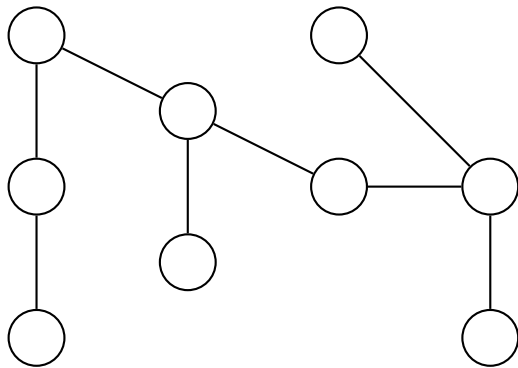


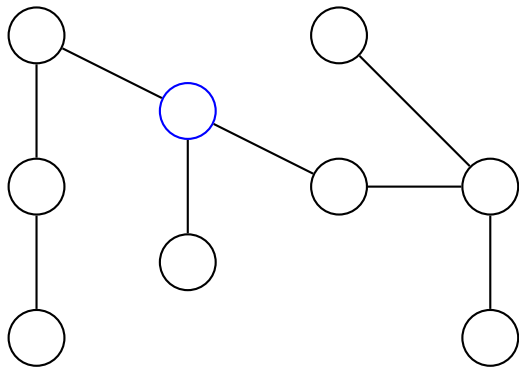
Hæsti sameiginlegi forfaðir

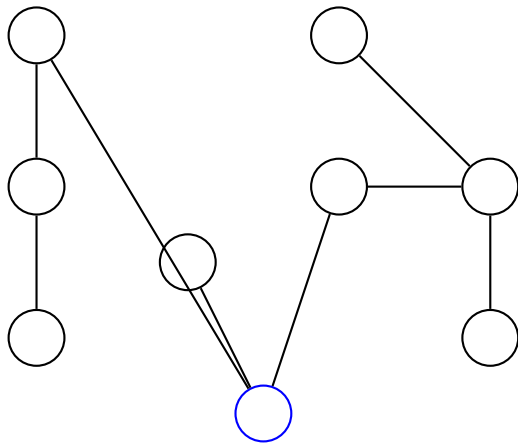
Bergur Snorrason

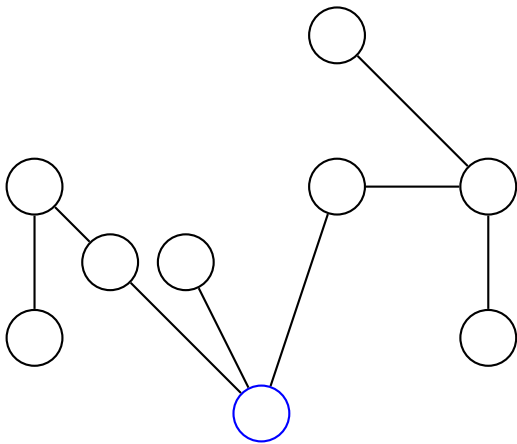
26. mars 2022

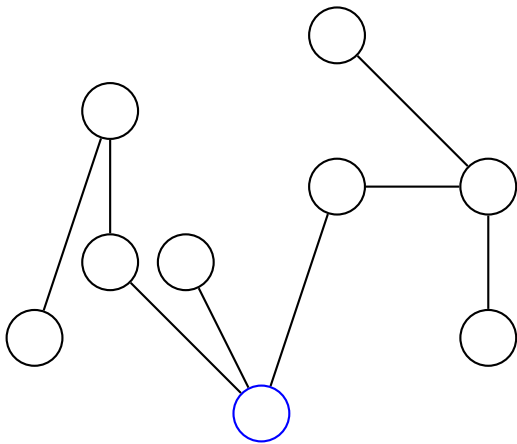
- ▶ Gerum ráð fyrir að við séum með tré.
- ▶ Látum einn hnút tákna rót trésins.
- ▶ Við getum þá talað um $hæð$ hnúts í trénu.
- ▶ Hæðin er þá fjarlægðin frá hnútnum í rótina.
- ▶ Þar sem við erum í tréi er aðeins einn einfaldur vegur í rótina.
- ▶ Ein leið til að finna hæð allra hnúta er með einni dýptarleit.
- ▶ Látum hæð hnútsins u vera $h(u)$.
- ▶ Við segjum líka að hæð trés sé H ef hnúturinn með hæstu hæð er H .

