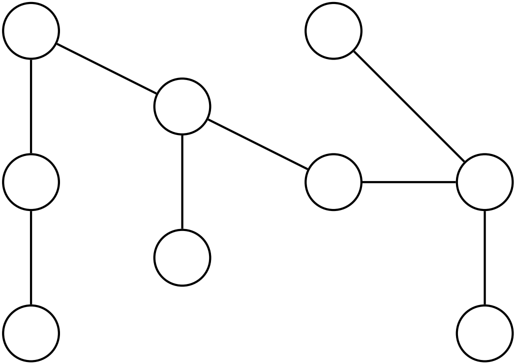
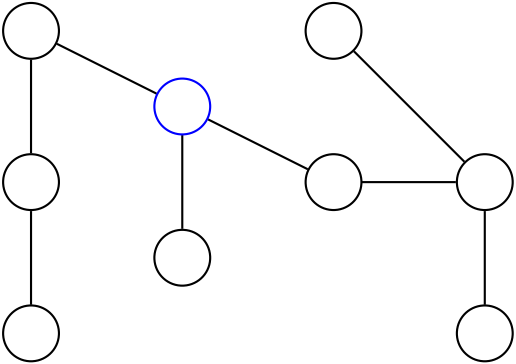
Næsti sameiginlegi forfaðir

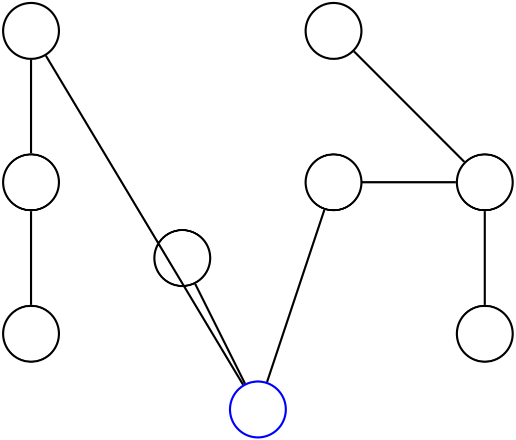
Bergur Snorrason

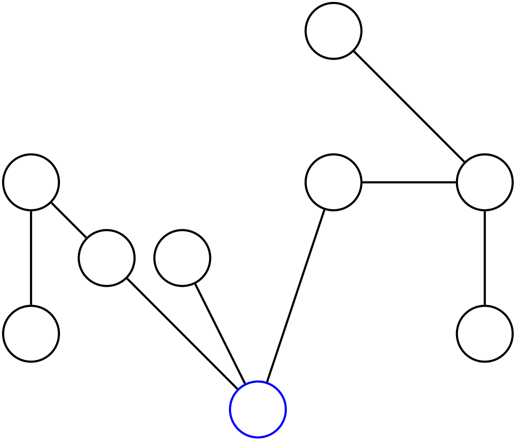
3. apríl 2023

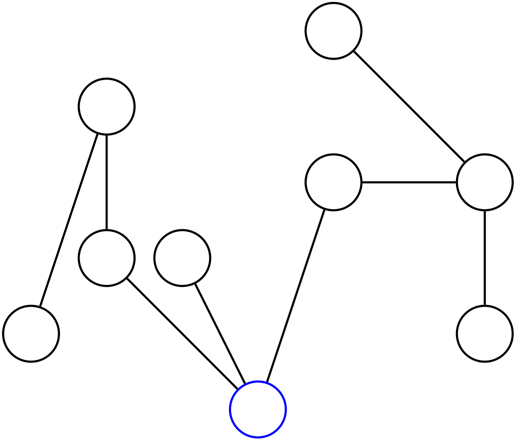
- Gerum ráð fyrir að við séum með tré.
- Látum einn hnút tákna rót trésins.
- Við getum þá talað um *metorð* (e. *rank*) hnúts í trénu.
- Metorð hnútsins er fjarlægðin frá hnútnum í rótina.
- Þar sem við erum í tréi er aðeins einn einfaldur vegur í rótina.
- Ein leið til að finna metorð allra hnúta er með einni dýptarleit.
- Látum metorð hnútsins u vera $r(u)$.
- Við segjum líka að metorð trés sé R ef hnúturinn með hæsta metorð er R.
- Oft er notað önnuð orð en „metorð", til dæmis „hæð" (e. *height*) eða „dýpt" (e. *depth*).

