Næsti sameiginlegi forfaðir

Bergur Snorrason

April 8, 2024

- Gerum ráð fyrir að við séum með tré.

- Gerum ráð fyrir að við séum með tré.
- Látum einn hnút tákna rót trésins.

- 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.

- 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.

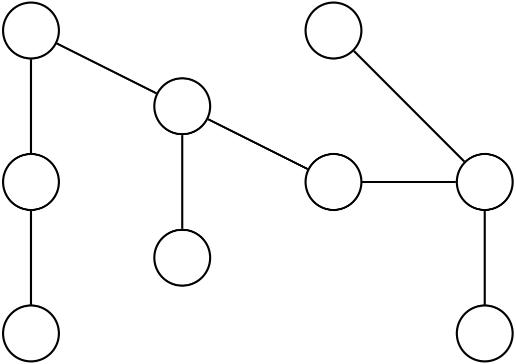
- Gerum ráð fyrir að við séum með tré.
- Látum einn hnút tákna rót trésins.
- Við getum þá talað um *metorð* (e. *rank*) hnúts í trénu.
- Metorð hnútsins er fjarlægðin frá hnútnum í rótina.
- Þar sem við erum í tréi er aðeins einn einfaldur vegur í rótina.

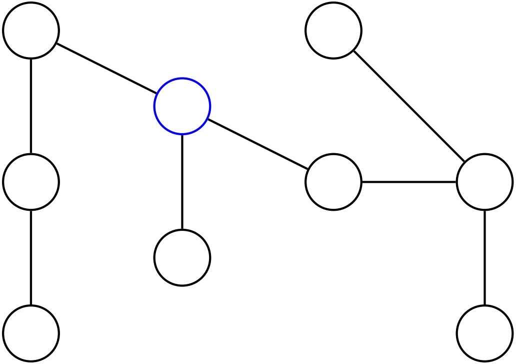
- Gerum ráð fyrir að við séum með tré.
- Látum einn hnút tákna rót trésins.
- Við getum þá talað um *metorð* (e. *rank*) hnúts í trénu.
- Metorð hnútsins er fjarlægðin frá hnútnum í rótina.
- Þar sem við erum í tréi er aðeins einn einfaldur vegur í rótina.
- Ein leið til að finna metorð allra hnúta er með einni dýptarleit.

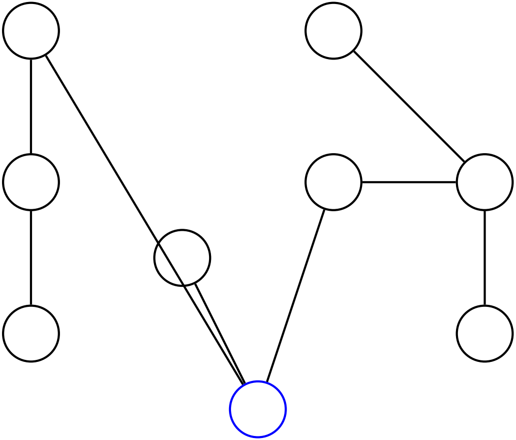
- Gerum ráð fyrir að við séum með tré.
- Látum einn hnút tákna rót trésins.
- Við getum þá talað um *metorð* (e. *rank*) hnúts í trénu.
- Metorð hnútsins er fjarlægðin frá hnútnum í rótina.
- Þar sem við erum í tréi er aðeins einn einfaldur vegur í rótina.
- Ein leið til að finna metorð allra hnúta er með einni dýptarleit.
- Látum metorð hnútsins u vera $r(u)$.

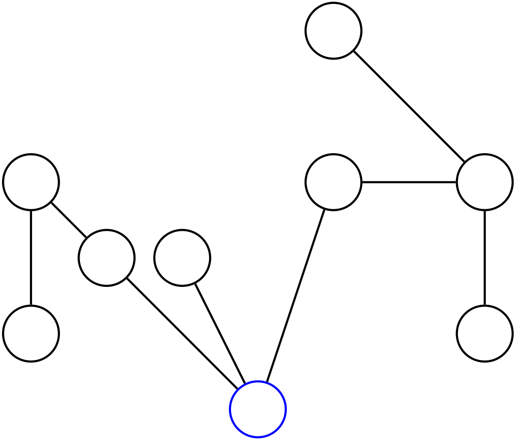
- Gerum ráð fyrir að við séum með tré.
- Látum einn hnút tákna rót trésins.
- Við getum þá talað um *metorð* (e. *rank*) hnúts í trénu.
- Metorð hnútsins er fjarlægðin frá hnútnum í rótina.
- Þar sem við erum í tréi er aðeins einn einfaldur vegur í rótina.
- Ein leið til að finna metorð allra hnúta er 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.

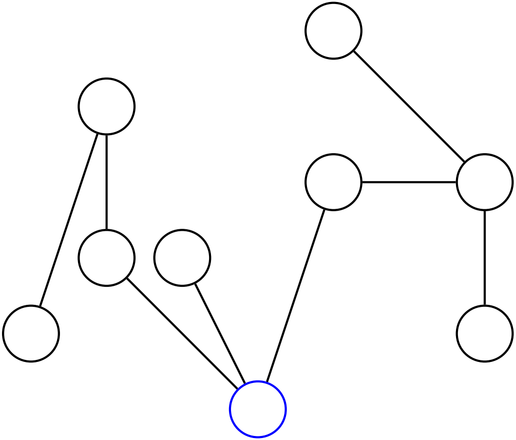
- Gerum ráð fyrir að við séum með tré.
- Látum einn hnút tákna rót trésins.
- Við getum þá talað um *metorð* (e. *rank*) hnúts í trénu.
- Metorð hnútsins er fjarlægðin frá hnútnum í rótina.
- Þar sem við erum í tréi er aðeins einn einfaldur vegur í rótina.
- Ein leið til að finna metorð allra hnúta er 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*).