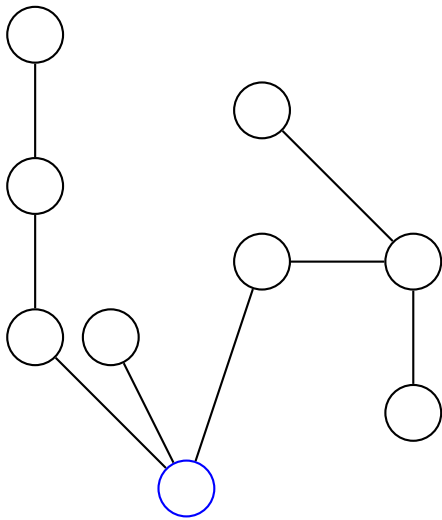


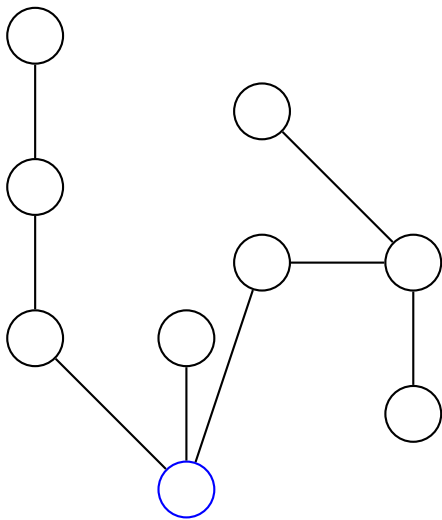


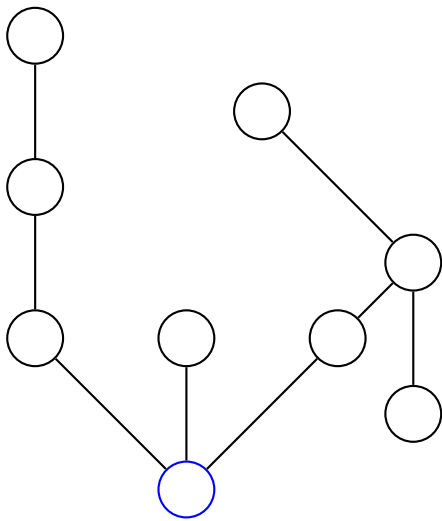


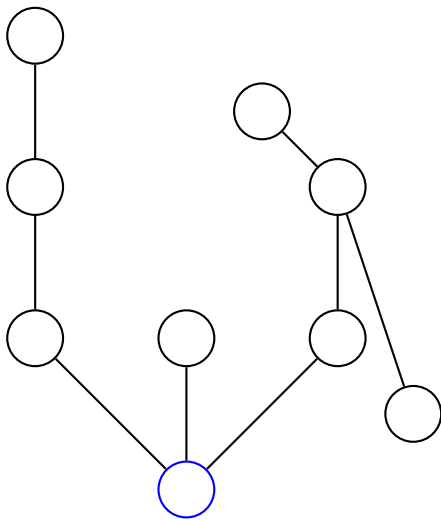


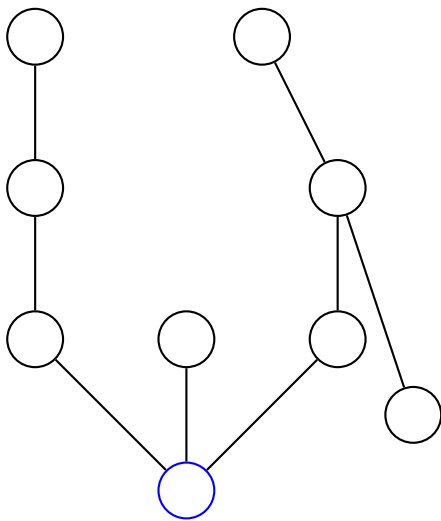


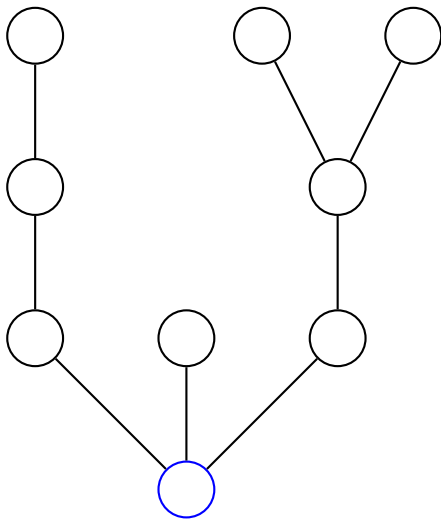


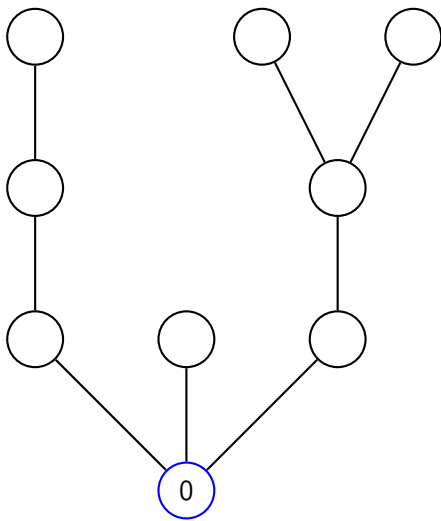


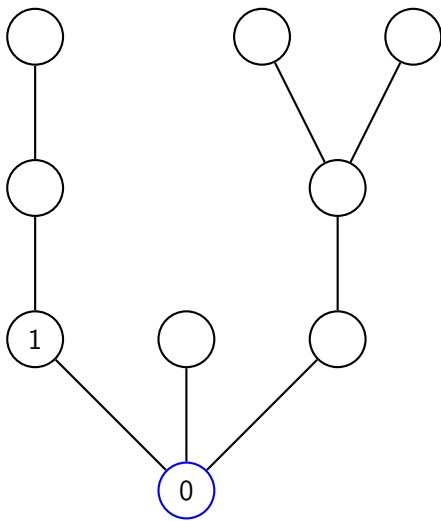


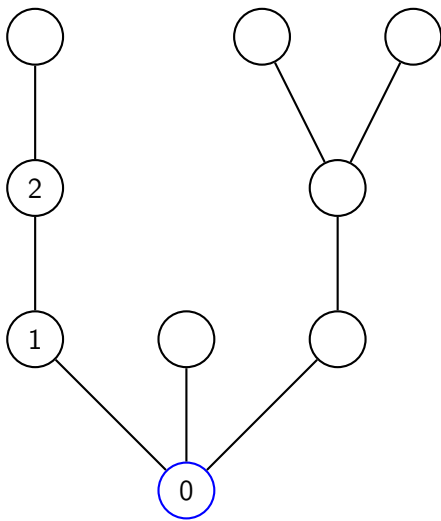


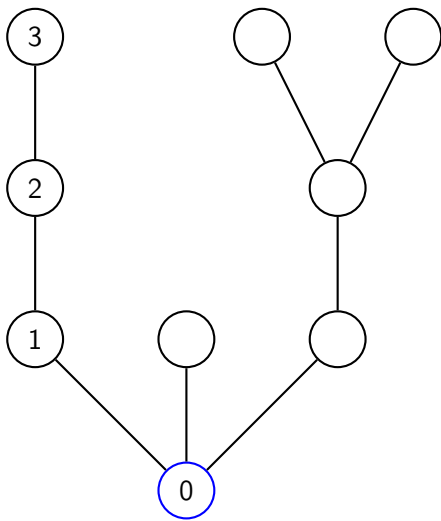