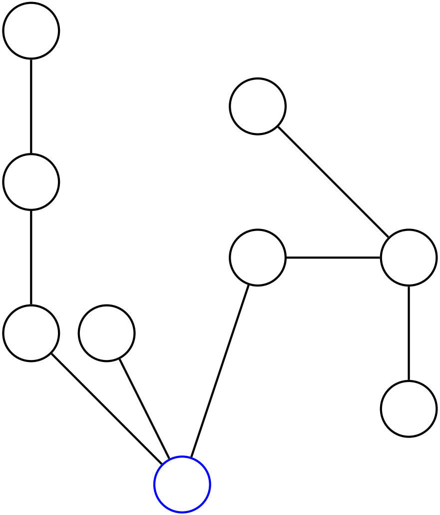


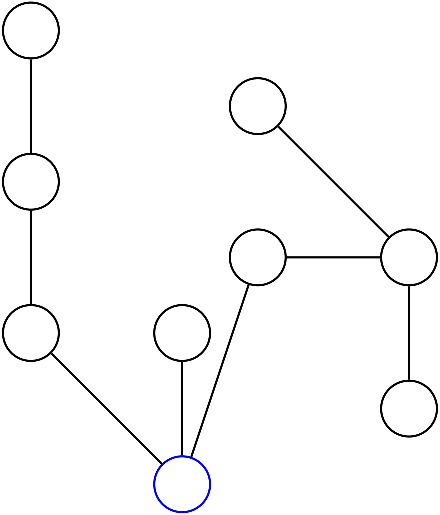


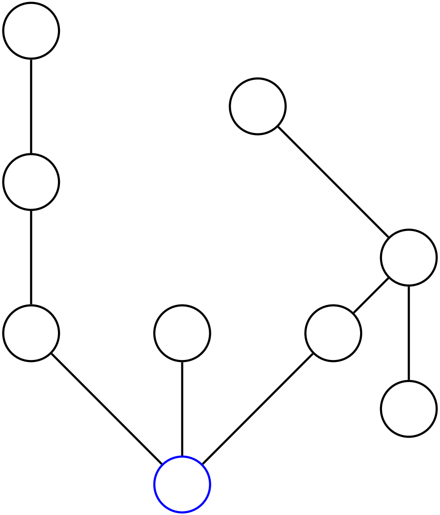


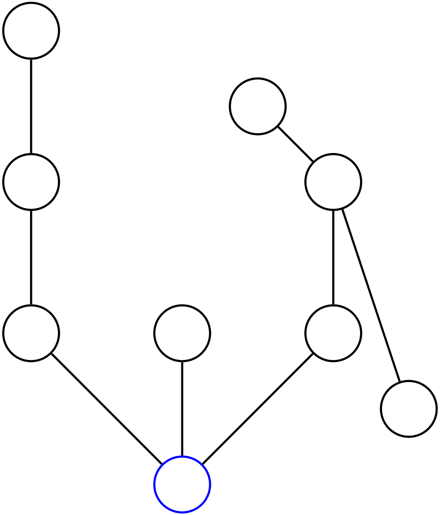


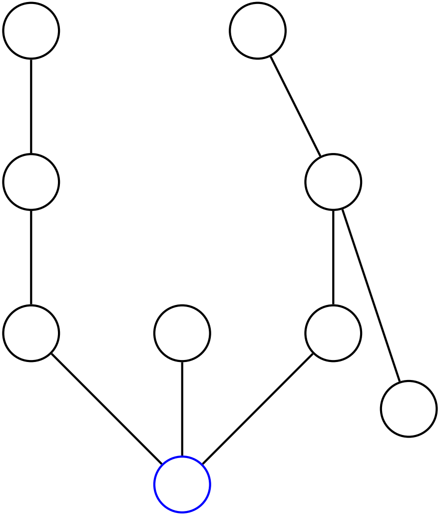


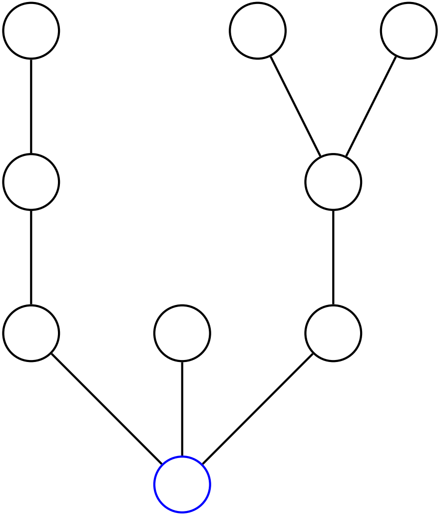


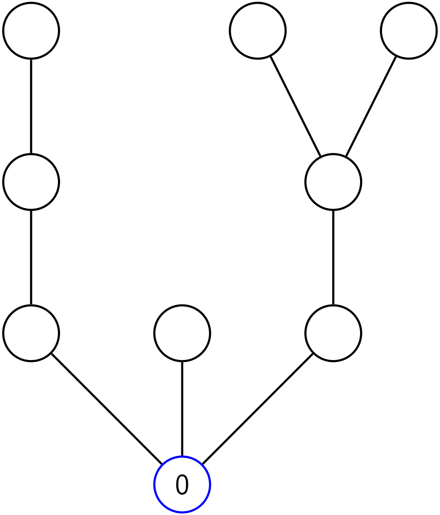


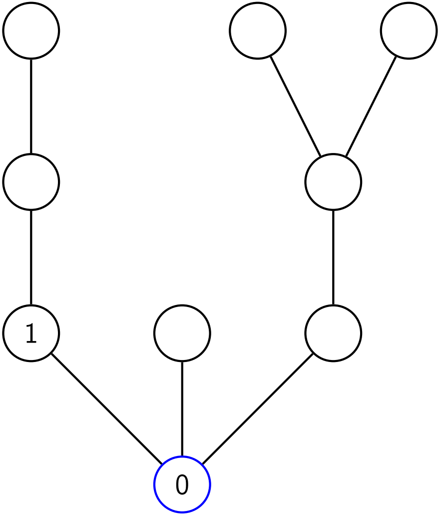


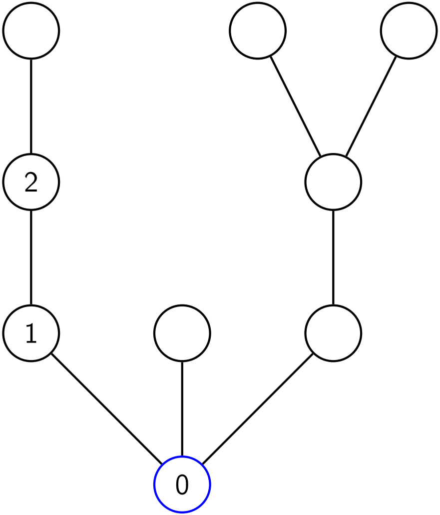




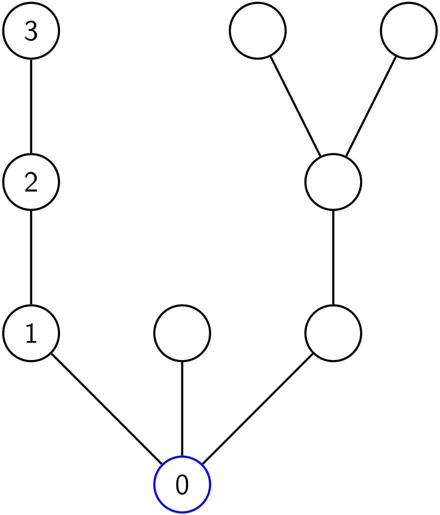


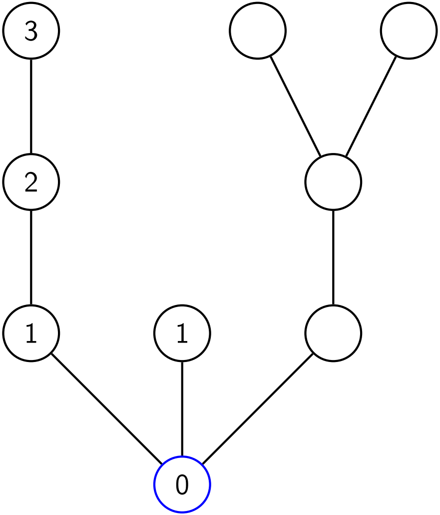

