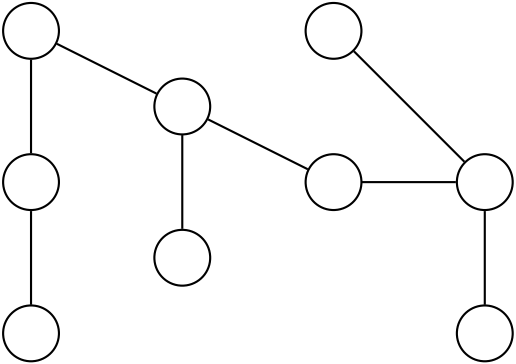
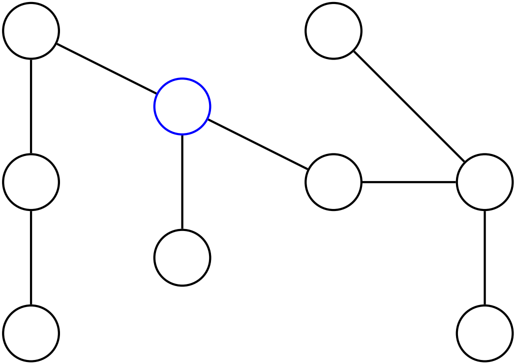
Hæsti sameiginlegi forfaðir

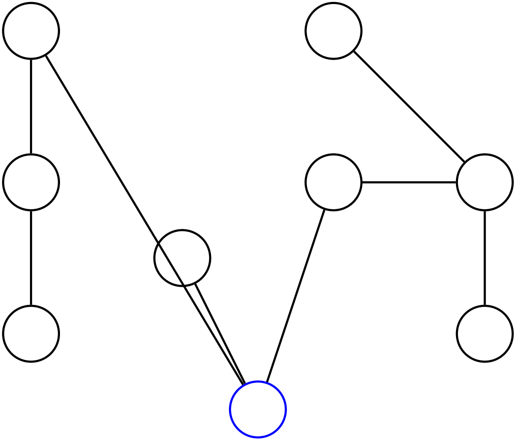
Bergur Snorrason

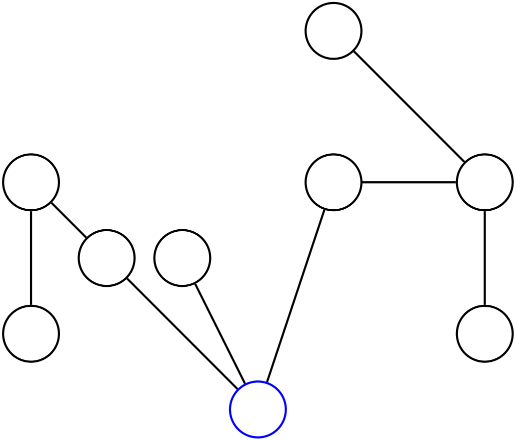
26. mars 2022

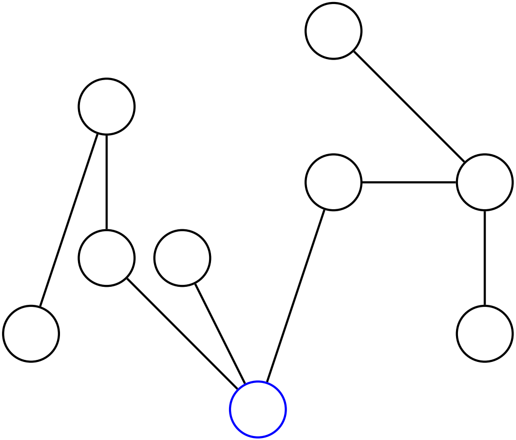
- Gerum ráð fyrir að við séum með tré.
- Látum einn hnút tákna rót trésins.
- Við getum þá talað um *hæð* hnúts í trénu.
- Hæðin er þá fjarlægðin frá hnútnum í rótina.
- Þar sem við erum í tréi er aðeins einn einfaldur vegur í rótina.
- Ein leið til að finna hæð allra hnúta er með einni dýptarleit.
-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