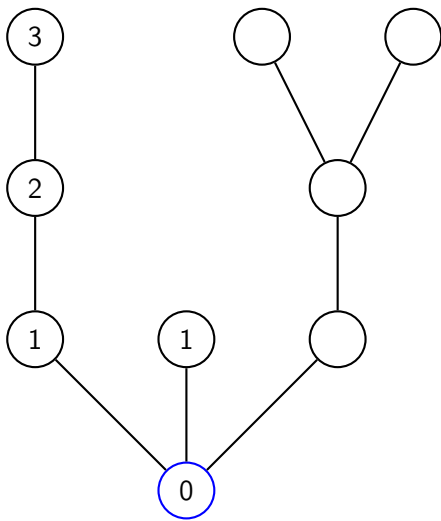


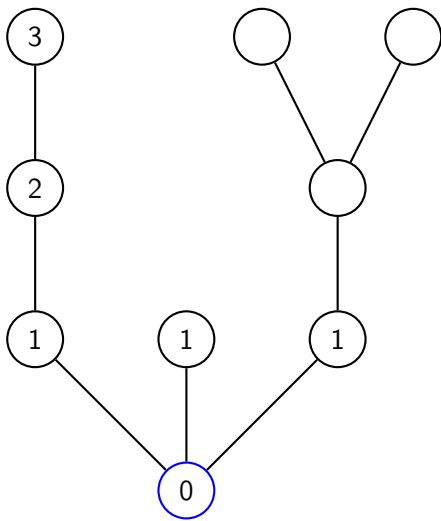


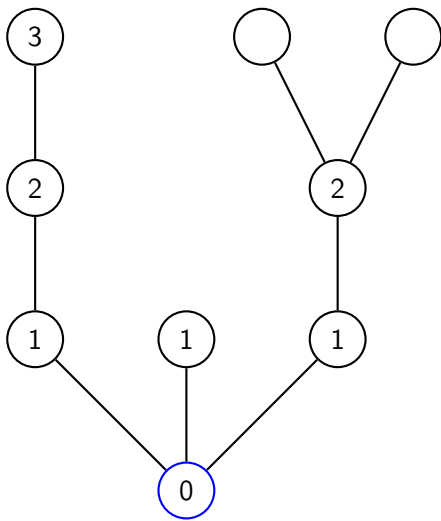


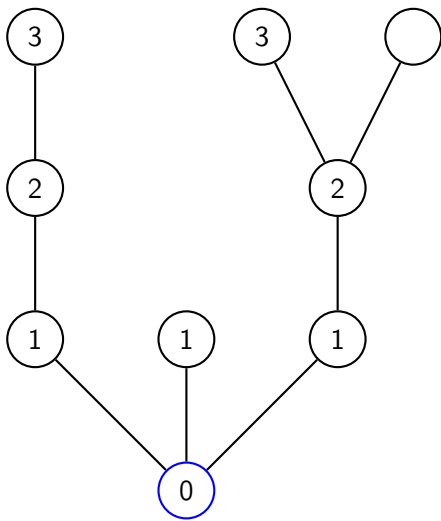


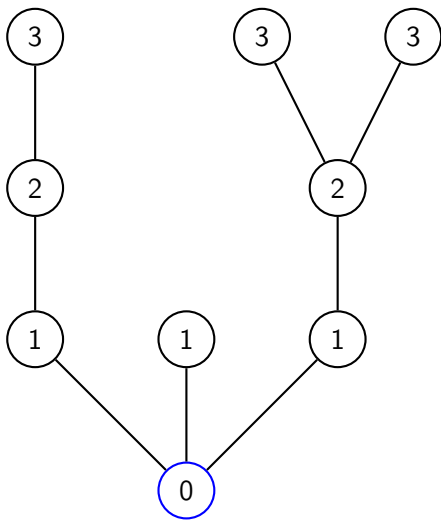




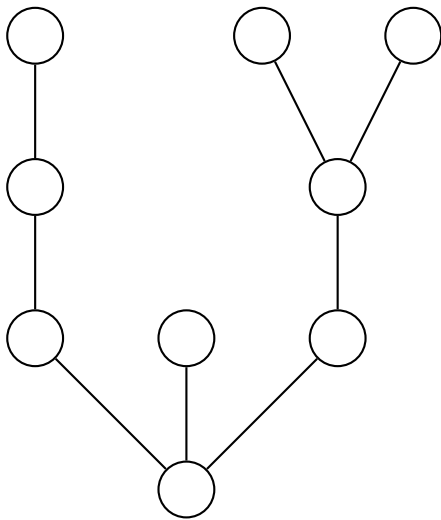


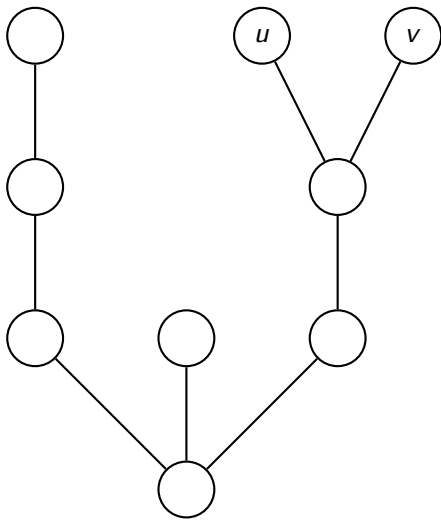


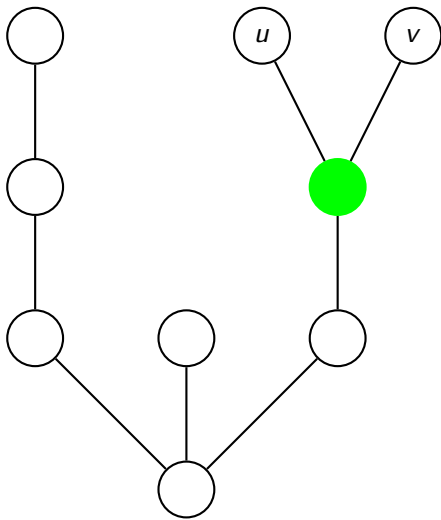


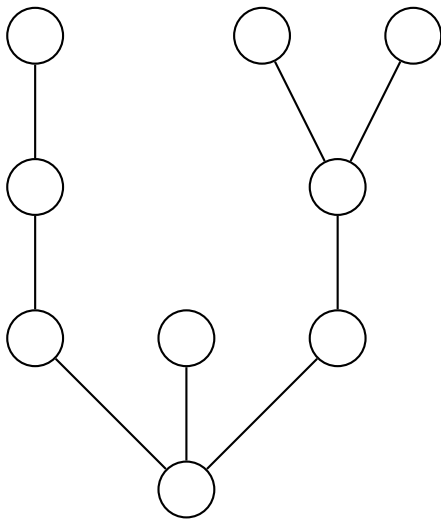