0

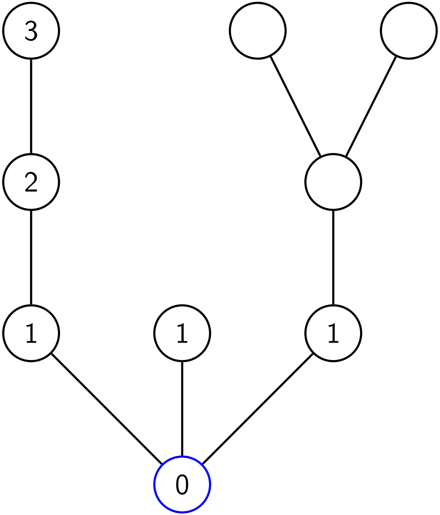

1
0

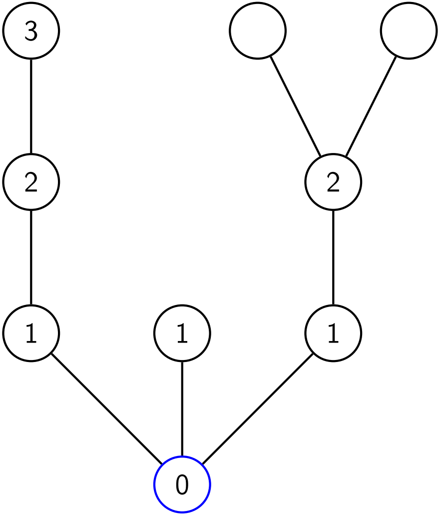


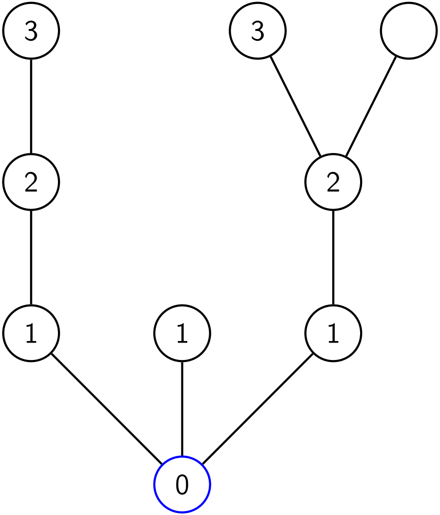
2
1
0

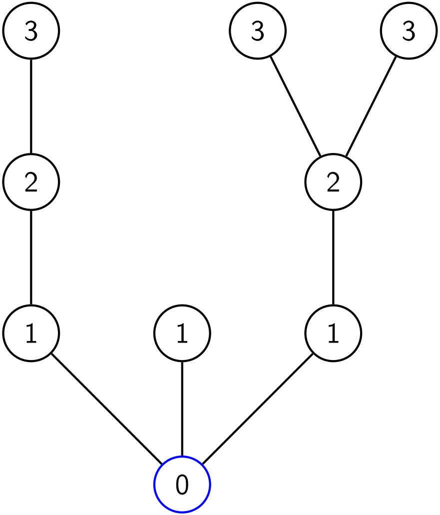

3
2
1
0


3
2
1
1
0

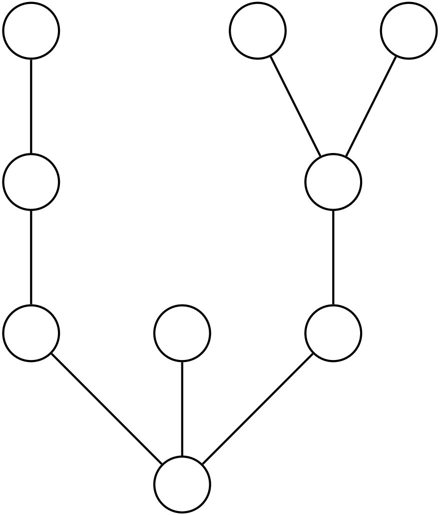

3
2
1
1
1
0

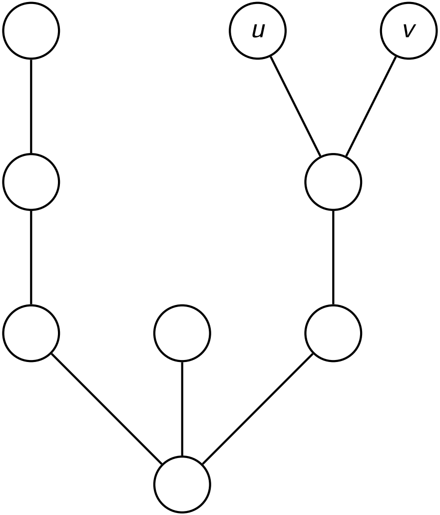

3
2
2
1
1
1
0

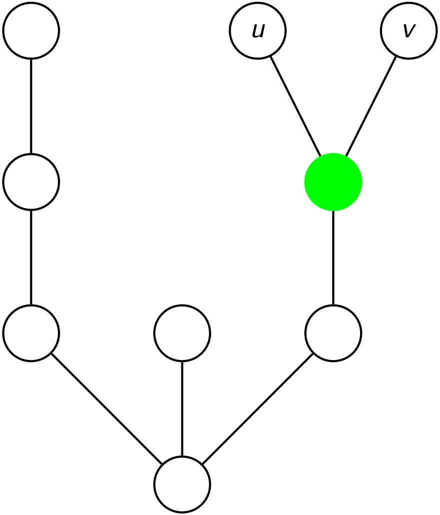

3
3
2
2
1
1
1
0


3
3
3
2
2
1
1
1
0

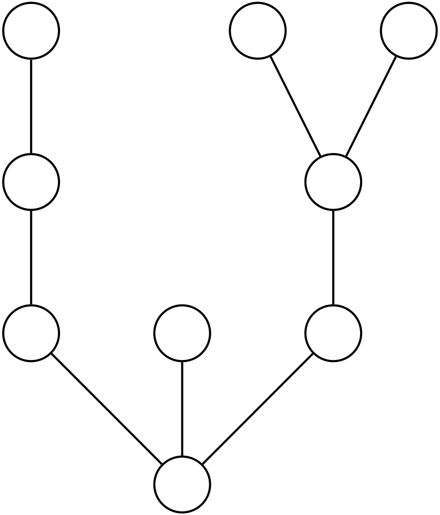
- Við köllum þann nágranna hnúts sem er nær rótinni *foreldri* (e. *parent*) hnútsins.
- Við getum þá fundið veginn að rótinni með því að ferðast eftir foreldrum.
- Þeir hnútar sem eru á veginum að rótinni kallast *forfeður* (e. *ancestors*) hnúts.
- Það er kannski skrýtið, en allir hnútar er forfeður sínir.
- Oft nýtist okkur að vita hvaða sameiginlegi forfaðir tveggja hnúta er með hæsta metorð.
- Forfaðirinn sem er með hæsta metorð kallast *næsti sameiginlegi forfaðir* hnútanna.