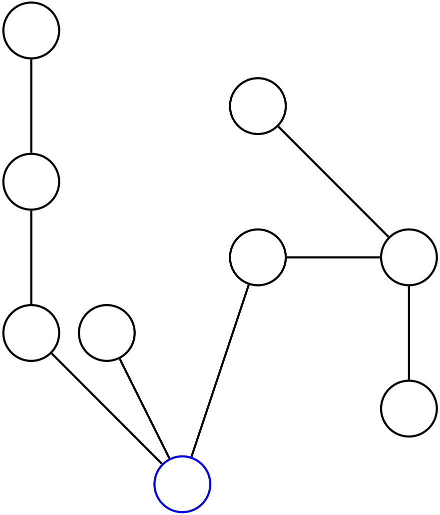


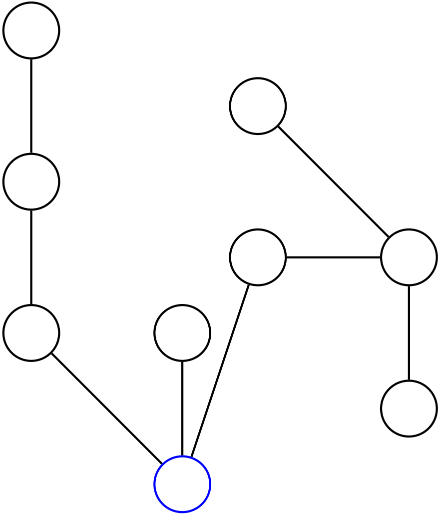


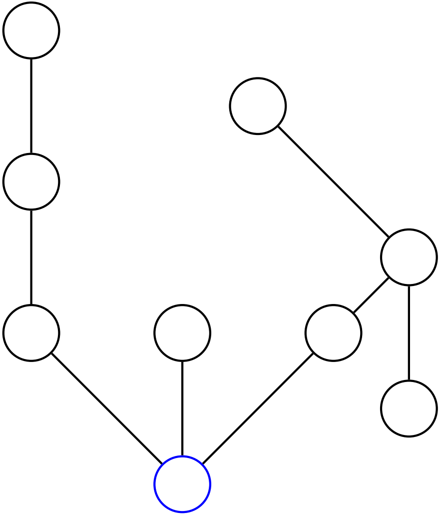


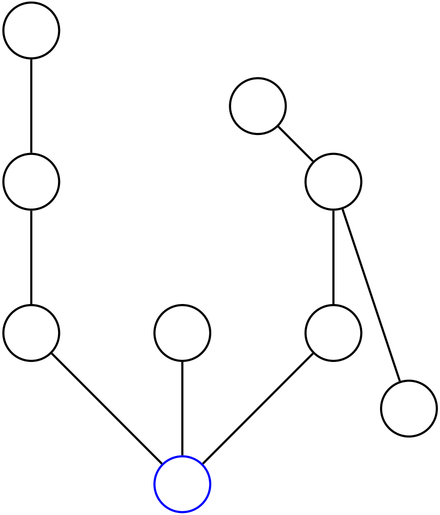


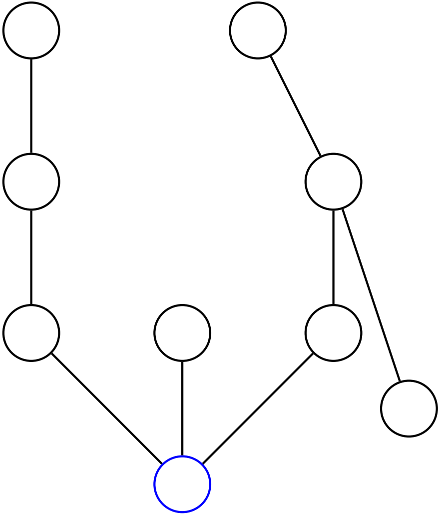


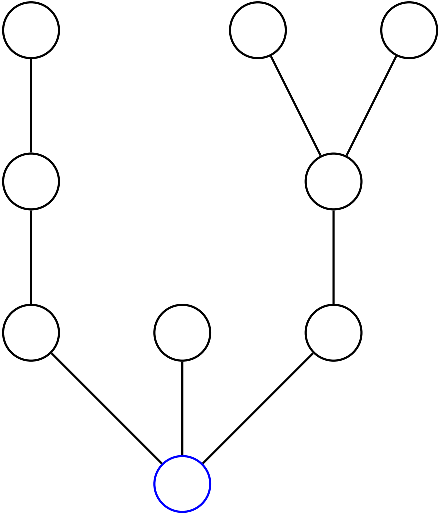


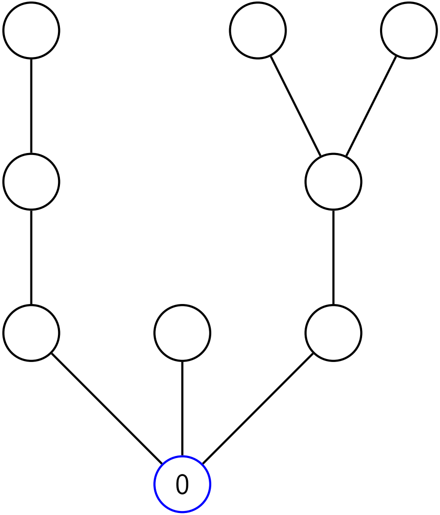


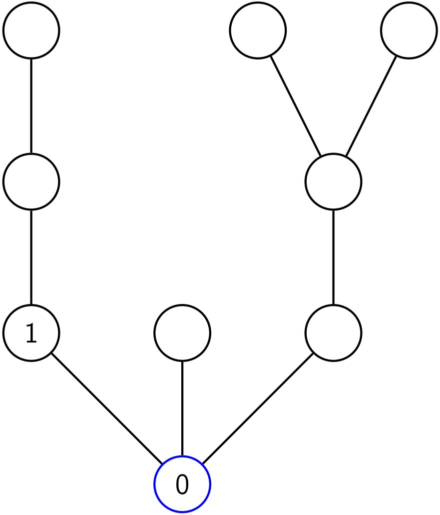


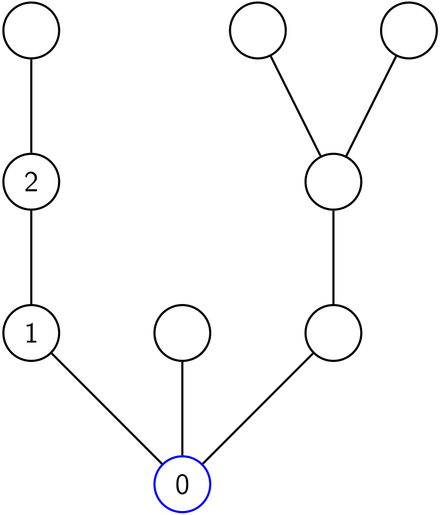


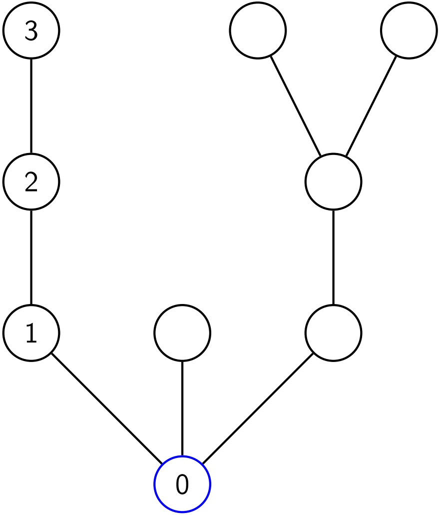


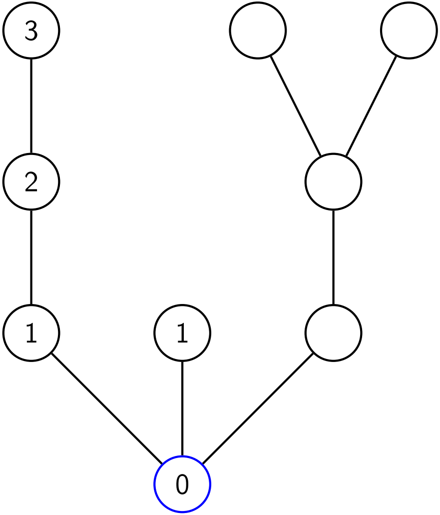


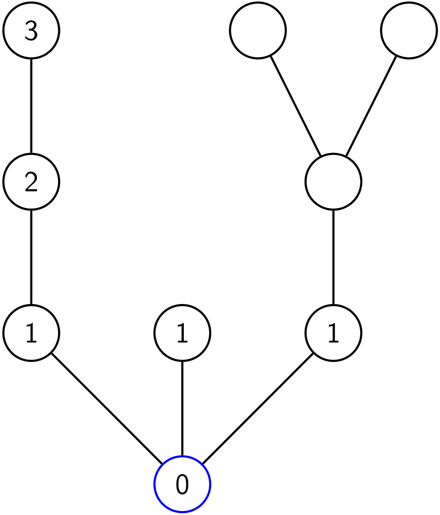

0

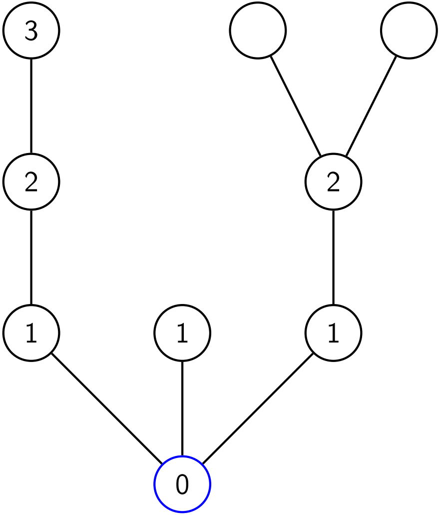

1
0

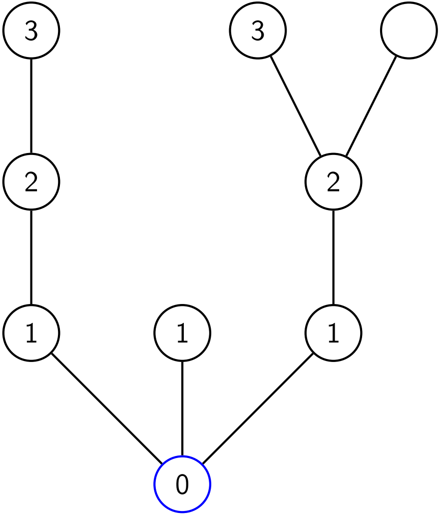

2
1
0

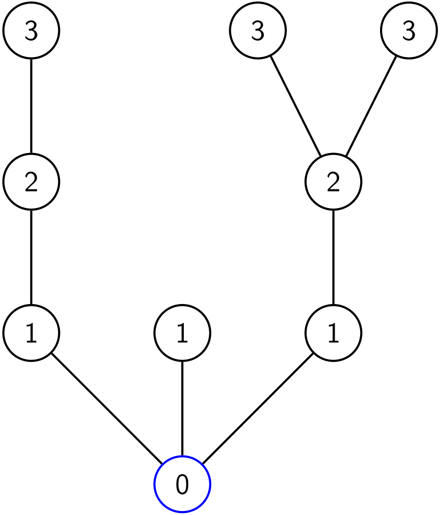

3
2
1
0


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.