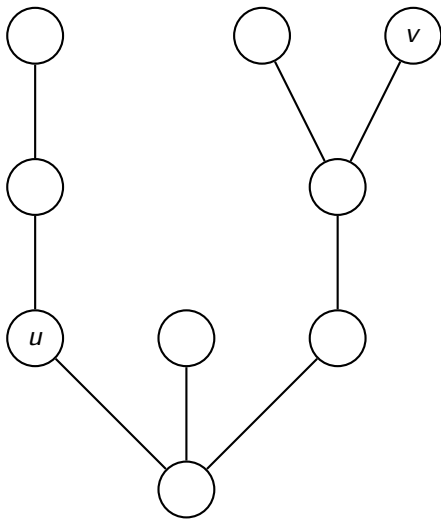
- ▶ Við köllum þann nágranna hnúts sem hefur einum lægri hæð *foreldri* hnútsins.
- ▶ Við getum þá fundið veginn að rótinni með því að ferðast eftir foreldrum.
- ▶ Þeir hnútar sem eru á veginum niður að rótinni kallast *forfeður* hnúts.
- ▶ Það er kannski skrítið, en allir hnútar er forfeður sínir.
- ▶ Oft nýtist okkur að vitað hvaða sameiginlegi forfaðir tveggja hnúta er hæstur í trénu.