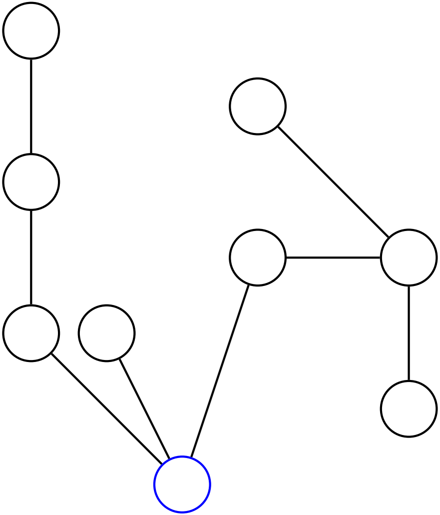


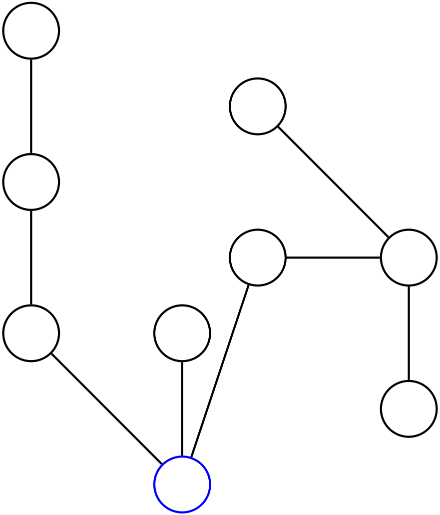


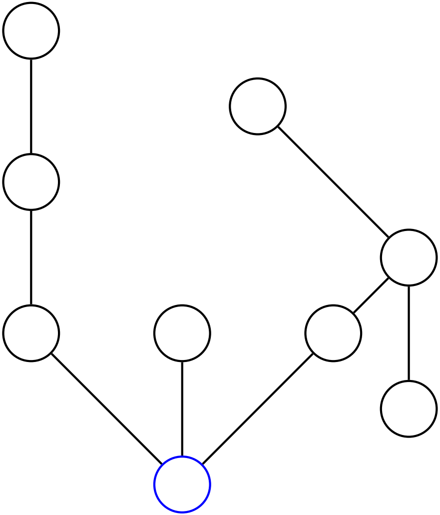


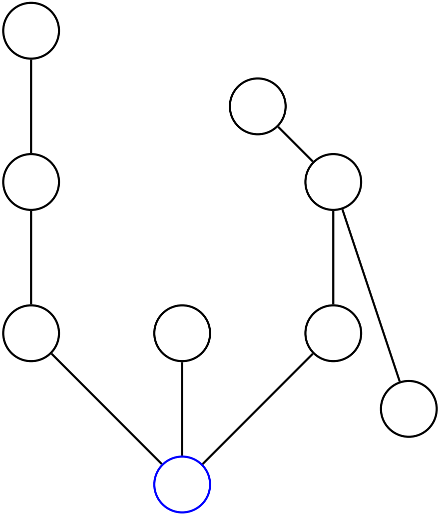


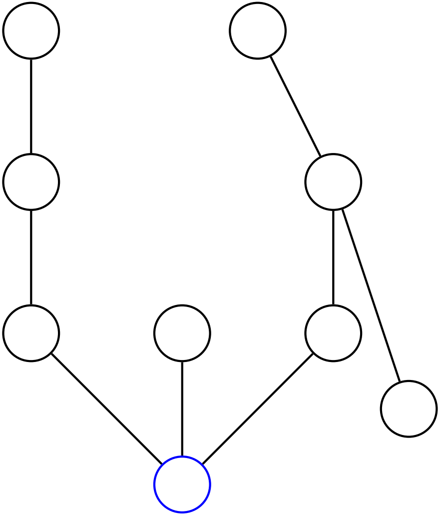


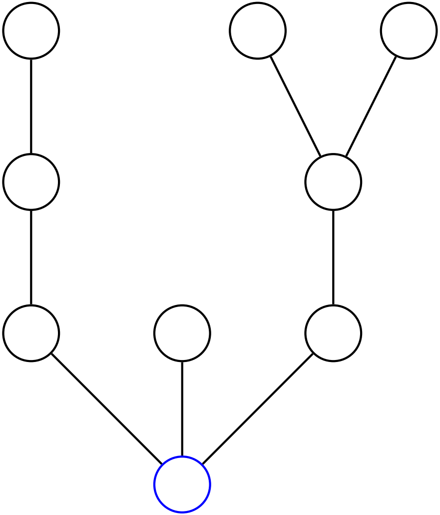


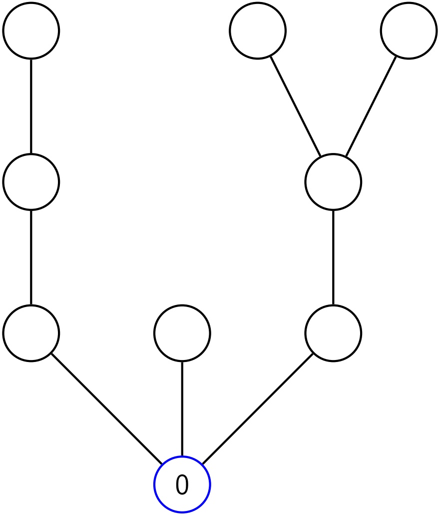


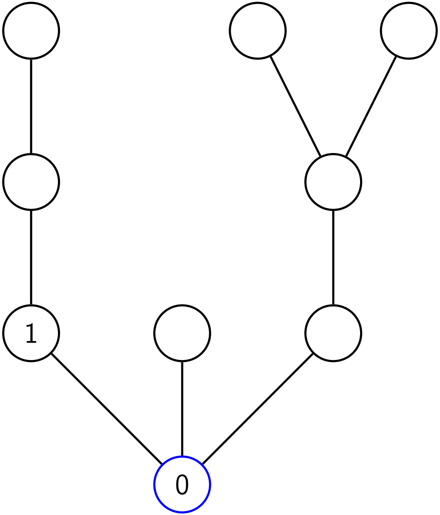


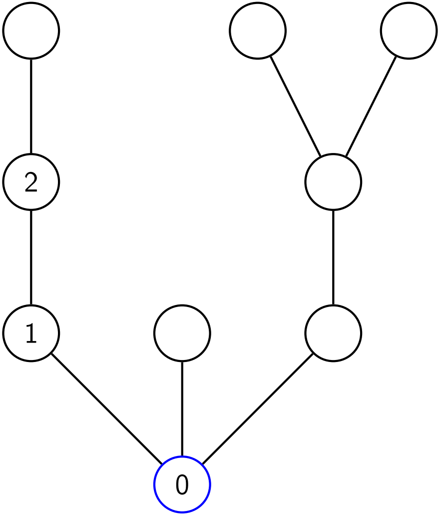


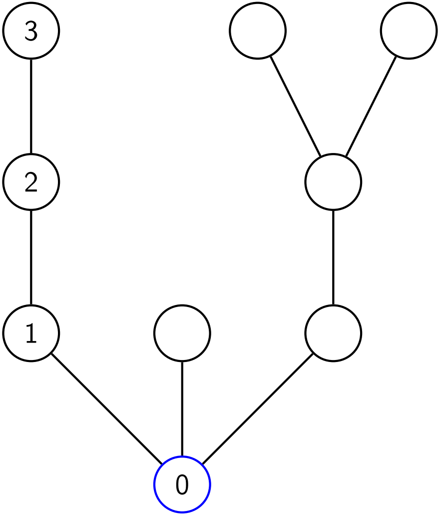


