

Reiknirit Dijkstra (1959)

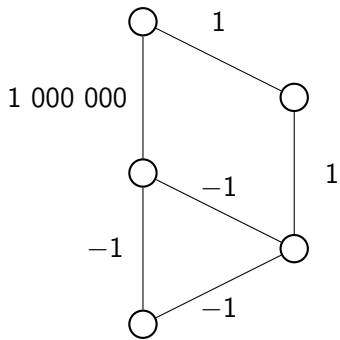
Bergur Snorrason

March 4, 2024

- ▶ Gerum ráð fyrir að við séum með vegið net $G = (V, E, w)$.
- ▶ Látum u_1, u_2, \dots, u_n vera vega í netinu (V, E) .
- ▶ Við segjum þá að *lengdin* á veginum sé

$$\sum_{j=1}^{n-1} w(u_j, u_{j+1}).$$

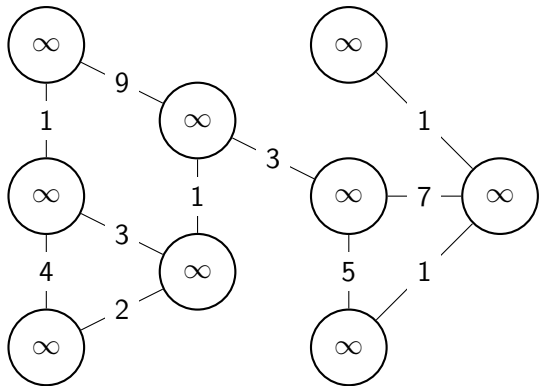
- ▶ Við höfum nú áhuga á að vita hvernig við finnum stysta veg milli tiltekinna hnúta.
- ▶ Tökum þó eftir einu.
- ▶ Þó svo að til sé vegur á milli hnúta þá þarf ekki að vera til stysti vegur.
- ▶ Tökum dæmi.

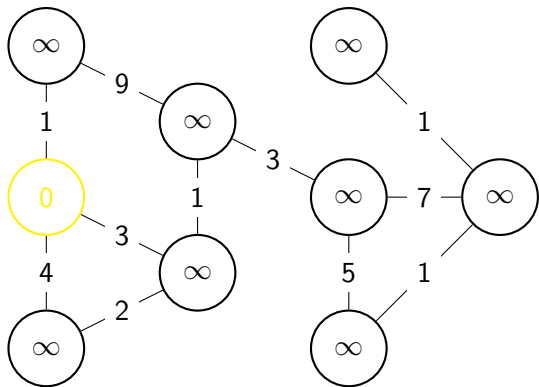


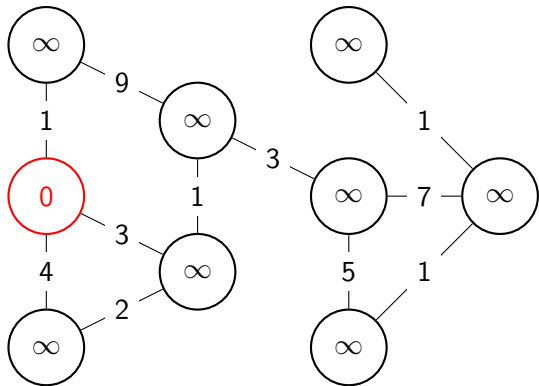
- ▶ Vandamálin myndast þegar við getum gert rásir af neikvæðri lengd.
- ▶ Við munum sníða tímabundið framhjá þessu með því að gera ráð fyrir að $w(e) > 0$ gildi fyrir öll e í E .
- ▶ Algengasta leiðin til að leysa þetta vandamál er með reikniriti Dijkstras.
- ▶ Það er ekki ósvipað breiddarleit.