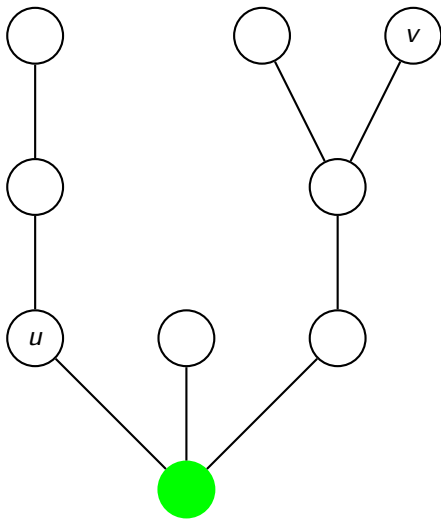


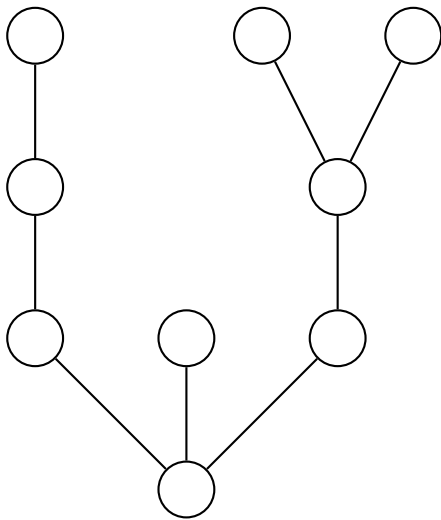


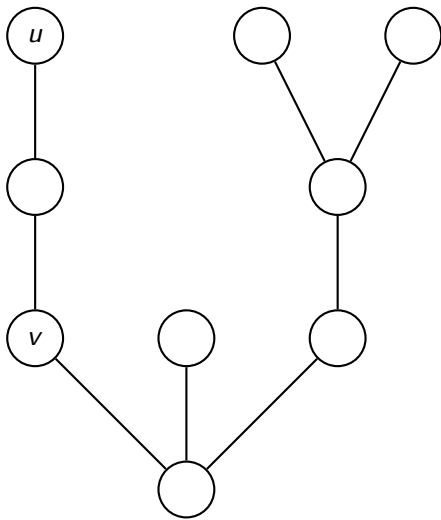


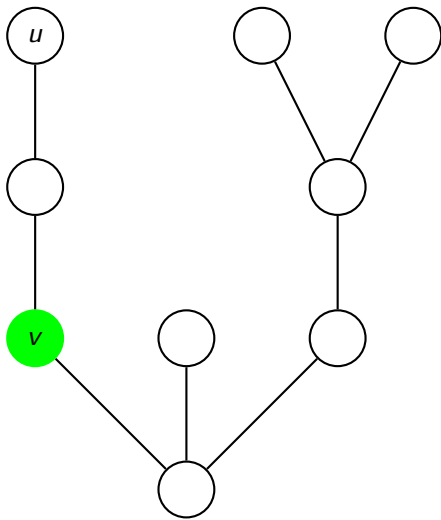


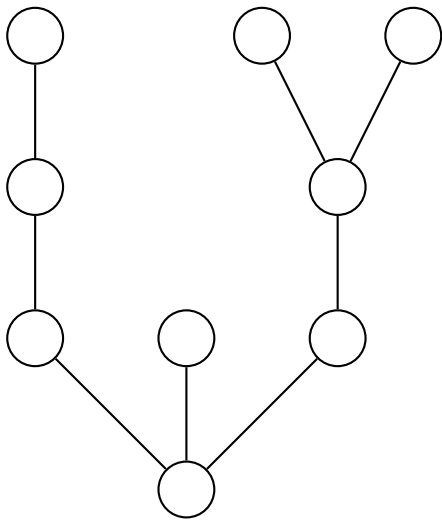




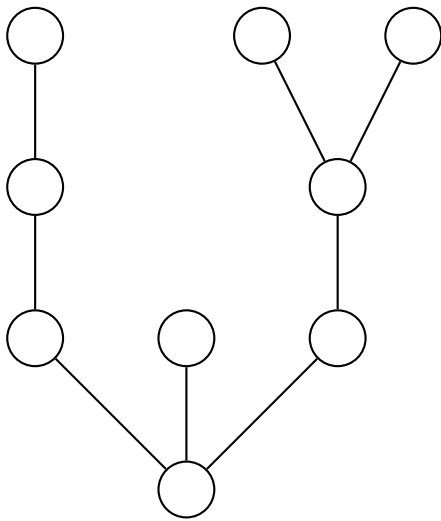


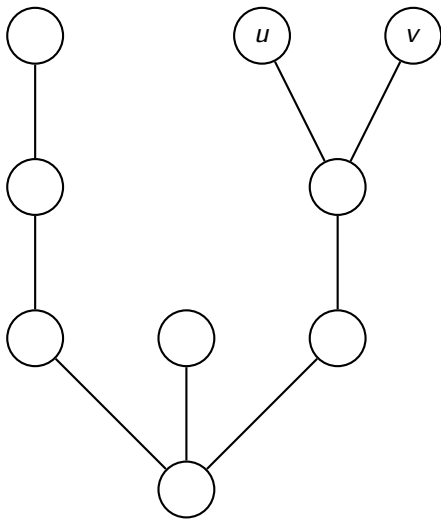


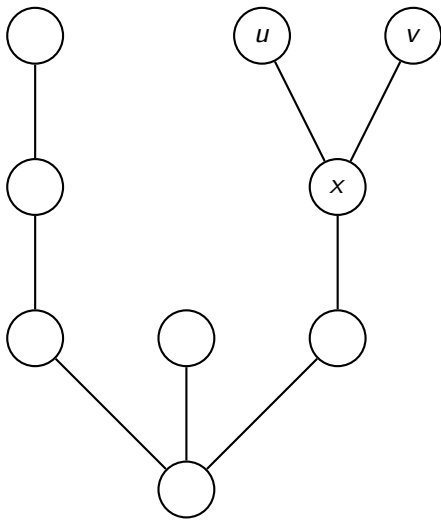


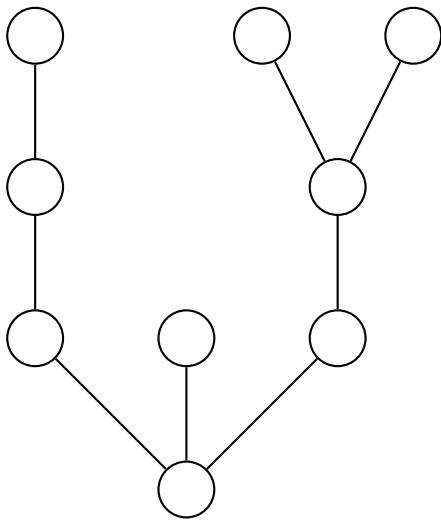


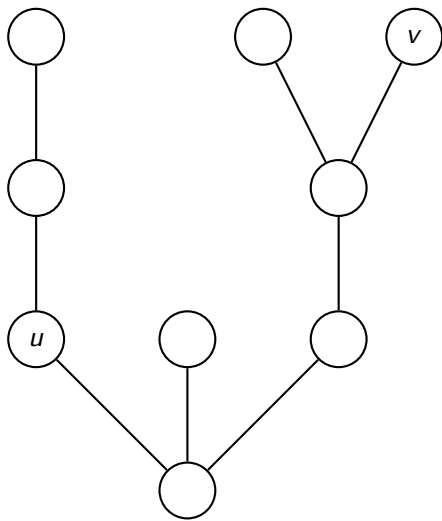
- ▶ Hvernig finnum þennan forföður?
- ▶ Látum u og v tákna hnútana og x hæsta sameiginlega forföður þeirra.
- ▶ Við getum gert ráð fyrir að u sé ofar í trénu, það er að segja $h(u) \geq h(v)$.
- ▶ Við vitum að allir forfeður u sem hafa hæð stærri en $d(v)$ eru ekki x .
- ▶ Svo við getum ferðast niður tréð frá u þar til við erum komin í sömu hæð og v .
- ▶ Við getum svo ferðast eftir foreldrum beggja á sama tíma þangað til við lendum í sama hnútnum.
- ▶ Sá hnútur er x .
- ▶ Sjáum hvernig þessi aðferð leysir sýnidæmið á undan.