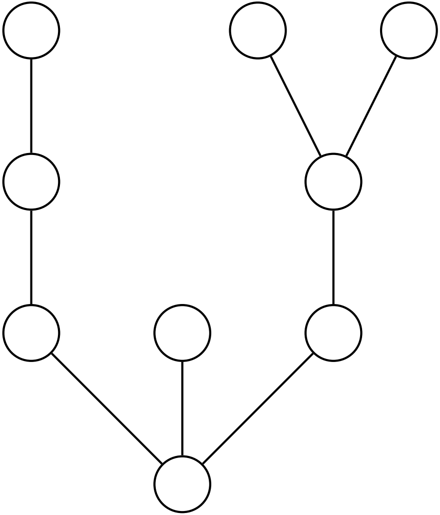
- Við köllum þann nágranna hnúts sem er nær rótinni *foreldri* (e. *parent*) hnútsins.
- Við getum þá fundið veginn að rótinni með því að ferðast eftir foreldrum.

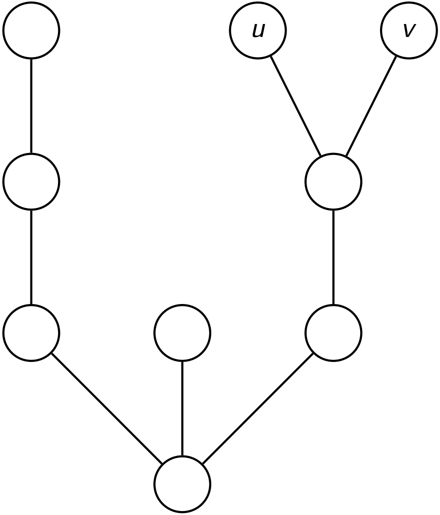
- Við köllum þann nágranna hnúts sem er nær rótinni *foreldri* (e. *parent*) hnútsins.
- Við getum þá fundið veginn að rótinni með því að ferðast eftir foreldrum.
- Þeir hnútar sem eru á veginum að rótinni kallast *forfeður* (e. *ancestors*) hnúts.

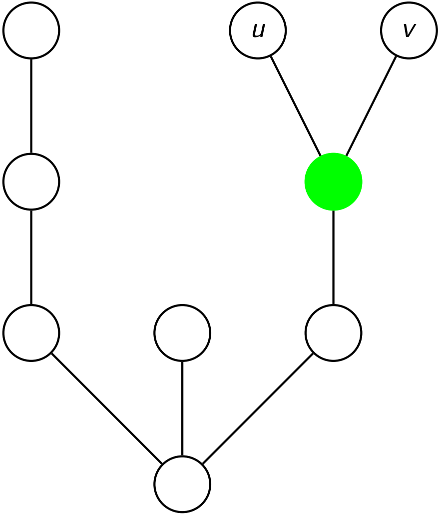
- Við köllum þann nágranna hnúts sem er nær rótinni *foreldri* (e. *parent*) hnútsins.
- Við getum þá fundið veginn að rótinni með því að ferðast eftir foreldrum.
- Þeir hnútar sem eru á veginum að rótinni kallast *forfeður* (e. *ancestors*) hnúts.
- Það er kannski skrýtið, en allir hnútar er forfeður sínir.

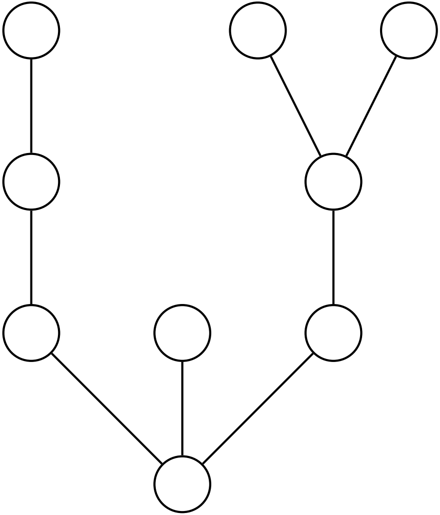
- Við köllum þann nágranna hnúts sem er nær rótinni *foreldri* (e. *parent*) hnútsins.
- Við getum þá fundið veginn að rótinni með því að ferðast eftir foreldrum.
- Þeir hnútar sem eru á veginum að rótinni kallast *forfeður* (e. *ancestors*) hnúts.
- Það er kannski skrýtið, en allir hnútar er forfeður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

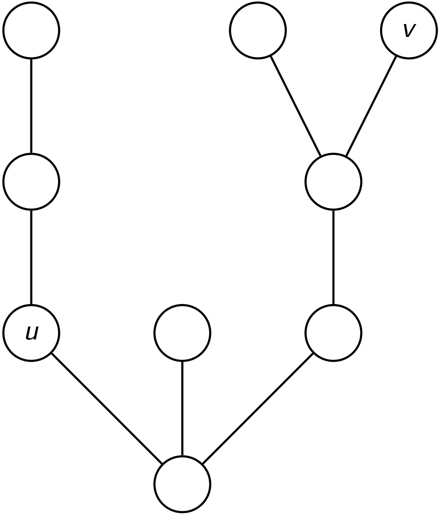
- Við köllum þann nágranna hnúts sem er nær rótinni *foreldri* (e. *parent*) hnútsins.
- Við getum þá fundið veginn að rótinni með því að ferðast eftir foreldrum.
- Þeir hnútar sem eru á veginum að rótinni kallast *forfeður* (e. *ancestors*) hnúts.
- Það er kannski skrýtið, en allir hnútar er forfeður 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.

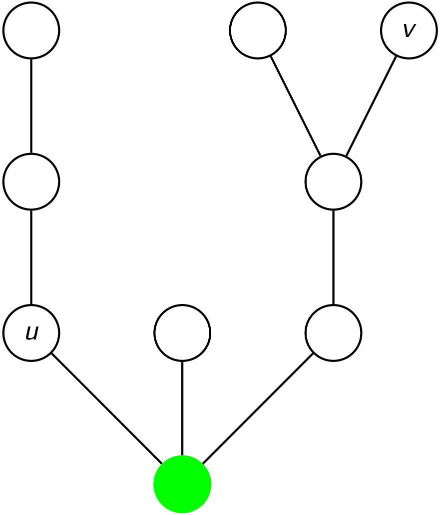


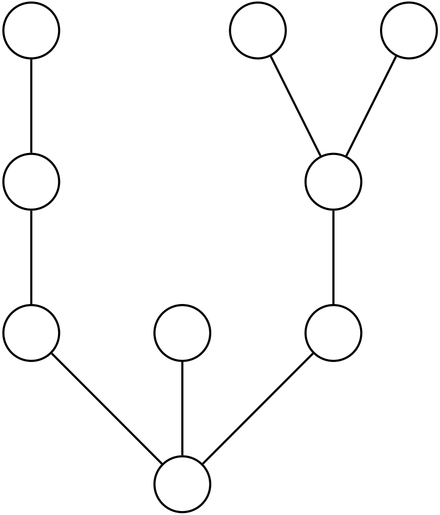

u
v

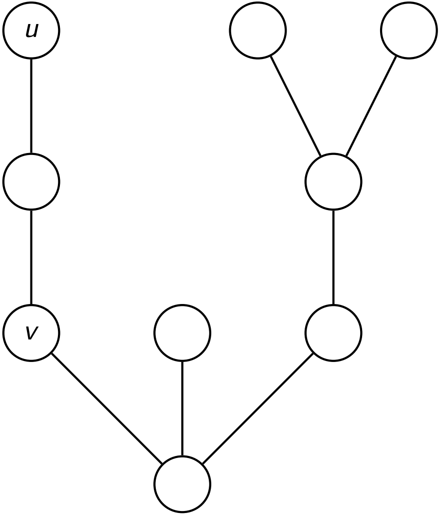

u
v

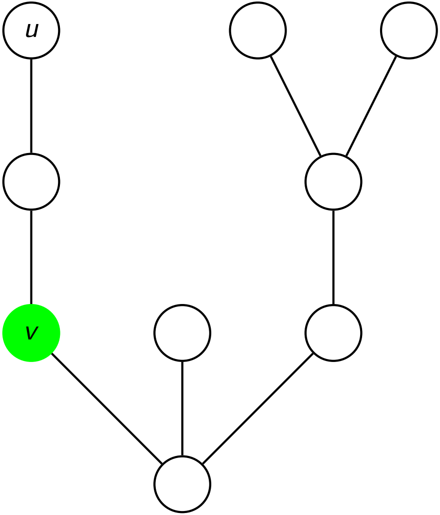


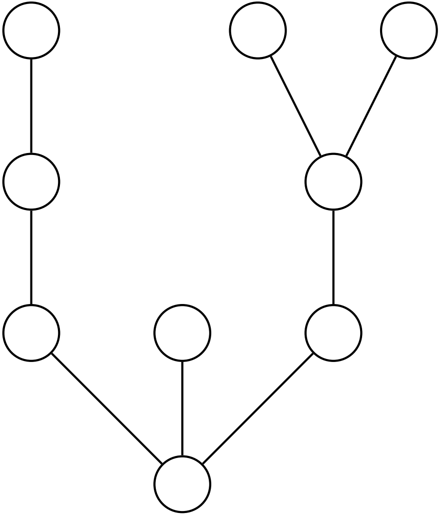

v
u


v
u




u
v


u
v



- Hvernig finnum við næsta sameiginlega forföður tveggja hnúta?