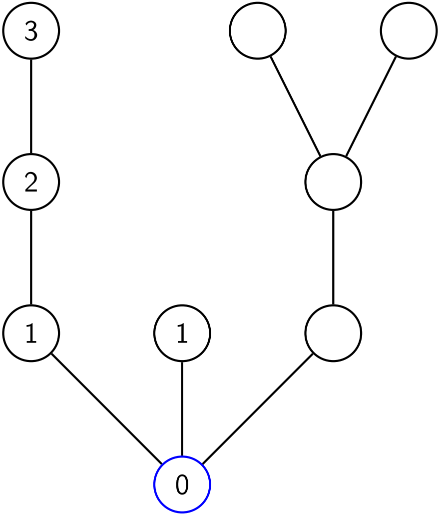


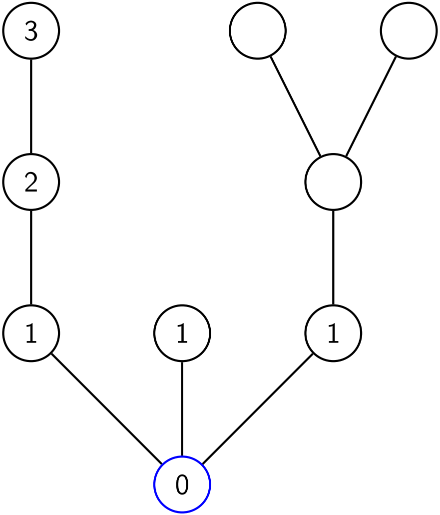

0

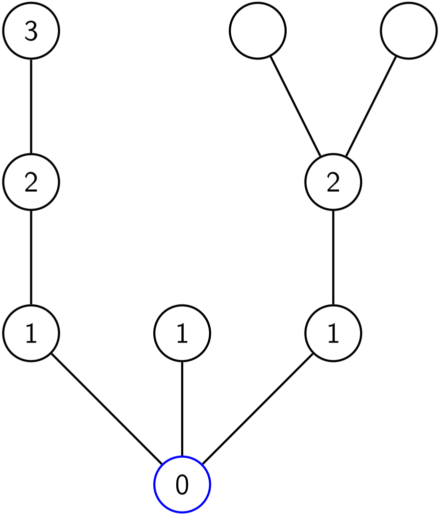

1
0

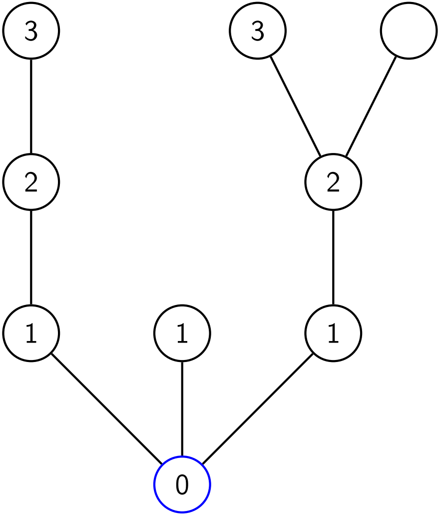

2
1
0

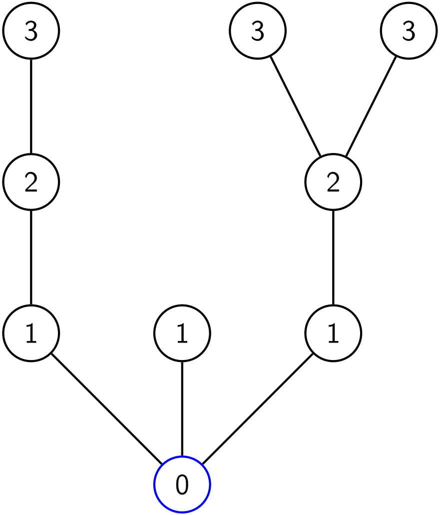

3
2
1
0


3
2
1
1
0

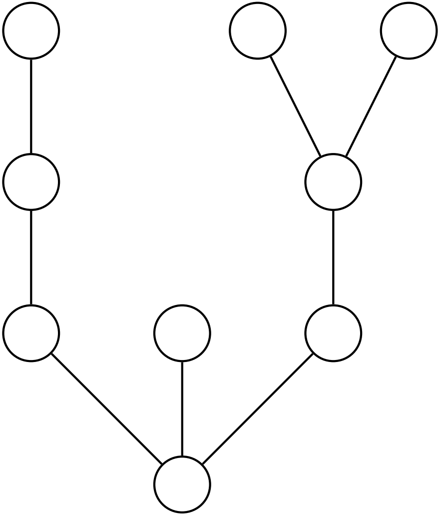

3
2
1
1
1
0

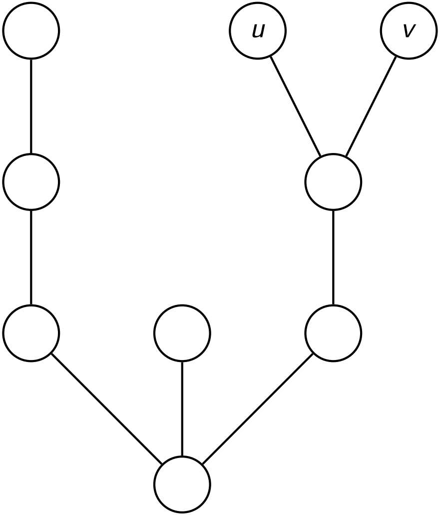

3
2
2
1
1
1
0

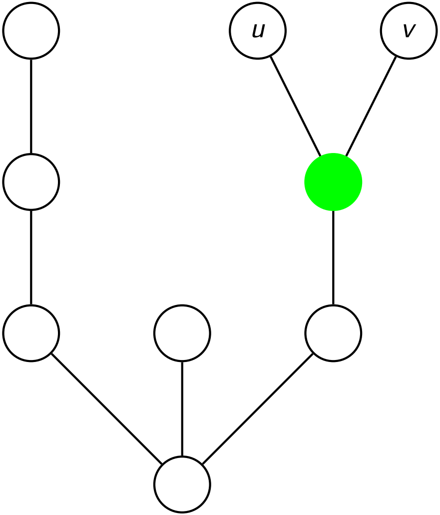

3
3
2
2
1
1
1
0


3
3
3
2
2
1
1
1
0

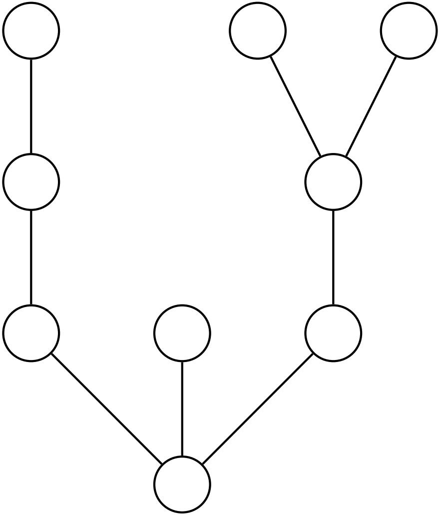
- Við köllum þann nágranna hnúts sem hefur einum lægri hæð *foreldri* hnútsins.
- Við getum þá fundið veginn að rótinni með því að ferðast eftir foreldrum.
- Þeir hnútar sem eru á veginum niður að rótinni kallast *forfeður* hnúts.
- Það er kannski skrýtið, en allir hnútar er forfeður sínir.
- Oft nýtist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

