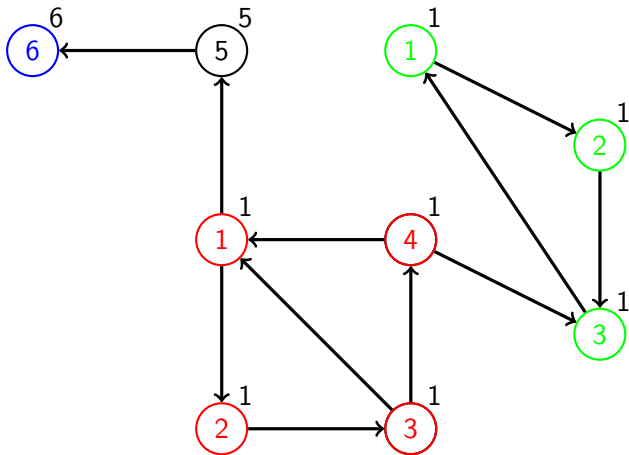


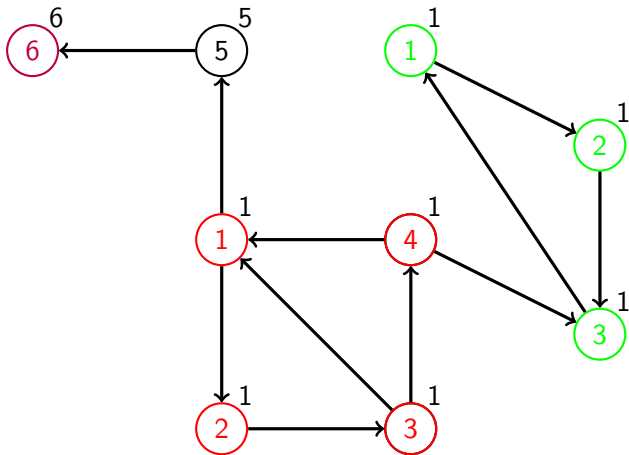


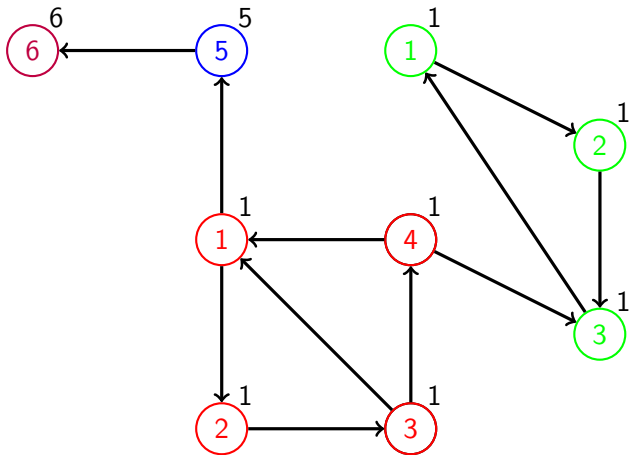


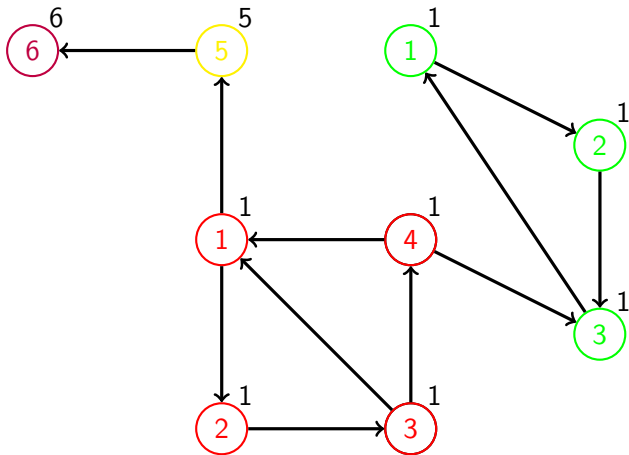












```

9  int dfs(vvi &g, int u, int d, int *s, int *a, int *l)
10 {
11     l[u] = d, s[++s[0]] = u;
12     int i, x, z = d;
13     for (i = 0; i < g[u].size(); i++)
14     {
15         x = g[u][i];
16         if (l[x] == -1) d = dfs(g, x, d + 1, s, a, l);
17         if (a[x] == -1) l[u] = min(l[u], l[x]);
18     }
19     if (l[u] == z) while (a[u] == -1) a[s[s[0]--]] = u;
20     return d;
21 }
22
23 void scc(vvi &g, int *a)
24 {
25     int i, n = g.size(), s[n + 1], l[n];
26     for (i = 0, s[0] = 0; i < n; i++) l[i] = a[i] = -1;
27     for (i = 0; i < n; i++) if (l[i] == -1) dfs(g, i, 0, s, a, l);
28 }

```

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