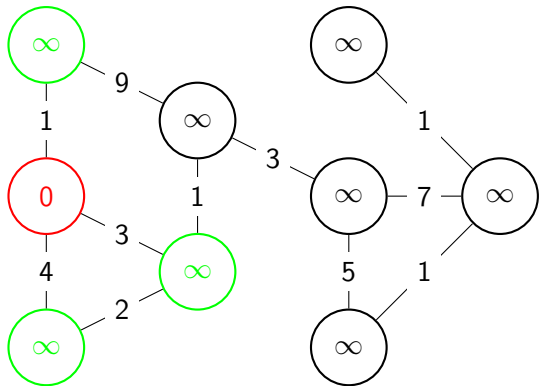
- ▶ Við merkjum alla hnúta „óséða”, nema einn sem við merkjum „séðan”.
- ▶ Sá hnútur er kallaður *upphafsnúturinn*.
- ▶ Við gefum svo öllum hnútum gildi sem upphafstillt er sem ∞ , nema upphafshnúturinn hefur gildi 0.
- ▶ Þetta gildi svara í raun til stysta vegur sem við höfum fundið hingað til.
- ▶ Við endurtökum svo eftirfarandi skref þangað til engir hnútar eru „séðir”:
 - ▶ Tökum þann „séða” hnút u sem hefur minnsta gildi.
 - ▶ Táknum gildi u með g .
 - ▶ Fyrir alla leggi af gerðinni $e_v = (u, v)$ þá uppfærum við gildið hjá v ef það er stærra en $g + w(e_v)$.
 - ▶ Þetta þýðir í raun að til sé styttri vegur til v í gegnum u .
 - ▶ Síðan merkjum við u sem „kláraðann”.

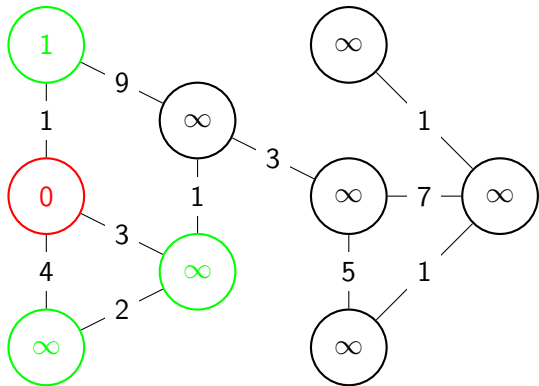
- ▶ Tökum eftir að ef $w(e) = 1$ fyrir alla leggi $e \in E$ þá er þetta breiddarleit.
- ▶ Þetta reiknirit er gráðugt og við munum ekki sanna að það skili alltaf réttum gildum.
- ▶ Reikniritið skilar í raun stysta veg frá upphafshnútnum í alla hnúta.