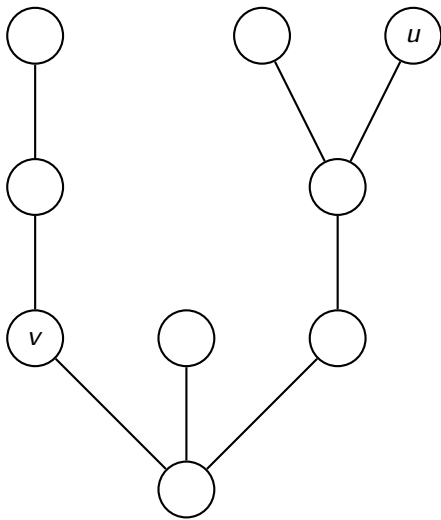


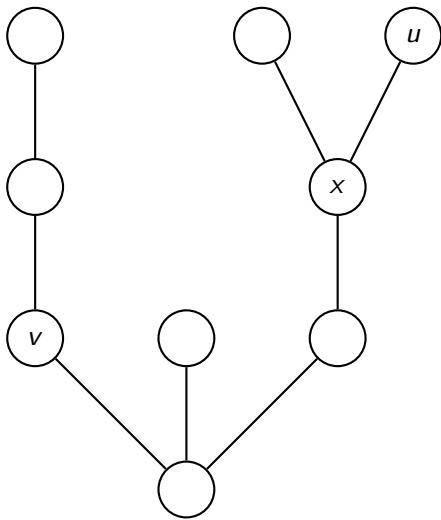


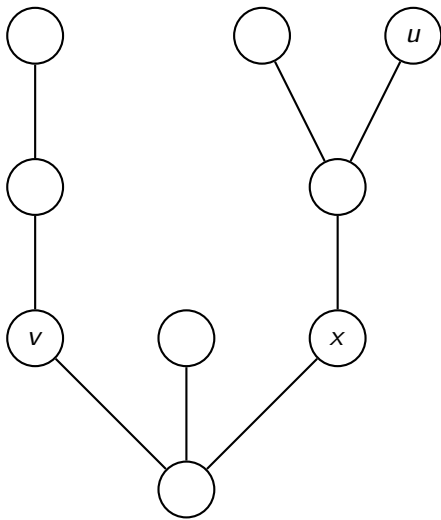


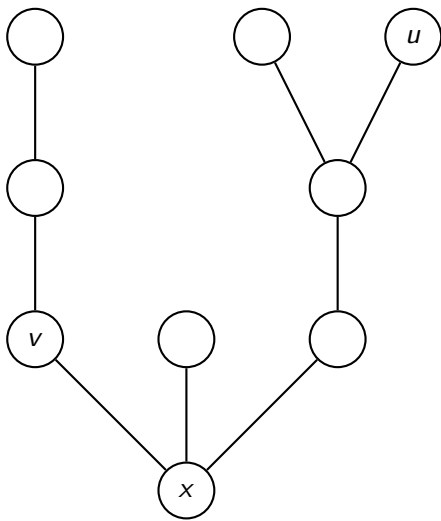


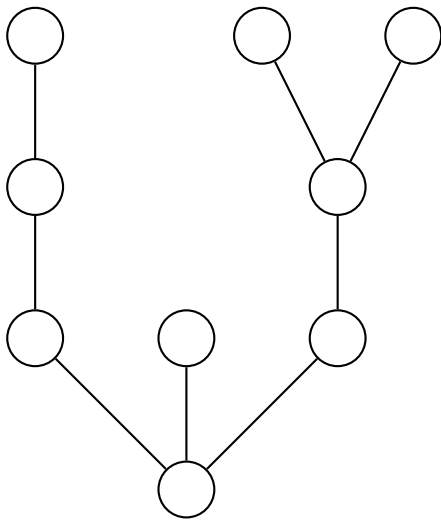


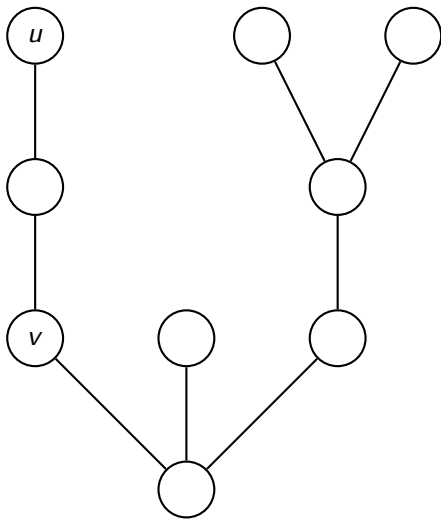


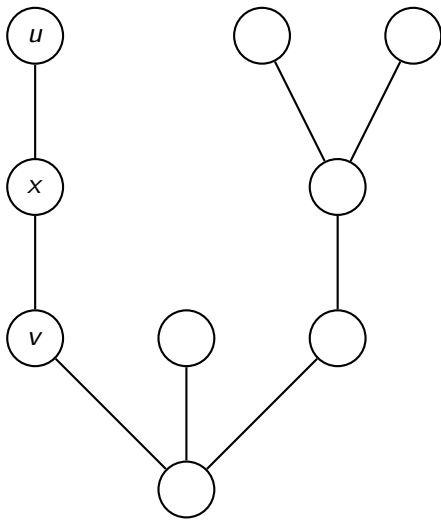


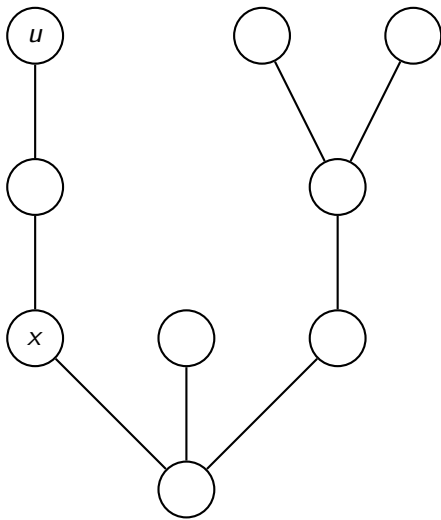


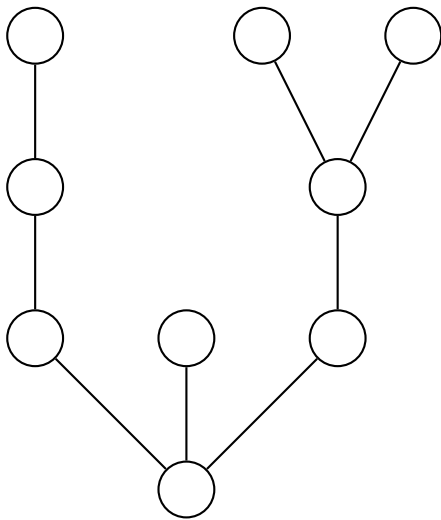


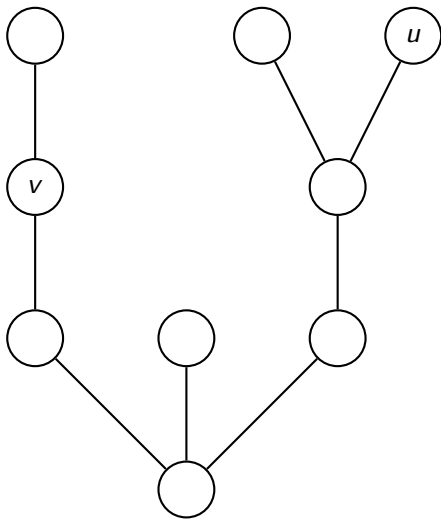


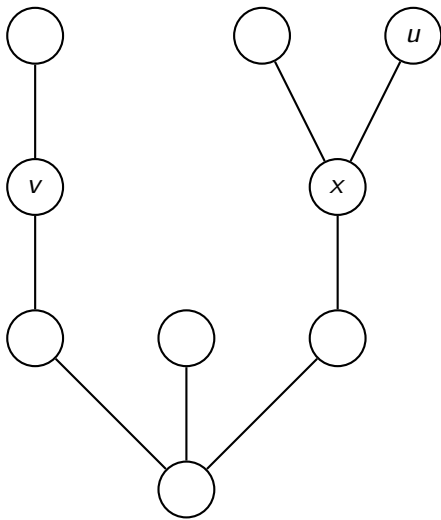


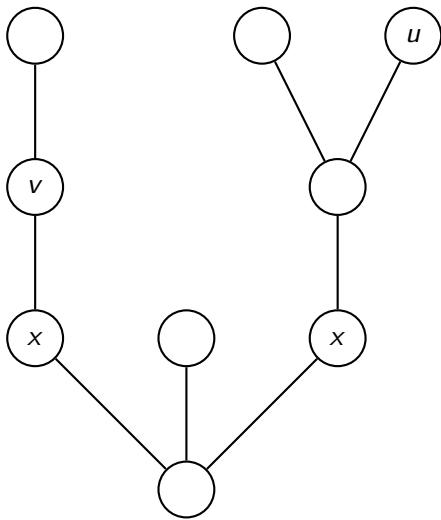


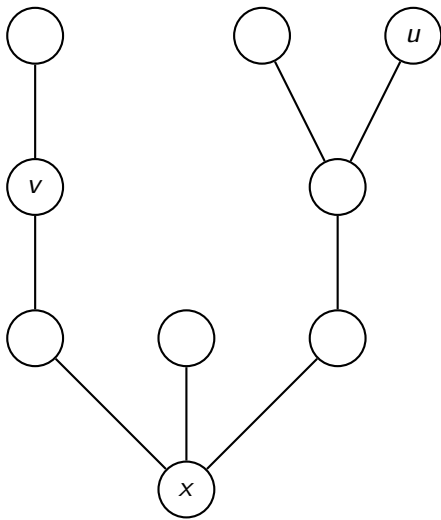












```

7 int p[MAXN], d[MAXN];
8 void lca_dfs(vvi& g, int x, int q, int w)
9 { // HjáTparfall.
10     int i;
11     d[x] = w, p[x] = q;
12     for (i = 0; i < g[x].size(); i++)
13         if (g[x][i] != q) lca_dfs(g, g[x][i], x, w + 1);
14 }
15
16 void lca_init(vvi& g, int x)
17 { // Upphafstillir fyrir netið g með rót x.
18     int i, n = g.size();
19     lca_dfs(g, 0, x, x);
20 }
21
22 int lca(int u, int v)
23 { // Skilar hæsta sameiginlega forfaðir u og v.
24     if (d[u] < d[v]) swap(u, v);
25     while (d[u] != d[v]) u = p[u];
26     while (u != v) u = p[u], v = p[v];
27     return u;