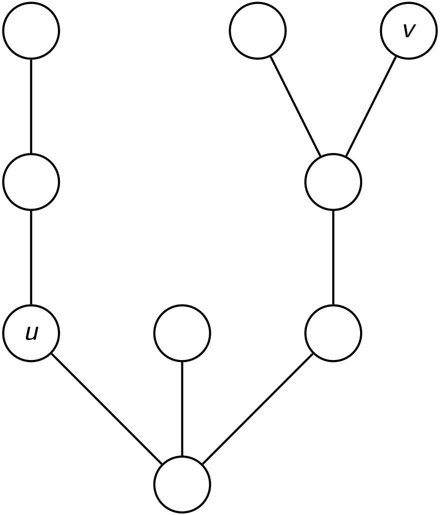



u
v

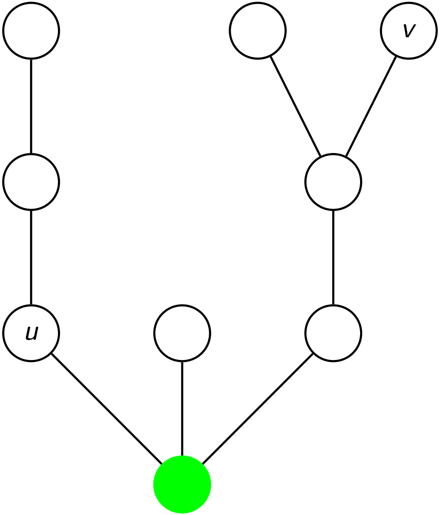


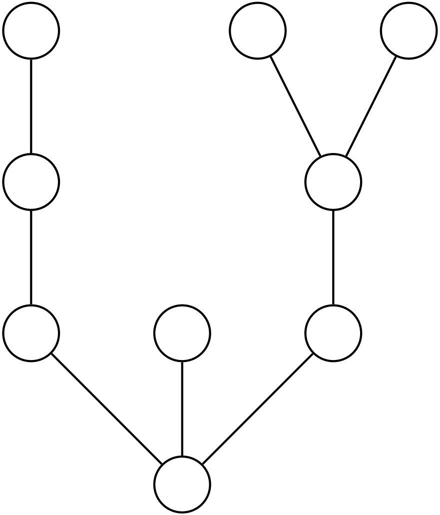
u
v

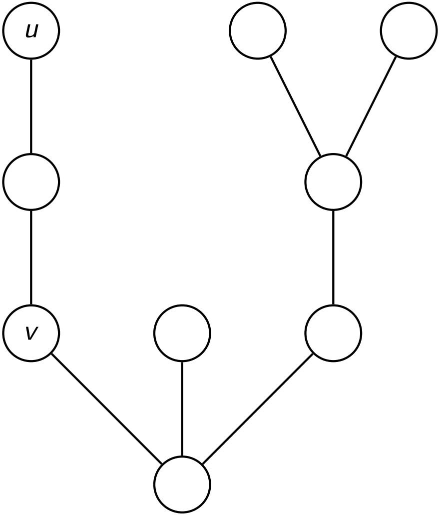




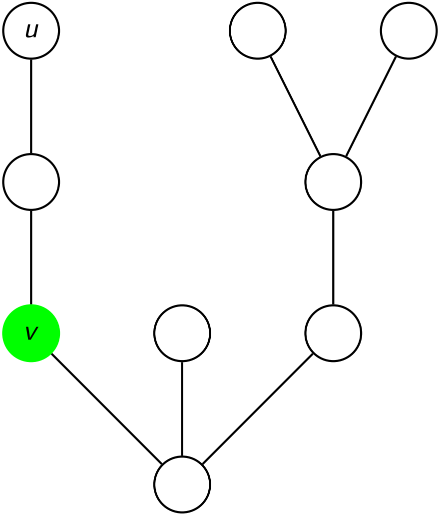
v
u

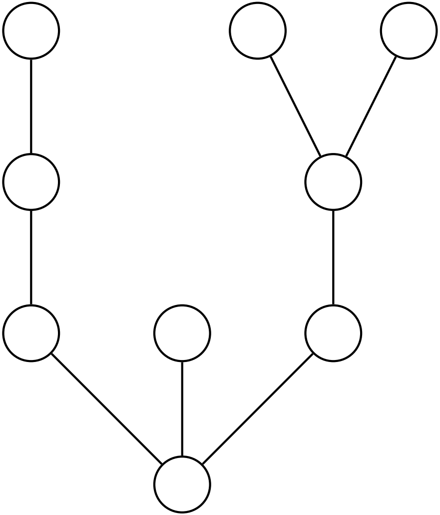

v
u



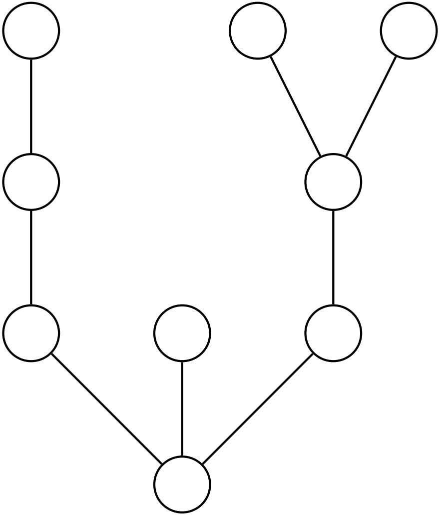


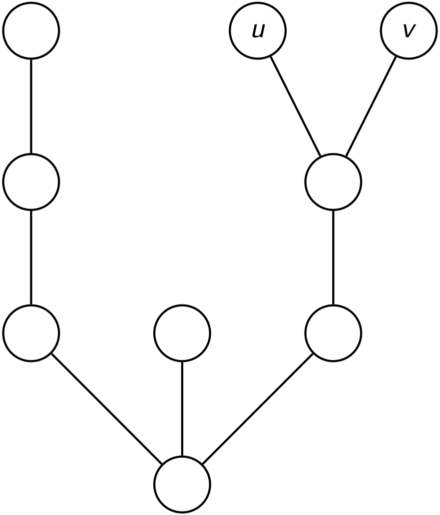
u
v

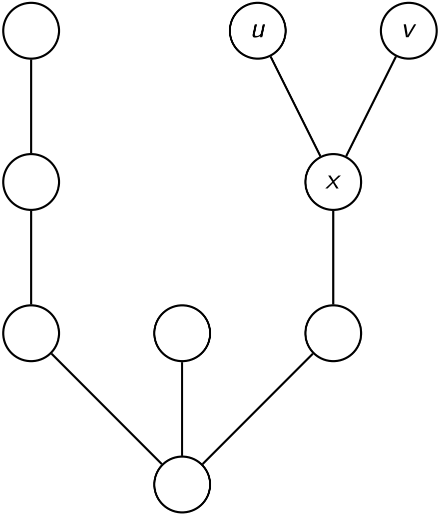

u
v

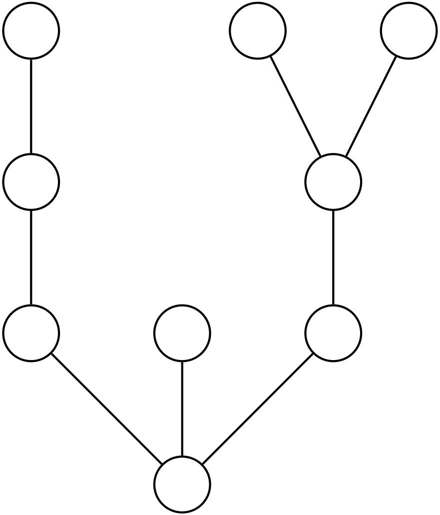


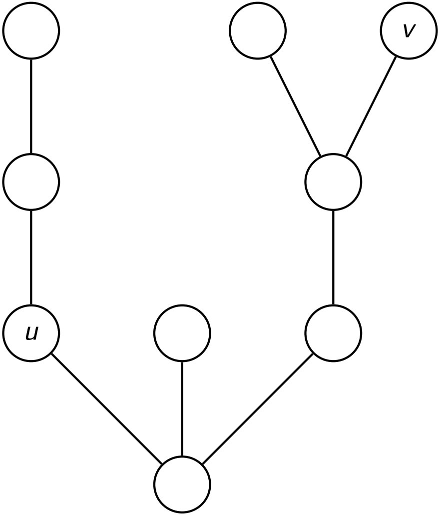
- Hvernig finnum við næsta sameiginlega forföður tveggja hnúta?
- Látum u og v tákna hnútana og x næsta sameiginlega forföður þeirra.
- Við getum gert ráð fyrir að u sé lengra frá rótinni en v, það er að segja $r(u) \geq r(v)$.
- Við vitum að allir forfeður u sem hafa metorð stærra en $r(v)$ eru ekki x.
- Svo við getum ferðast frá u að rótinni (eftir foreldrum, það er að segja) 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 sýnidæmin sem við sáum áðan.