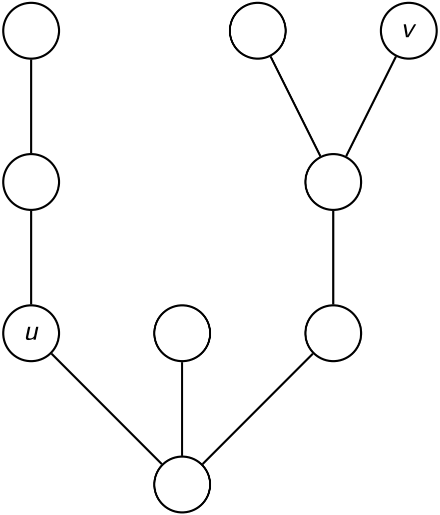


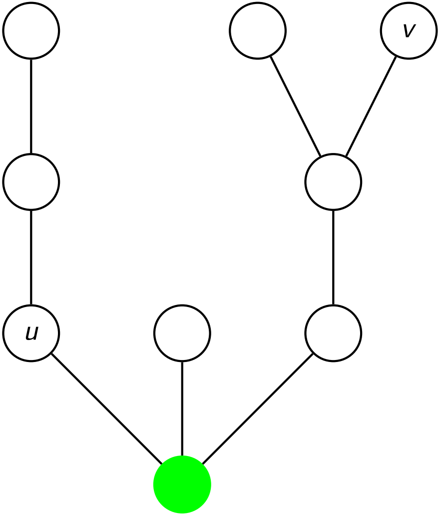

u
v

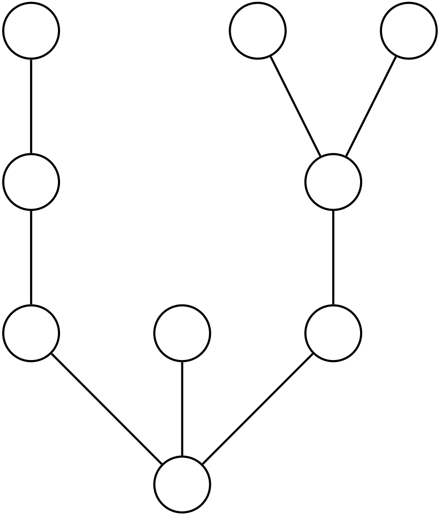


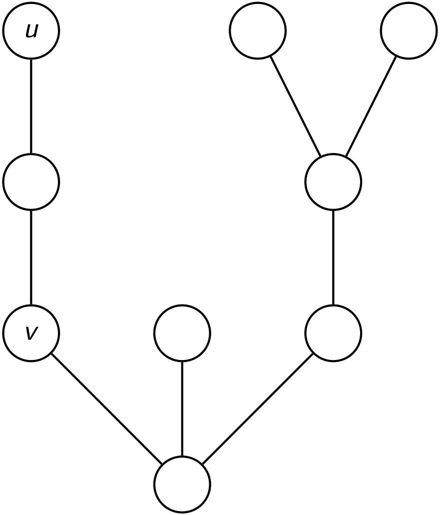
u
v

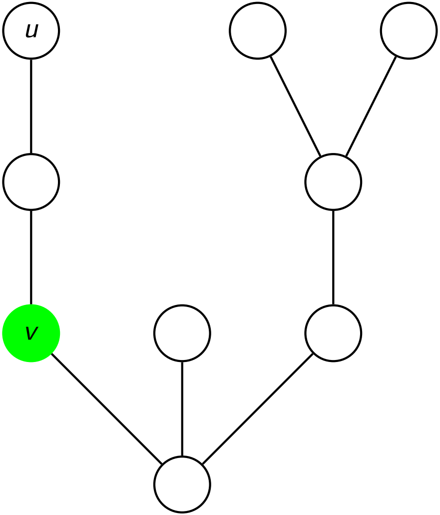


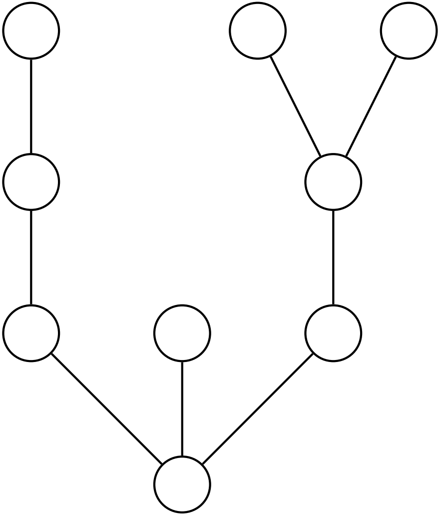

v
u


v
u

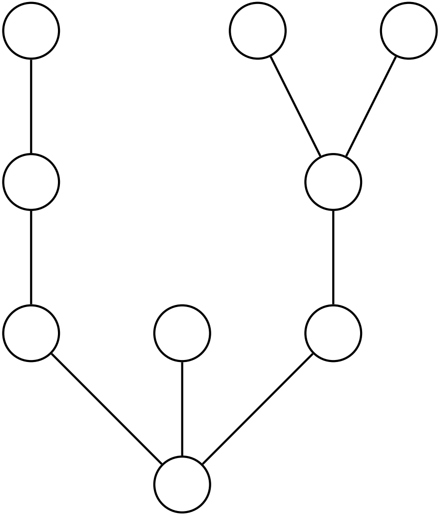


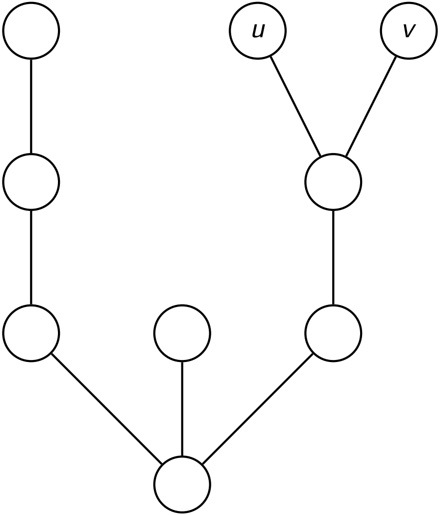

u
v

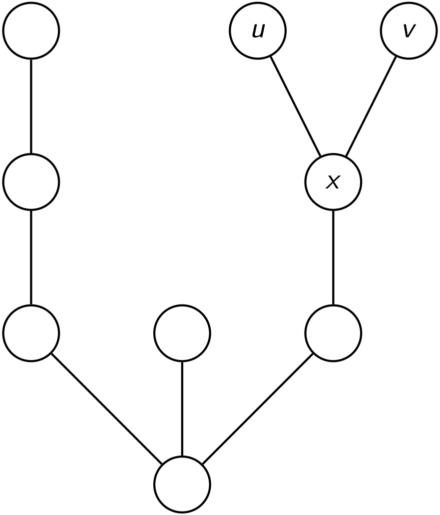

u
v