- Hvernig finnum við næsta sameiginlega forföður tveggja hnúta?
- Látum u og v tákna hnútana og x næsta sameiginlega forföður þeirra.

- 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 $r(u) \geq r(v)$.

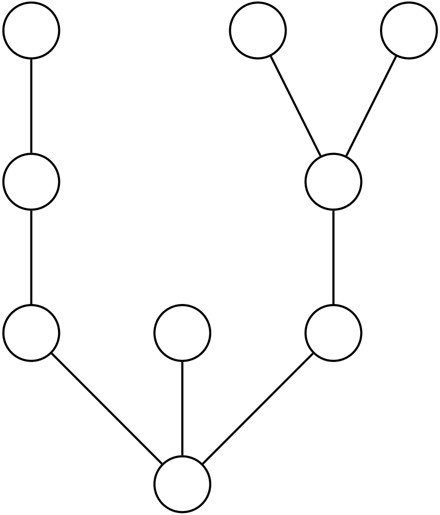
- Hvernig finnum við næsta sameiginlega forföður tveggja hnúta?
- Látum u og v tákna hnútana og x næsta sameiginlega forföður þeirra.
- Við getum gert ráð fyrir að u sé lengra frá rótinni en v, það er að segja $r(u) \geq r(v)$.
- Við vitum að allir forfeður u sem hafa metorð stærra en $r(v)$ eru ekki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

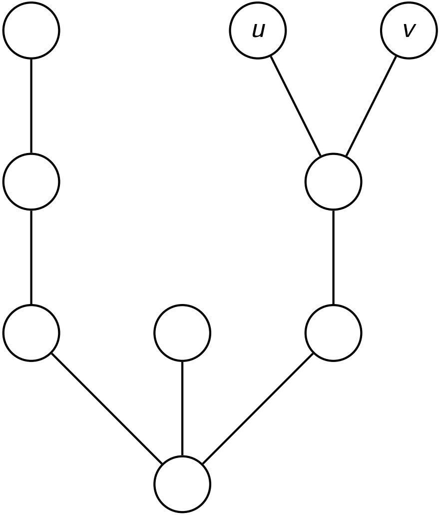
- Hvernig finnum við næsta sameiginlega forföður tveggja hnúta?
- Látum u og v tákna hnútana og x næsta sameiginlega forföður þeirra.
- Við getum gert ráð fyrir að u sé lengra frá rótinni en v, það er að segja $r(u) \geq r(v)$.
- Við vitum að allir forfeður u sem hafa metorð stærra en $r(v)$ eru ekki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

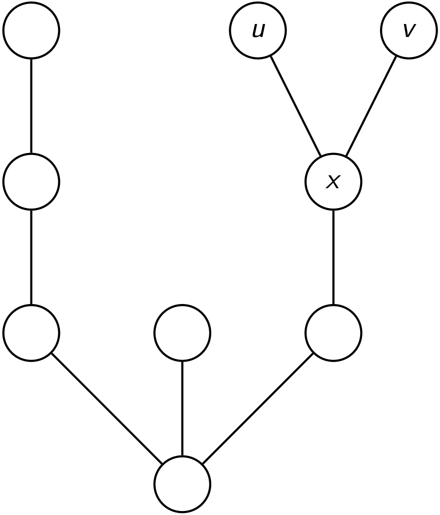
- Hvernig finnum við næsta sameiginlega forföður tveggja hnúta?
- Látum u og v tákna hnútana og x næsta sameiginlega forföður þeirra.
- Við getum gert ráð fyrir að u sé lengra frá rótinni en v, það er að segja $r(u) \geq r(v)$.
- Við vitum að allir forfeður u sem hafa metorð stærra en $r(v)$ eru ekki x.
- Svo við getum ferðast frá u að rótinni (eftir foreldrum, það er að segja) þar til við erum komin með sama metorð og v.
- Við getum svo ferðast eftir foreldrum beggja á sama tíma þangað til við lendum í sama hnútnum.

- 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 með sama metorð og v.
- Við getum svo ferðast eftir foreldrum beggja á sama tíma þangað til við lendum í sama hnútnum.
- Sá hnútur er x.

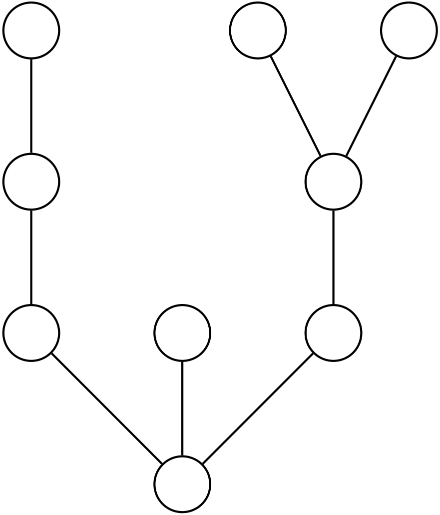
- Hvernig finnum við næsta sameiginlega forföður tveggja hnúta?
- Látum u og v tákna hnútana og x næsta sameiginlega forföður þeirra.
- Við getum gert ráð fyrir að u sé lengra frá rótinni en v, það er að segja $r(u) \geq r(v)$.
- Við vitum að allir forfeður u sem hafa metorð stærra en $r(v)$ eru ekki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