28 }

```

- ▶ Gerum ráð fyrir að hæð trésins sé H .
- ▶ Þá er tímaflækjan á þessari aðferð $\mathcal{O}(H)$.
- ▶ Í versta falli er hæð trés með n hnúta $n - 1$.
- ▶ Svo tímaflækjan er í versta falli $\mathcal{O}(n)$.
- ▶ Við getum þó bætt þetta með því að taka stærri stökk.

- ▶ Aðferðin skiptist í tvö skref:
 - ▶ Jöfnum hæðina á hnútunum.
 - ▶ Löbbum saman niður þangað til við finnum svarið.
- ▶ Fyrri skrefinu má lýsa nánar.
- ▶ Látum hnútana okkar vera u og v , þannig að $d(u) \geq d(v)$.
- ▶ Við viljum því ferðast niður nákvæmlega $d(v) - d(u)$ sinnum.
- ▶ Ein leið til að gera þetta hratt er að geyma ekki bara foreldri hvers hnúts, heldur alla hnúta sem eru 2^k fyrir neðan hnútin i trénu.
- ▶ Við þurfum því að geyma $\mathcal{O}(\log n)$ stökk fyrir hvern hnút.
- ▶ Táknum með $p(u, k)$ þann hnút sem þú endar í ef þú ferðast 2^k sinnum niður tréð frá u gegnum foreldrin.
- ▶ Til dæmis er $p(u, 1)$ foreldri u .
- ▶ Til þæginda segjum við að foreldri rótarinnar sé rótin sjálf.
- ▶ Við finnum þessi gildi með $p(u, i) = p(p(u, i-1), i-1)$.

- ▶ Við tökum því eins löng stökk og við getum án þess að $d(u) < d(v)$ þangað til $d(u) = d(v)$.