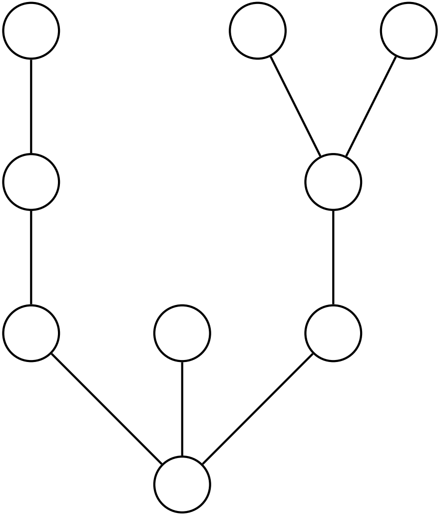


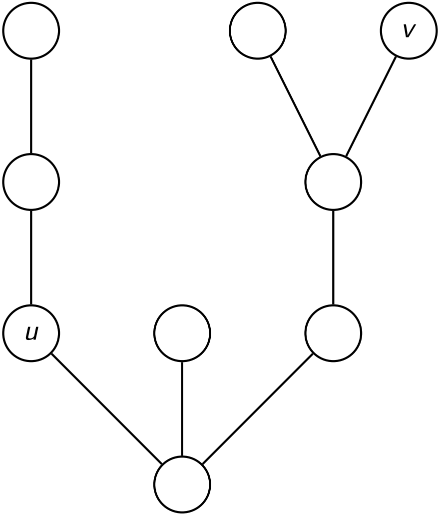
- Hvernig finnum þennan forföður?
- Látum u og v tákna hnútana og x hæsta sameiginlega forföður þeirra.
- Við getum gert ráð fyrir að u sé ofar í trénu, það er að segja $h(u) \geq h(v)$.
- Við vitum að allir forfeður u sem hafa hæð stærri en $d(v)$ eru ekki x.
- Svo við getum ferðast niður tréð frá u 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.