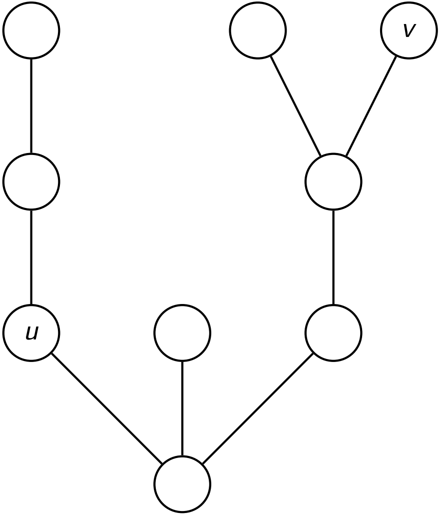


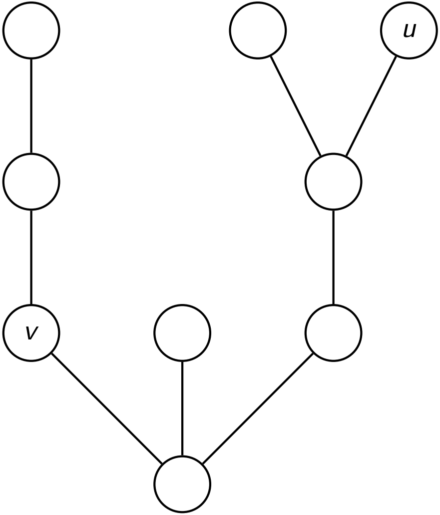

u
v

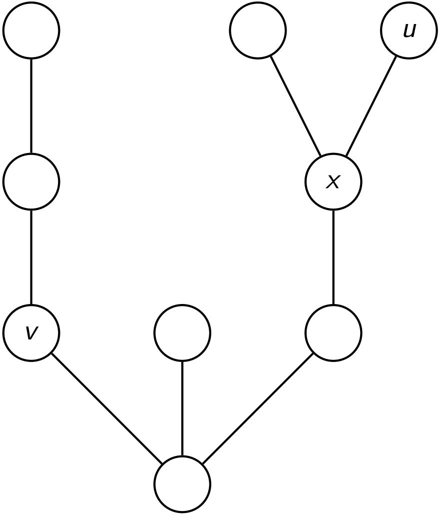


u
v
x

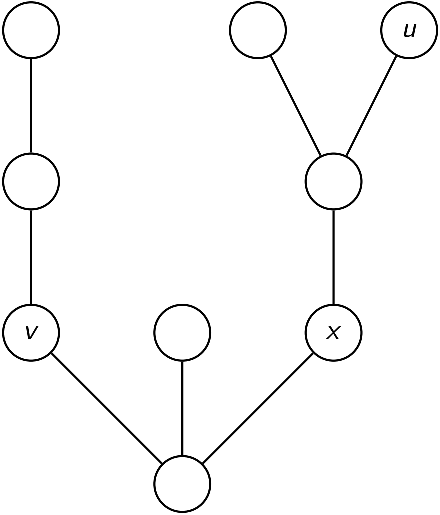


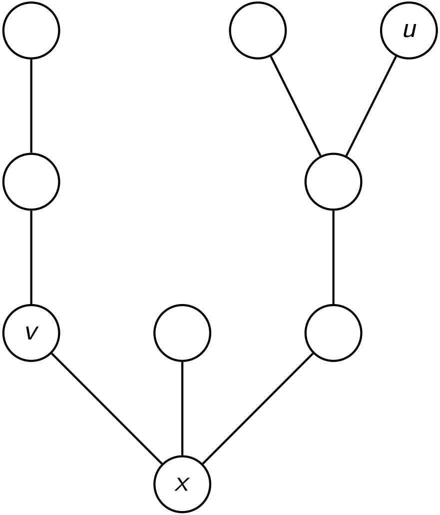

v
u

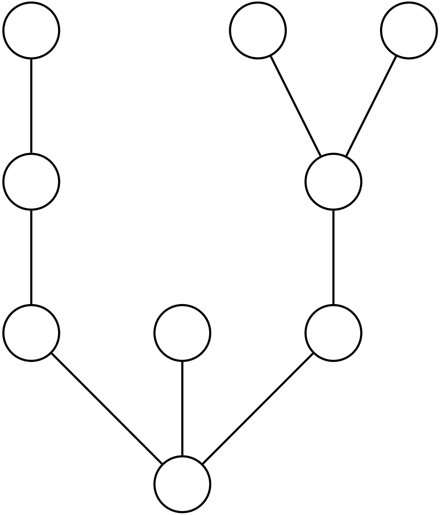

u
v

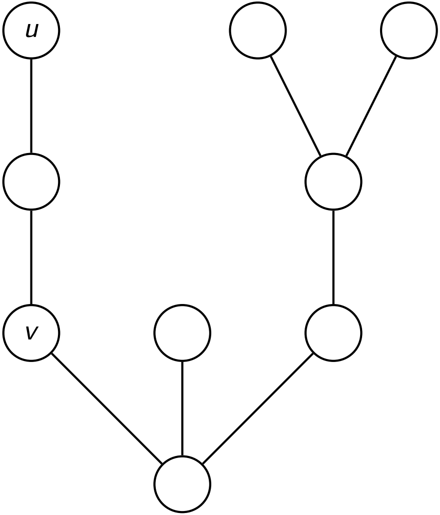


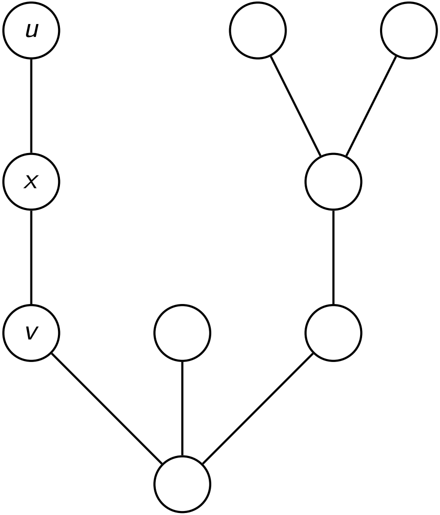
u
x
v

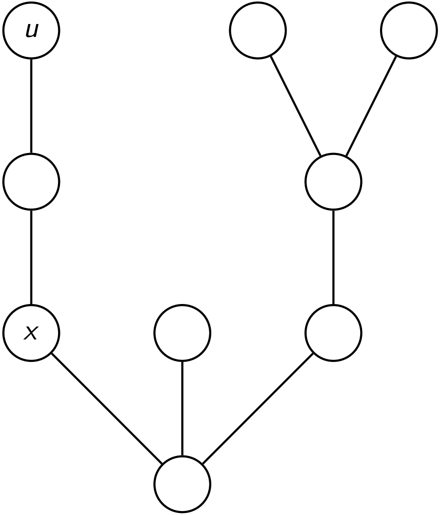

u
v
x


u
v
x

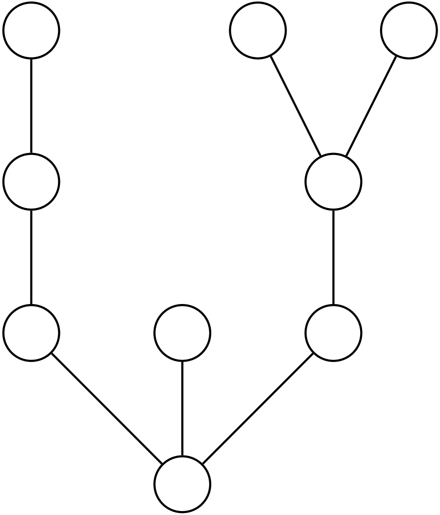


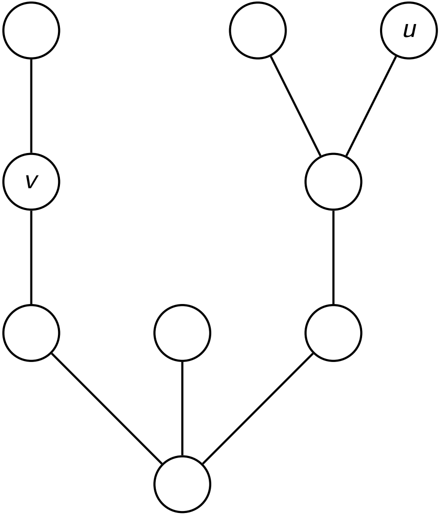

u
v

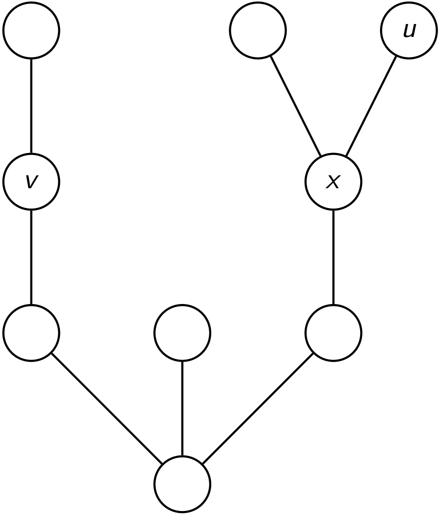

u
x
v

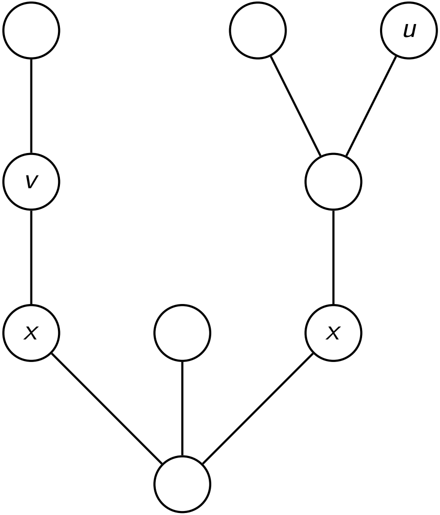


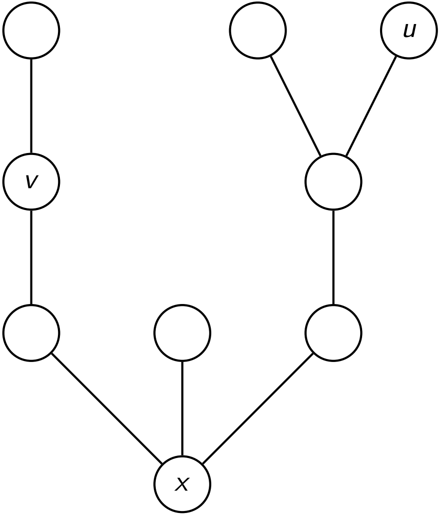
u
x




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)
{
    r[x] = w, p[x] = q;
    for (int 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, x, x, 0);
}

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.