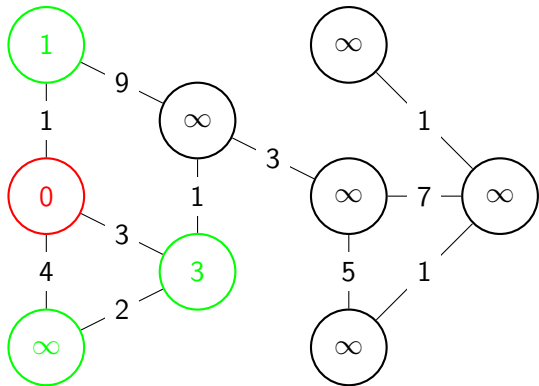


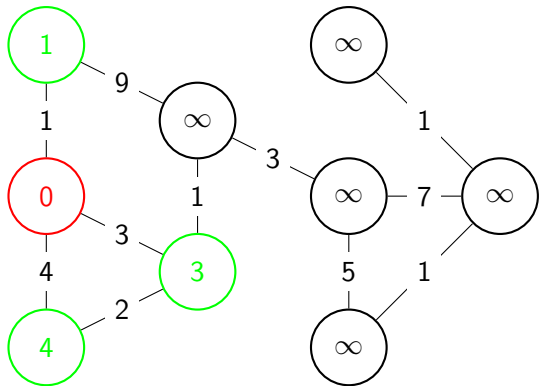


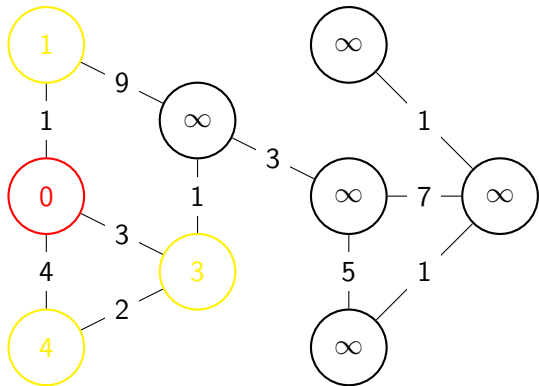


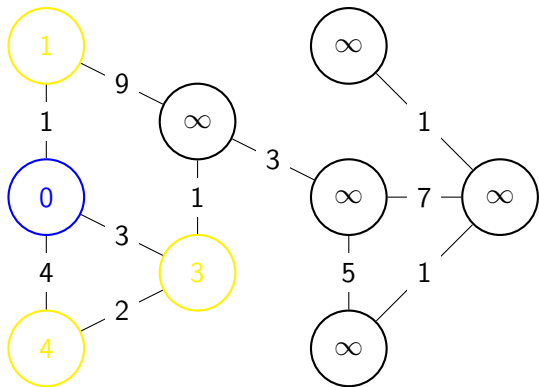


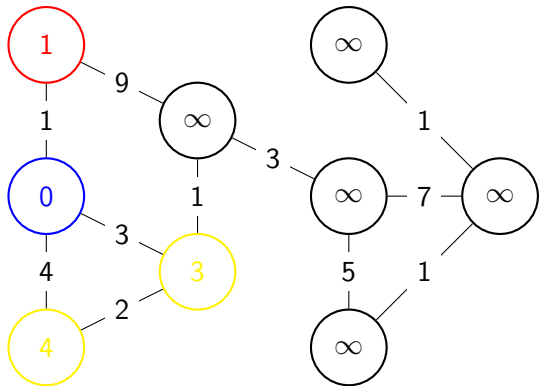


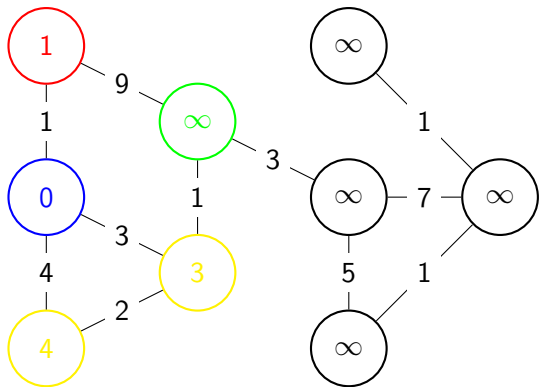


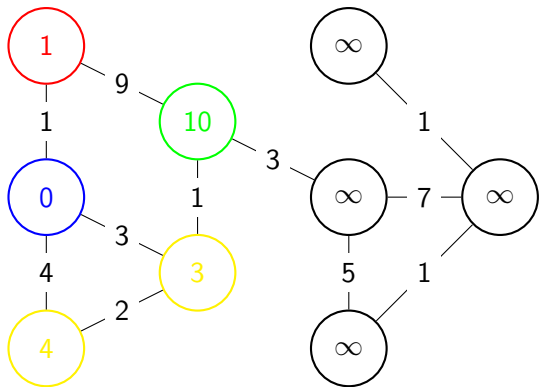