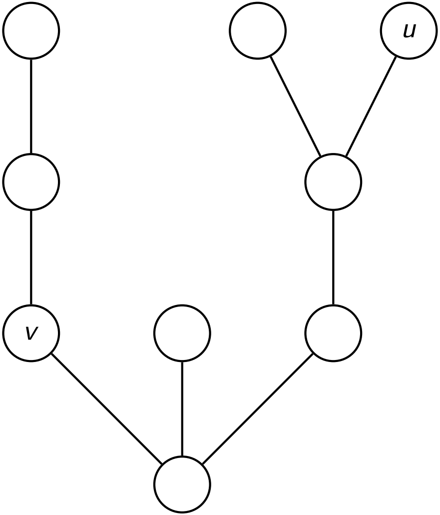


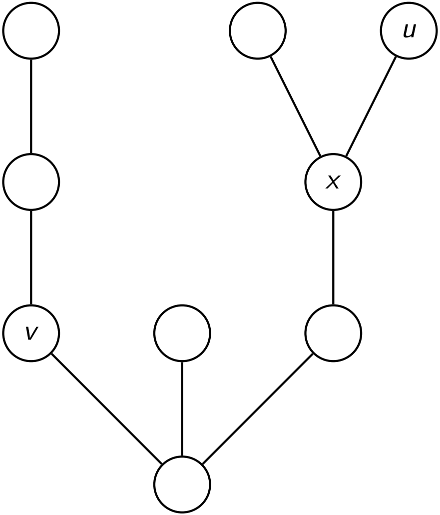

u
v

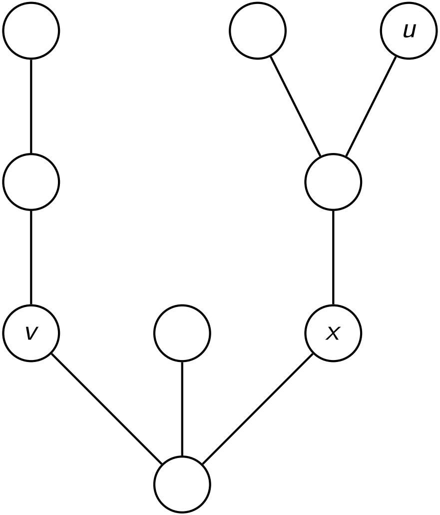

u
v
x



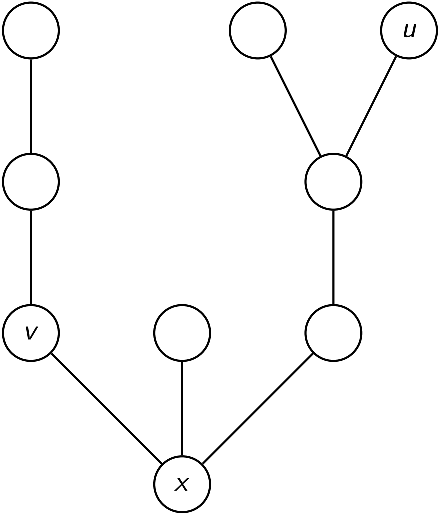

v
u

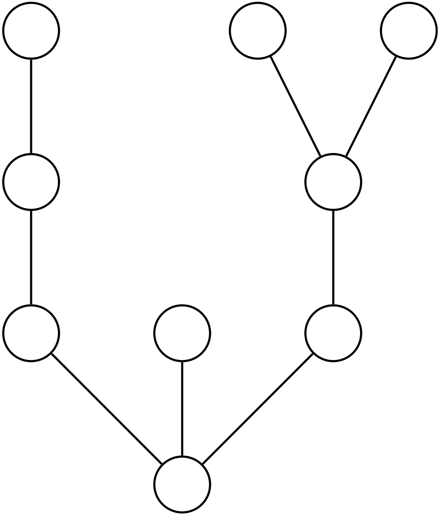

u
v

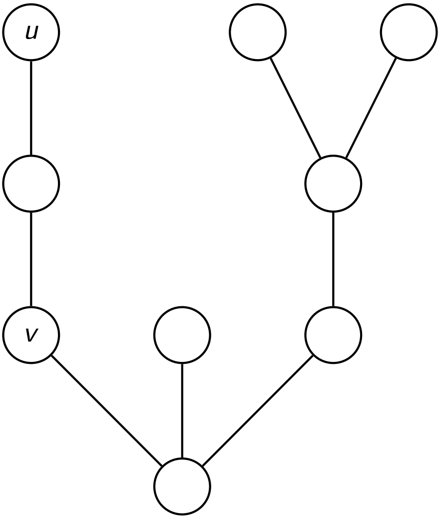

u
x
v

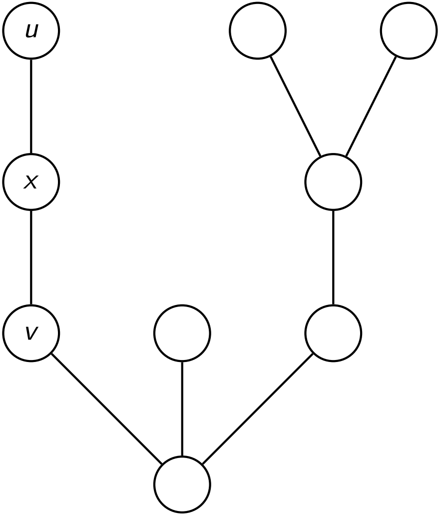


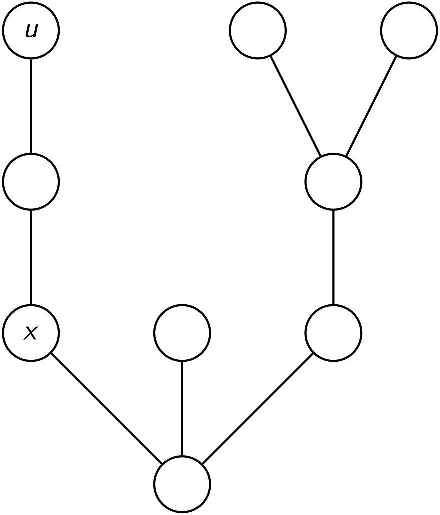
u
v
x


u
v
x

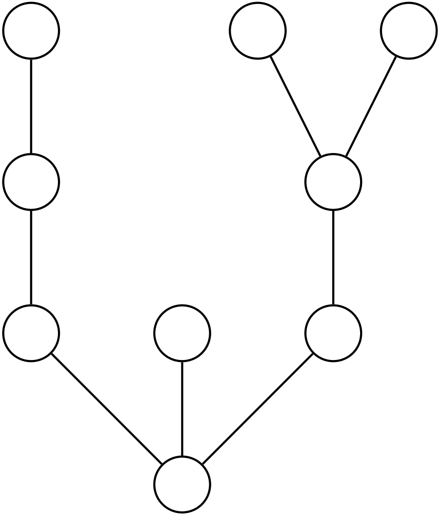


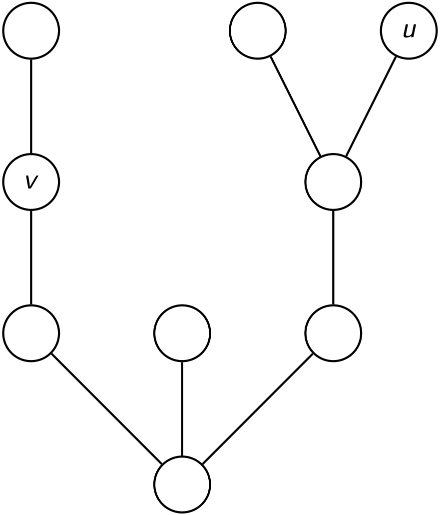

u
v

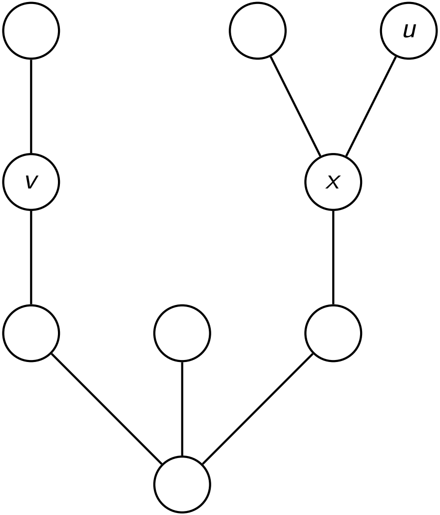

u
x
v

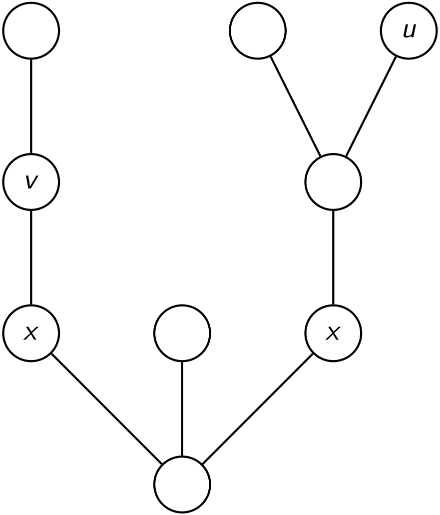


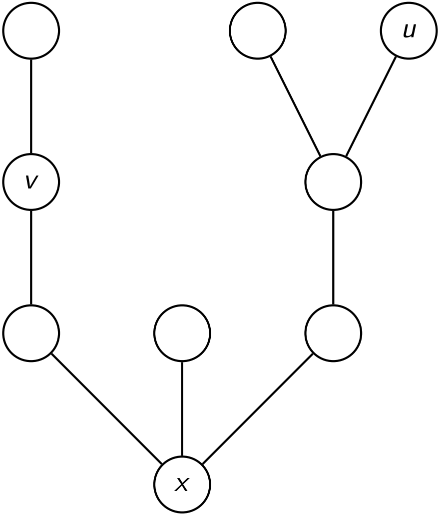
u
x




u
v


u
v
x


u
v
x
x



u
v
x

```cpp
int p[MAXN], d[MAXN];
void lca_dfs(vvi& g, int x, int q, int w)
{ // Hjálparfall.
    int i;
    d[x] = w, p[x] = q;
    for (i = 0; i < g[x].size(); i++)
        if (g[x][i] != q) lca_dfs(g, g[x][i], x, w + 1);
}

void lca_init(vvi& g, int x)
{ // Upphafstillir fyrir netið g með rót x.
    int i, n = g.size();
    lca_dfs(g, 0, x, x);
}

int lca(int u, int v)
{ // Skilar hæsta sameiginlega forfaðir u og v.
    if (d[u] < d[v]) swap(u, v);