- Gerum ráð fyrir að metorð trésins sé R.
- Þá er tímaflækjan á þessari aðferð $\mathcal{O}(\quad)$.

- Gerum ráð fyrir að metorð trésins sé R.
- Þá er tímaflækjan á þessari aðferð $\mathcal{O}(\,R\,)$.

- Gerum ráð fyrir að metorð trésins sé R.
- Þá er tímaflækjan á þessari aðferð $\mathcal{O}(\ R\ )$.
- Í versta falli er metorð trés með n hnúta .

- Gerum ráð fyrir að metorð trésins sé R.
- Þá er tímaflækjan á þessari aðferð $\mathcal{O}(R)$.
- Í versta falli er metorð trés með n hnúta $n - 1$.

- Gerum ráð fyrir að metorð trésins sé R.
- Þá er tímaflækjan á þessari aðferð $\mathcal{O}(\ R\ )$.
- Í versta falli er metorð trés með n hnúta $n - 1$.
- Svo tímaflækjan er í versta falli $\mathcal{O}(\quad)$.

- Gerum ráð fyrir að metorð trésins sé R.
- Þá er tímaflækjan á þessari aðferð $\mathcal{O}(\,R\,)$.
- Í versta falli er metorð trés með n hnúta $n-1$.
- Svo tímaflækjan er í versta falli $\mathcal{O}(\,n\,)$.

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

- Gerum ráð fyrir að metorð trésins sé R.
- Þá er tímaflækjan á þessari aðferð $\mathcal{O}(\, R\,)$.
- Í versta falli er metorð 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:

- Aðferðin skiptist í tvö skref:
  - Jöfnum metorð hnútanna.

- Aðferðin skiptist í tvö skref:
  - Jöfnum metorð hnútanna.
  - Löbbum saman að rótinni þangað til við finnum svarið.

- Aðferðin skiptist í tvö skref:
  - Jöfnum metorð hnútanna.
  - Löbbum saman að rótinni þangað til við finnum svarið.
- Fyrra skrefinu má lýsa nánar.

- 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 $r(u) \geq r(v)$.

- 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.

- 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 lausn er að geyma ekki bara foreldri hvers hnúts u, heldur alla forfeður u sem hafa metorð $r(u) - 2^k$.

- 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 lausn er að geyma ekki bara foreldri hvers hnúts u, heldur alla forfeður u sem hafa metorð $r(u) - 2^k$.
- Við þurfum því að geyma $\mathcal{O}($ $)$ stökk fyrir hvern hnút.

- 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 lausn er að geyma ekki bara foreldri hvers hnúts u, heldur alla forfeður u sem hafa metorð $r(u) - 2^k$.
- Við þurfum því að geyma $\mathcal{O}(\log R)$ stökk fyrir hvern hnút.

- 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 lausn er að geyma ekki bara foreldri hvers hnúts u, heldur alla forfeður u sem hafa metorð $r(u) - 2^k$.
- 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.

- 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 lausn er að geyma ekki bara foreldri hvers hnúts u, heldur alla forfeður u sem hafa metorð $r(u) - 2^k$.
- 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.

- 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 lausn er að geyma ekki bara foreldri hvers hnúts u, heldur alla forfeður u sem hafa metorð $r(u) - 2^k$.
- 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.

- 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 lausn er að geyma ekki bara foreldri hvers hnúts u, heldur alla forfeður u sem hafa metorð $r(u) - 2^k$.
- 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 $p(u, k) = p(p(u, k-1), k-1)$.

- Við tökum því eins löng stökk og við getum án þess að $r(u) < r(v)$ þangað til $r(u) = r(v)$.

- 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 $r(u) = r(v)$.

- 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 $u \neq v$.

- 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:

- 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.

- 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.