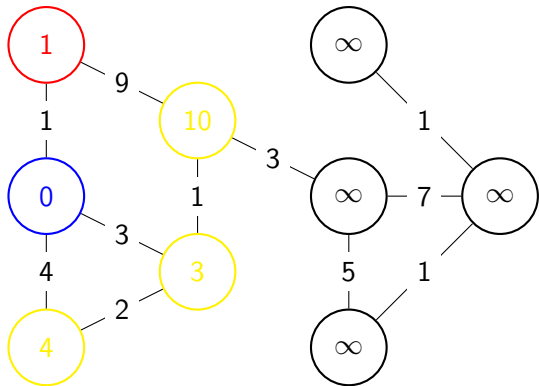


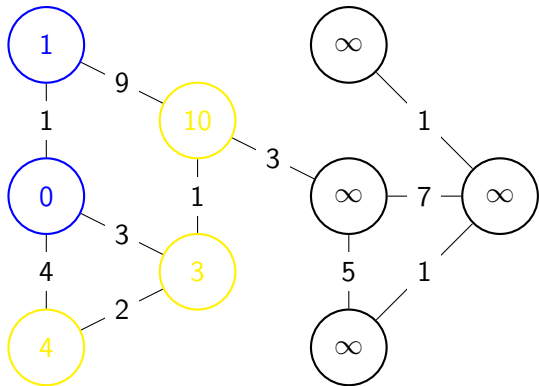


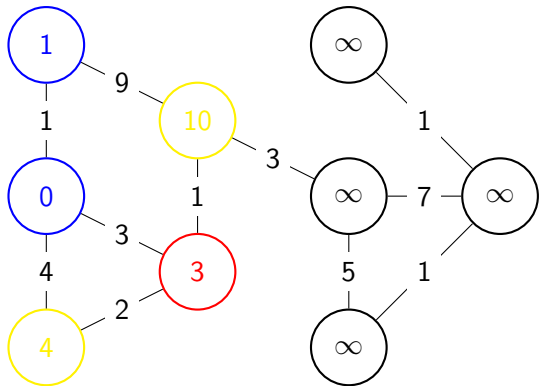


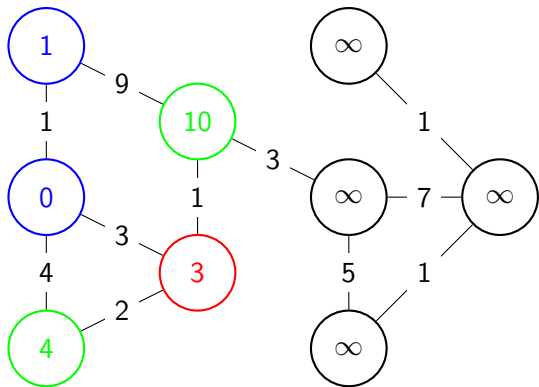


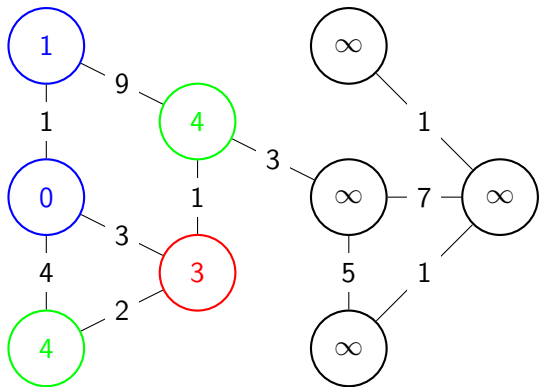


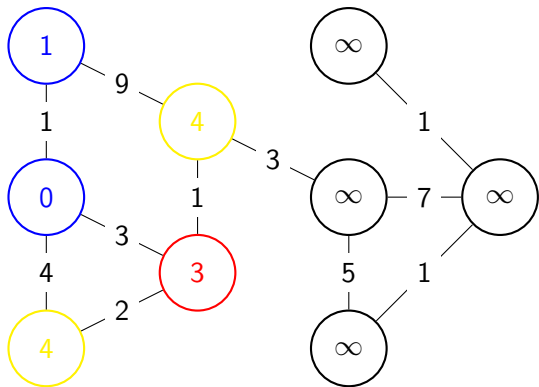


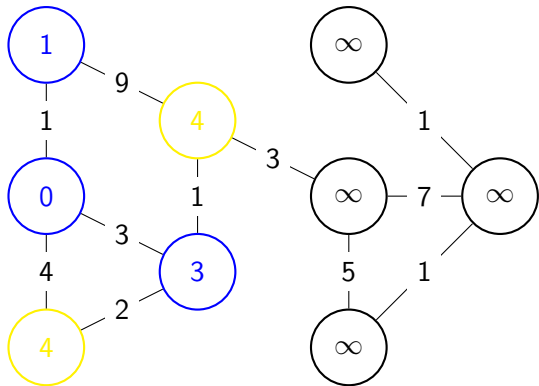


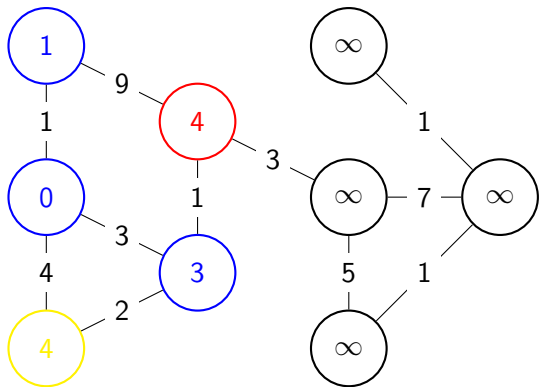