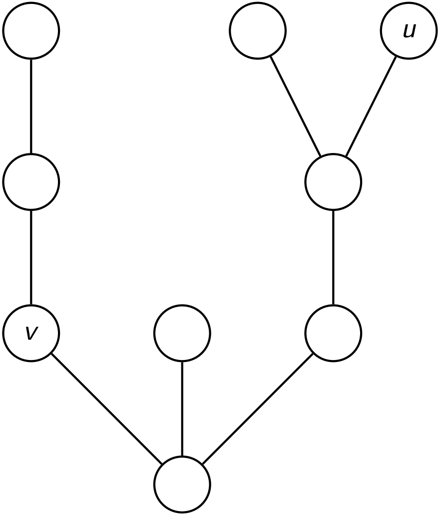


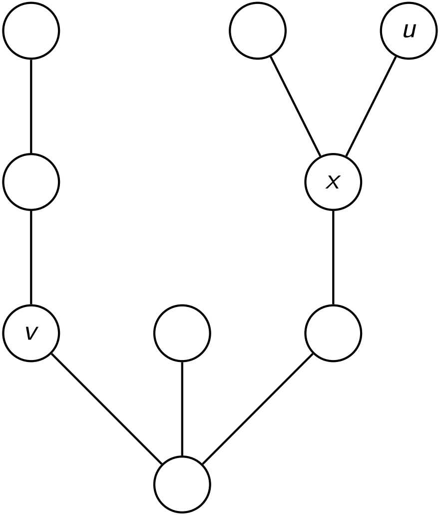

u
v

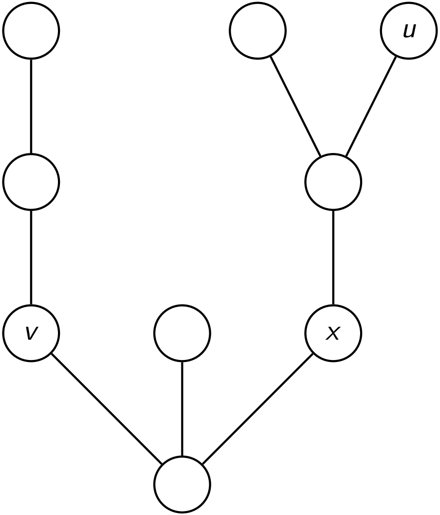

u
v
x

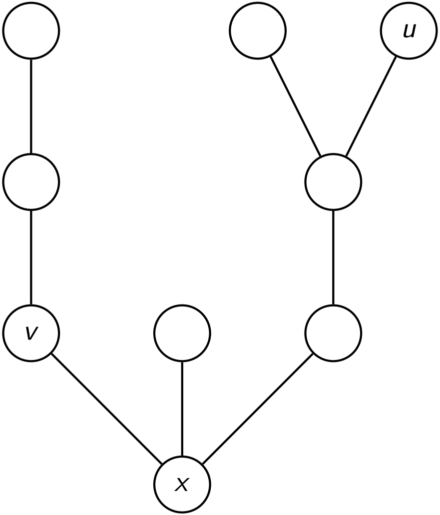


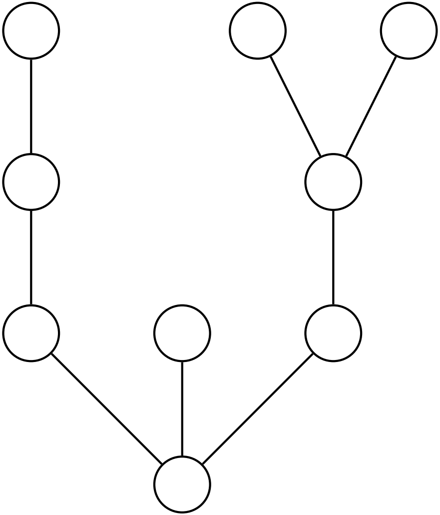

v
u

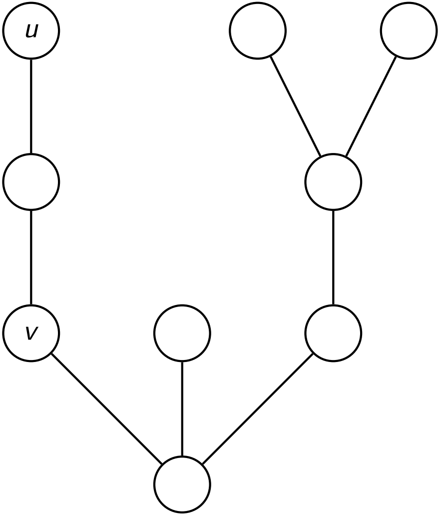

u
v

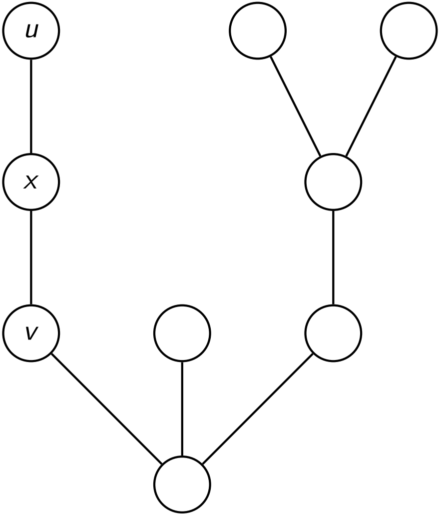

u
x
v

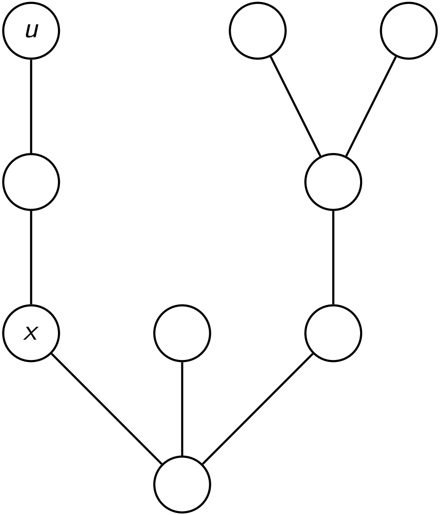

u
v
x


u
v
x

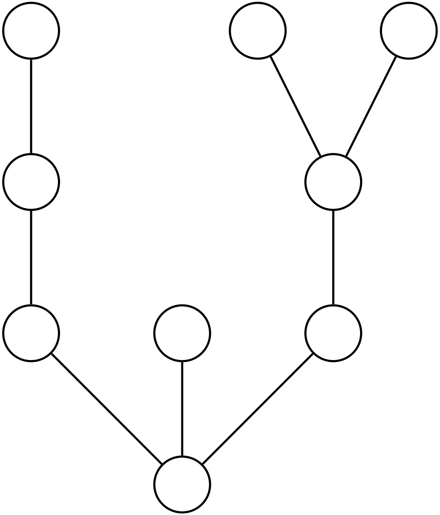


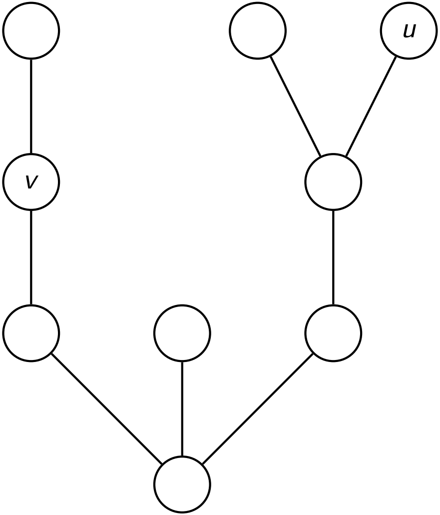

u
v


u
x
v

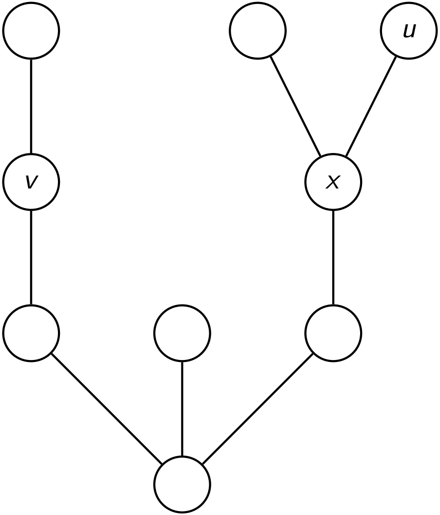


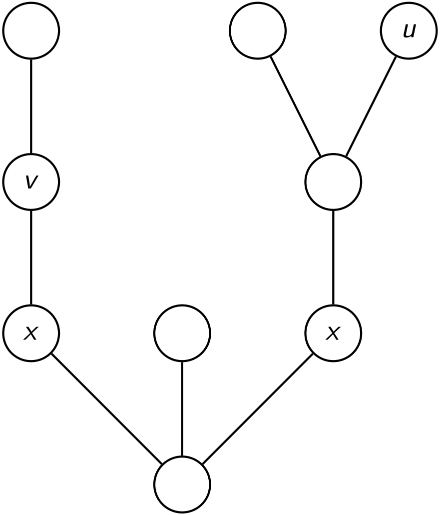
u
x



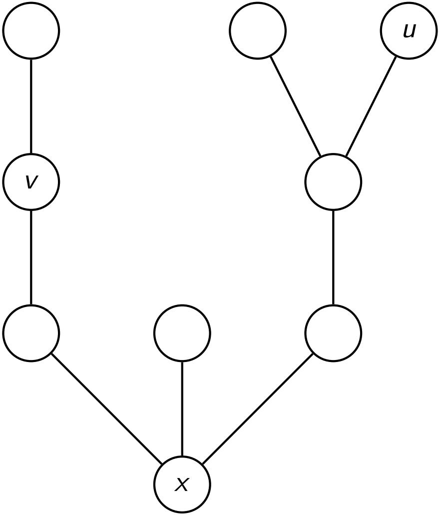


u
v


u
v
x



u
v
x
x



u
v
x

```
int p[MAXN], r[MAXN];
void lca_dfs(vvi& g, int x, int q, int w)
{
    int i;
    r[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)
{
    lca_dfs(g, 0, x, x);
}

int lca(int u, int v)
{
    if (r[u] < r[v]) swap(u, v);
    while (r[u] != r[v]) u = p[u];
    while (u != v) u = p[u], v = p[v];
    return u;
}
```

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

- Aðferðin skiptist í tvö skref:
  - Jöfnum metorð hnútanna.
  - Löbbum saman að rótinni þangað til við finnum svarið.
- Fyrra skrefinu má lýsa nánar.
- Látum hnútana okkar vera u og v, þannig að $r(u) \geq r(v)$.
- Við viljum því ferðast nákvæmlega $r(v) - r(u)$ sinnum í áttina að rótinni.