- ▶ Við getum þú núna gert ráð fyrir að $d(u) = d(v)$.
- ▶ Þá viljum við taka eins löng stökk og við getum þannig að $u \neq v$.
- ▶ Að því loknu munu u og v hafa sama foreldri.


```

9  int p[MAXN][MAXK], d[MAXN];
10 void lca_dfs(vvi& g, int x, int q, int w)
11 { // Hjálfarfall.
12     int i;
13     d[x] = w;
14     for (i = 0; i < MAXK; i++) p[x][i] = i == 0 ? q : p[p[x][i - 1]][i - 1];
15     for (i = 0; i < g[x].size(); i++) if (g[x][i] != q)
16         lca_dfs(g, g[x][i], x, w + 1);
17 }
18
19 void lca_init(vvi& g, int x)
20 { // Upphafstillir fyrir netið g með rót x.
21     int i, n = g.size();
22     for (i = 0; i < MAXK; i++) p[x][i] = x;
23     lca_dfs(g, 0, x, x);
24 }
25
26 int lca(int u, int v)
27 { // Skilar hæsta sameiginlega forfaðir u og v.
28     int i;
29     if (d[u] < d[v]) swap(u, v);
30     for (i = MAXK - 1; i >= 0; i--) if (d[p[u][i]] >= d[v]) u = p[u][i];
31     for (i = MAXK - 1; i >= 0; i--) if (p[u][i] != p[v][i])
32         u = p[u][i], v = p[v][i];
33     return u == v ? u : p[u][0];
34 }

```

- ▶ Í hverju skrefi þurfum við bara að taka $\mathcal{O}(\log n)$.
- ▶ Svo tímaflækjan er $\mathcal{O}(\log n)$.