- 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.

```cpp
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, x, x, 0);
}

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 $\mathcal{O}(\qquad)$.

- Í hverju skrefi þurfum við bara að taka $\mathcal{O}(\log R)$.

- Í hverju skrefi þurfum við bara að taka $\mathcal{O}(\log R)$.
- Svo tímaflækjan er $\mathcal{O}(\qquad)$ fyrir hverja fyrirspurn.

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

- 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 $\mathcal{O}(\qquad)$ tíma.

- 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.

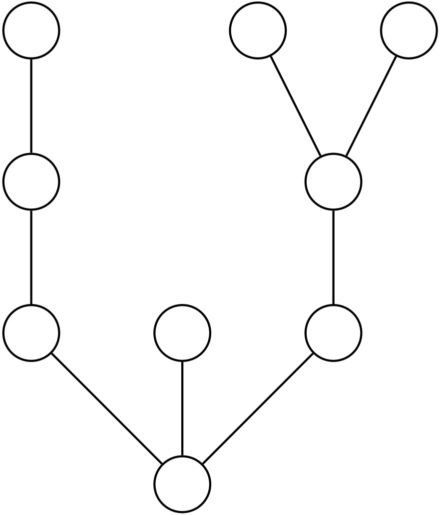
- Skoðum nú aðferð sem notar minna minni, en hefur sömu tímaflækju.

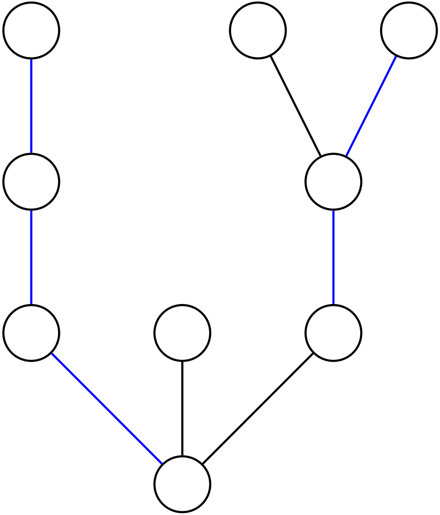
- Skoðum nú aðferð sem notar minna minni, en hefur sömu tímaflækju.
- Hún byggir á að skipta trénu upp í sundurlæga vegi.

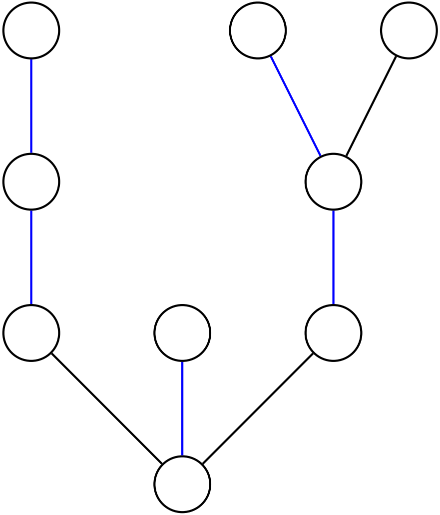
- Skoðum nú aðferð sem notar minna minni, en hefur sömu tímaflækju.
- Hún byggir á að skipta trénu upp í sundurlæga vegi.
- Úthlutum hverjum hnúti sem er ekki lauf nákvæmlega eitt barnið sitt.

- Skoðum nú aðferð sem notar minna minni, en hefur sömu tímaflækju.
- Hún byggir á að skipta trénu upp í sundurlæga vegi.
- Úthlutum hverjum hnúti sem er ekki lauf nákvæmlega eitt barnið sitt.
- Með öðrum orðum veljum við einn legg úr hverjum hnút sem er ekki lauf frá rótinni.

- Skoðum nú aðferð sem notar minna minni, en hefur sömu tímaflækju.
- Hún byggir á að skipta trénu upp í sundurlæga vegi.
- Úthlutum hverjum hnúti sem er ekki lauf nákvæmlega eitt barnið sitt.
- Með öðrum orðum veljum við einn legg úr hverjum hnút sem er ekki lauf frá rótinni.
- Tréð okkar þáttast þá í vegi af völdum leggjum.







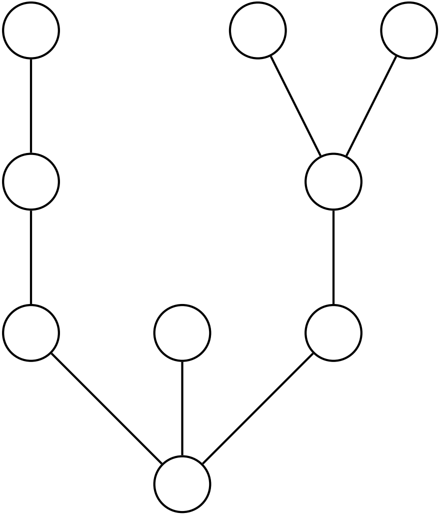
- Spurningin er: Hvernig getum við valið leggina til að þetta hjálpi okkur?

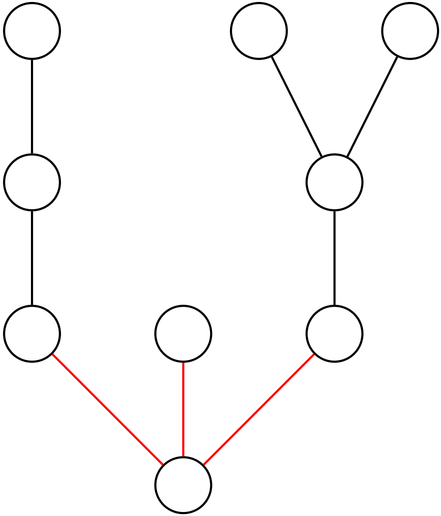
- Spurningin er: Hvernig getum við valið leggina til að þetta hjálpi okkur?
- Fyrir hvern hnút veljum við legginn sem fer í það undirtré sem er stærst.

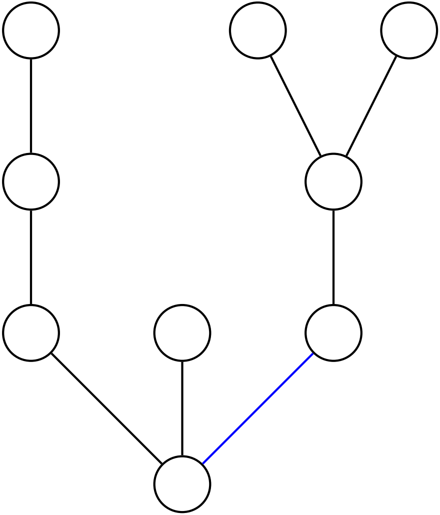
- Spurningin er: Hvernig getum við valið leggina til að þetta hjálpi okkur?
- Fyrir hvern hnút veljum við legginn sem fer í það undirtré sem er stærst.
- Þá innihalda allir vegir frá rót til laufs í mesta lagi $\log_2 n$ leggi sem eru ekki valdir, þar sem n er fjöldi hnúta í trénu.

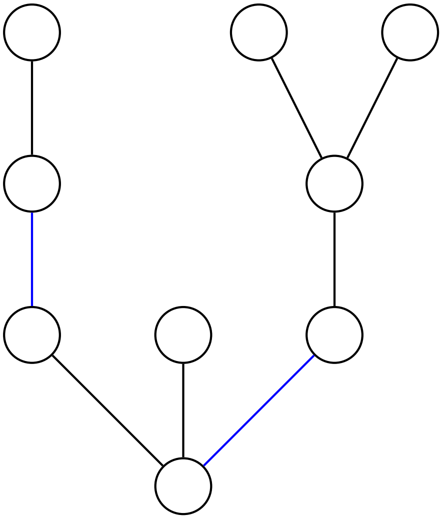
- Spurningin er: Hvernig getum við valið leggina til að þetta hjálpi okkur?
- Fyrir hvern hnút veljum við legginn sem fer í það undirtré sem er stærst.
- Þá innihalda allir vegir frá rót til laufs í mesta lagi $\log_2 n$ leggi sem eru ekki valdir, þar sem n er fjöldi hnúta í trénu.
- Þetta fæst því ef við erum í tré með n hnúta og ferðumst eftir óvöldum legg þá endum við í tréi með $m \leq n/2$ hnúta.

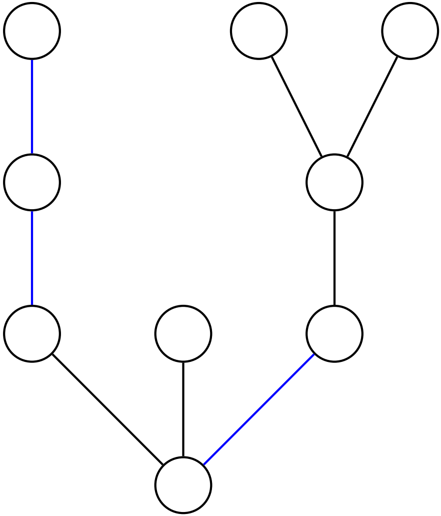
- Spurningin er: Hvernig getum við valið leggina til að þetta hjálpi okkur?
- Fyrir hvern hnút veljum við legginn sem fer í það undirtré sem er stærst.
- Þá innihalda allir vegir frá rót til laufs í mesta lagi $\log_2 n$ leggi sem eru ekki valdir, þar sem n er fjöldi hnúta í trénu.
- Þetta fæst því ef við erum í tré með n hnúta og ferðumst eftir óvöldum legg þá endum við í tréi með $m \leq n/2$ hnúta.
- Við fáum því að ef við byrjum í einhverjum hnút tekur það mest $2 \cdot \log_2 n$ stökk að komast í rótina ef við stökkvum alltaf fremst í vegin af völdum leggjum sem við erum í.

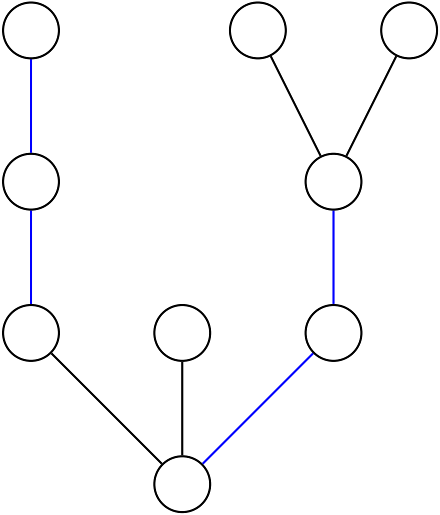


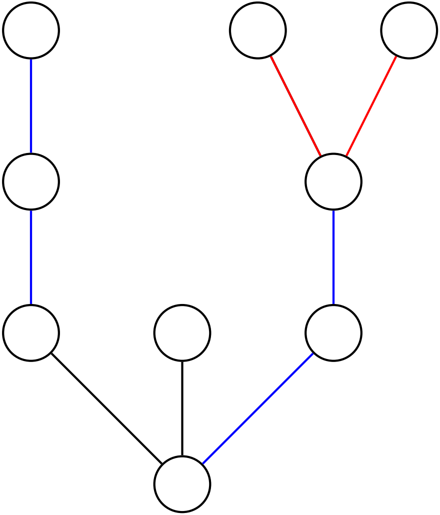


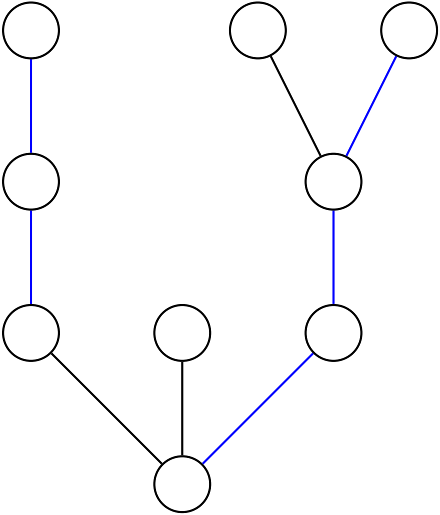


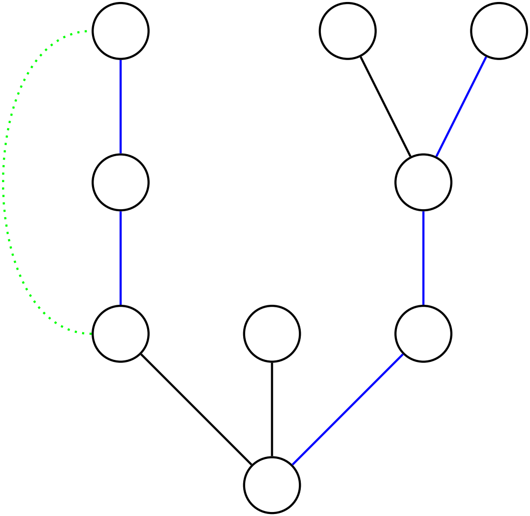


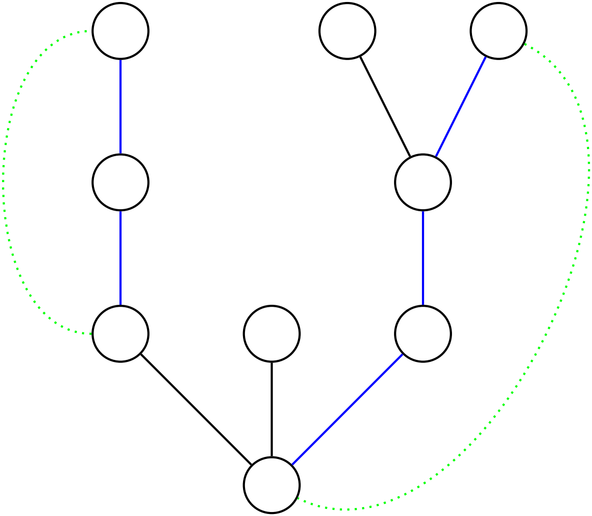


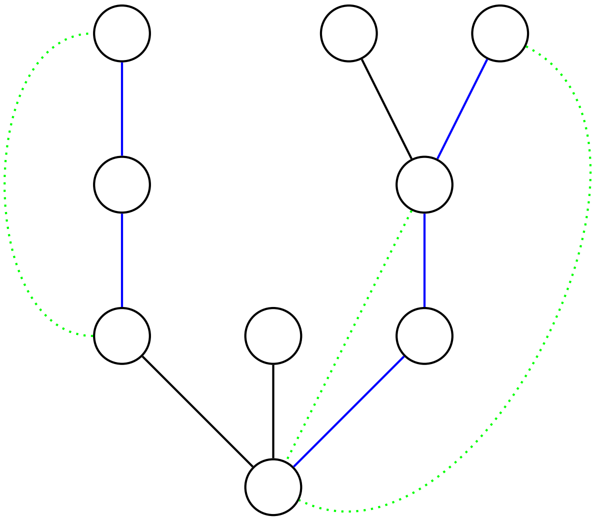












- En hvernig finnum við næsta sameiginlega forföður tveggja hnúta u og v með vegaþáttun.

- En hvernig finnum við næsta sameiginlega forföður tveggja hnúta u og v með vegaþáttun.
- Köllum næsta forföðurinn x.

- En hvernig finnum við næsta sameiginlega forföður tveggja hnúta u og v með vegaþáttun.
- Köllum næsta forföðurinn x.
- Ef u og v eru á sama veg þá er x sá hnútur u og v sem er nær rótinni.

- En hvernig finnum við næsta sameiginlega forföður tveggja hnúta u og v með vegaþáttun.
- Köllum næsta forföðurinn x.
- Ef u og v eru á sama veg þá er x sá hnútur u og v sem er nær rótinni.
- Gerum því ráð fyrir að u og v séu ekki á sama veg og látum h_u vera hnútinn fremst á veginum sem u er á og h_v vera hnútinn sem er fremst á veginum sem v er á.

- En hvernig finnum við næsta sameiginlega forföður tveggja hnúta u og v með vegaþáttun.
- Köllum næsta forföðurinn x.
- Ef u og v eru á sama veg þá er x sá hnútur u og v sem er nær rótinni.
- Gerum því ráð fyrir að u og v séu ekki á sama veg og látum h_u vera hnútinn fremst á veginum sem u er á og h_v vera hnútinn sem er fremst á veginum sem v er á.
- Gerum ráð fyrir að h_u sé lengra frá rótinni en h_v.

- En hvernig finnum við næsta sameiginlega forföður tveggja hnúta u og v með vegaþáttun.
- Köllum næsta forföðurinn x.
- Ef u og v eru á sama veg þá er x sá hnútur u og v sem er nær rótinni.
- Gerum því ráð fyrir að u og v séu ekki á sama veg og látum h_u vera hnútinn fremst á veginum sem u er á og h_v vera hnútinn sem er fremst á veginum sem v er á.
- Gerum ráð fyrir að h_u sé lengra frá rótinni en h_v.
- Þá er x líka sameiginlegur forfaðir h_u og v.

- En hvernig finnum við næsta sameiginlega forföður tveggja hnúta u og v með vegaþáttun.
- Köllum næsta forföðurinn x.
- Ef u og v eru á sama veg þá er x sá hnútur u og v sem er nær rótinni.
- Gerum því ráð fyrir að u og v séu ekki á sama veg og látum h_u vera hnútinn fremst á veginum sem u er á og h_v vera hnútinn sem er fremst á veginum sem v er á.
- Gerum ráð fyrir að h_u sé lengra frá rótinni en h_v.
- Þá er x líka sameiginlegur forfaðir h_u og v.
- Enn fremur er x sameiginlegur forfaðir v og foreldris h_u.