    while (d[u] != d[v]) u = p[u];
    while (u != v) u = p[u], v = p[v];
    return u;
}
```

- 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.
- Fyrra 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 hnútinn í 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.

```
int p[MAXN][MAXK], d[MAXN];
void lca_dfs(vvi& g, int x, int q, int w)
{ // Hjálparfall.
    int i;
    d[x] = w;
    for (i = 0; i < MAXK; i++) p[x][i] = i == 0 ? q : p[p[x][i - 1]][i - 1];
    for (i = 0; i < g[x].size(); i++) if (g[x][i] != q)
        lca_dfs(g, g[x][i], x, w + 1);
}

void lca_init(vvi& g, int x)
{ // Upphafstillir fyrir netið g með rót x.
    int i, n = g.size();
    for (i = 0; i < MAXK; i++) p[x][i] = x;
    lca_dfs(g, 0, x, x);
}

int lca(int u, int v)
{ // Skilar hæsta sameiginlega forfaðir u og v.
    int i;
    if (d[u] < d[v]) swap(u, v);
    for (i = MAXK - 1; i >= 0; i--) if (d[p[u][i]] >= d[v]) u = p[u][i];
    for (i = MAXK - 1; i >= 0; i--) if (p[u][i] != p[v][i])
        u = p[u][i], v = p[v][i];
    return u == v ? u : p[u][0];
}
```

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