- 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 $\mathcal{O}(\log R)$ stökk fyrir hvern hnút.
- Táknum með $p(u, k)$ þann hnút sem þú endar í ef þú ferðast 2^k sinnum frá u gegnum foreldrin.
- Til þæginda segjum við að foreldri rótarinnar sé rótin sjálf.
- Til dæmis er $p(u, 0)$ foreldri u.
- Við finnum þessi gildi með rakningunni $p(u, i) = p(p(u, i-1), i-1)$.

- Við tökum því eins löng stökk og við getum án þess að $r(u) < r(v)$ þangað til $r(u) = r(v)$.
- Við getum því núna gert ráð fyrir að $r(u) = r(v)$.
- Þá viljum við taka eins löng stökk og við getum þannig að $u \neq v$.
- Að því loknu munum við hafa þrjú tilvik:
  - u og v hafa sama foreldri.
  - u er foreldri v.
  - v er foreldri u.

```
int p[MAXN][MAXK], r[MAXN];
void lca_dfs(vvi& g, int x, int q, int w)
{
    int i;
    r[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)
{
    int i;
    for (i = 0; i < MAXK; i++) p[x][i] = x;
    lca_dfs(g, 0, x, x);
}

int lca(int u, int v)
{
    int i;
    if (r[u] < r[v]) swap(u, v);
    for (i = MAXK - 1; i >= 0; i--) if (r[p[u][i]] >= r[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 $\mathcal{O}(\log R)$.
- Svo tímaflækjan er $\mathcal{O}(\log R)$ fyrir hverja fyrirspurn.
- Ef tréð hefur n hnúta er þarf í versta falli $\mathcal{O}(\log n)$ tíma.