```cpp
int f[MAXN], d[MAXN], p[MAXN];
void hld_init(vvi& g, int r)
{
    int i, j, n = g.size(), k = n, qs = 0, qe = 0, a[n], s[n], h[n];
    for (i = 0; i < n; i++)
        s[i] = 1, f[i] = i, p[i] = 0, d[i] = g[i].size() + (i == r);
    for (i = 0; i < n; i++) if (d[i] == 1) h[qe++] = a[--k] = i;
    for (i = h[qs++]; qs <= qe; i = h[qs++])
        for (j = 0; j < g[i].size(); j++) if (--d[g[i][j]] == 1)
            h[qe++] = g[i][j], a[--k] = g[i][j];
    for (i = 0; i < n; i++) d[i] = (i == r ? 0 : -1), h[i] = -1;
    for (i = 0; i < n; i++) for (j = 0; j < g[a[i]].size(); j++)
        if (d[g[a[i]][j]] == -1) d[g[a[i]][j]] = d[a[i]] + 1;
    for (i = 0; i < n; i++) for (j = 0; j < g[i].size(); j++)
        if (d[i] == d[g[i][j]] + 1) p[i] = g[i][j];
    for (i = n - 1; i >= 0; i--) if (i != 0) s[p[a[i]]] += s[a[i]];
    for (i = 0; i < n; i++) if (i != r) h[p[i]] = i;
    for (i = 0; i < n; i++) if (i != r)
        h[p[i]] = s[h[p[i]]] < s[i] ? i : h[p[i]];
    for (i = 0; i < n; i++) if (h[a[i]] != -1) f[h[a[i]]] = f[a[i]];
}

int hld_lca(int u, int v)
{
    while (f[u] != f[v]) (d[f[u]] > d[f[v]]) ? (u = p[f[u]]) : (v = p[f[v]]);
    return d[u] < d[v] ? u : v;
}
```

- Í hverju skrefi ferðumst við eftir einum ómerktum legg í áttina að rótinni.

- Í hverju skrefi ferðumst við eftir einum ómerktum legg í áttina að rótinni.
- Það eru í mesta lagi $\log_2 n$ ómerktir leggir á veginum að rótinni frá hverjum hnút.

- Í hverju skrefi ferðumst við eftir einum ómerktum legg í áttina að rótinni.
- Það eru í mesta lagi $\log_2 n$ ómerktir leggir á veginum að rótinni frá hverjum hnút.
- Svo tímaflækjan er $\mathcal{O}(\qquad)$.

- Í hverju skrefi ferðumst við eftir einum ómerktum legg í áttina að rótinni.
- Það eru í mesta lagi $\log_2 n$ ómerktir leggir á veginum að rótinni frá hverjum hnút.
- Svo tímaflækjan er $\mathcal{O}(\log n)$.

- Í hverju skrefi ferðumst við eftir einum ómerktum legg í áttina að rótinni.
- Það eru í mesta lagi $\log_2 n$ ómerktir leggir á veginum að rótinni frá hverjum hnút.
- Svo tímaflækjan er $\mathcal{O}(\log n)$.
- Einn kostur þessara aðferðar er að minnisflækjan er $\mathcal{O}(\quad)$ í stað þess að vera $\mathcal{O}(\qquad)$.

- Í hverju skrefi ferðumst við eftir einum ómerktum legg í áttina að rótinni.
- Það eru í mesta lagi $\log_2 n$ ómerktir leggir á veginum að rótinni frá hverjum hnút.
- Svo tímaflækjan er $\mathcal{O}(\log n)$.
- Einn kostur þessara aðferðar er að minnisflækjan er $\mathcal{O}(n)$ í stað þess að vera $\mathcal{O}(\quad)$.

- Í hverju skrefi ferðumst við eftir einum ómerktum legg í áttina að rótinni.
- Það eru í mesta lagi $\log_2 n$ ómerktir leggir á veginum að rótinni frá hverjum hnút.
- Svo tímaflækjan er $\mathcal{O}(\log n)$.
- Einn kostur þessara aðferðar er að minnisflækjan er $\mathcal{O}(n)$ í stað þess að vera $\mathcal{O}(n \log n)$.