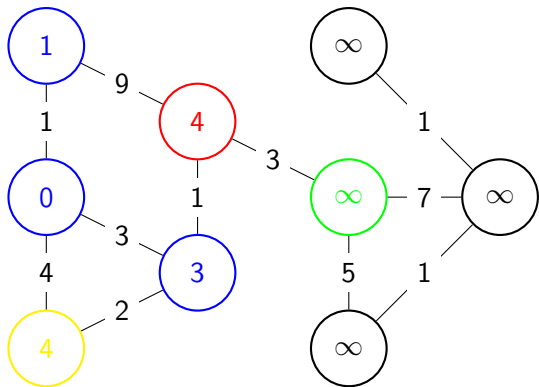


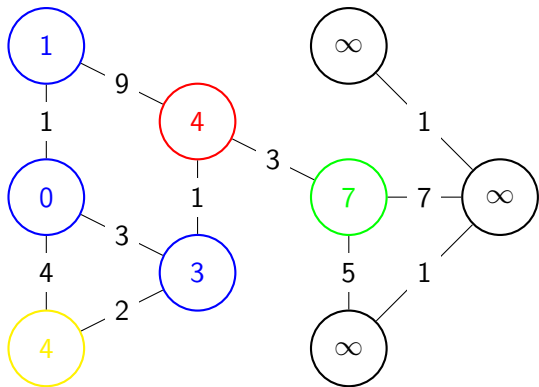


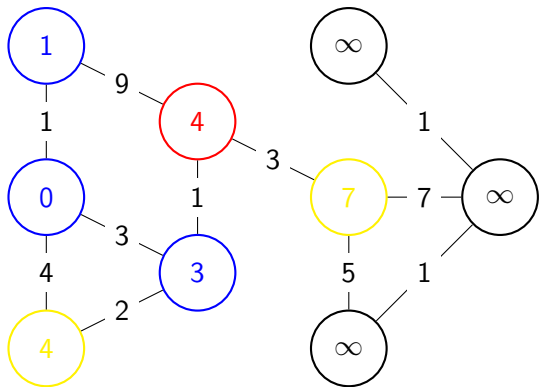


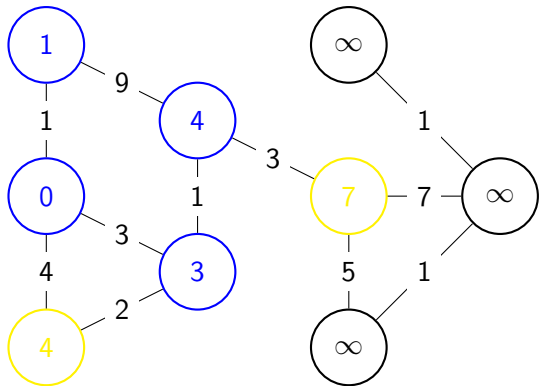


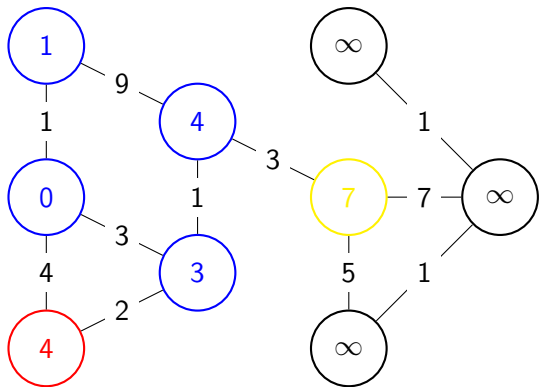


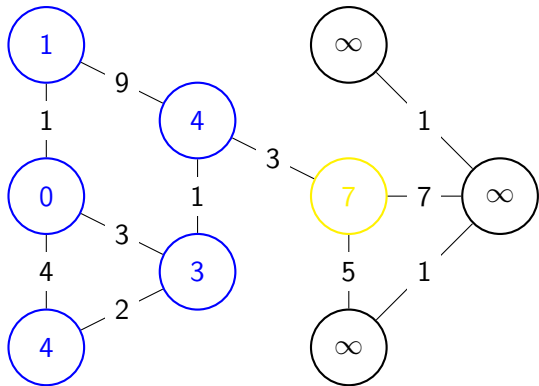


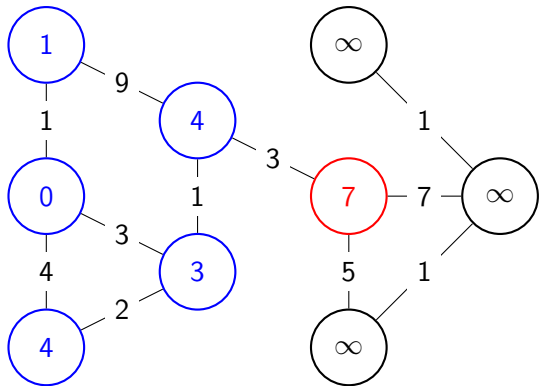


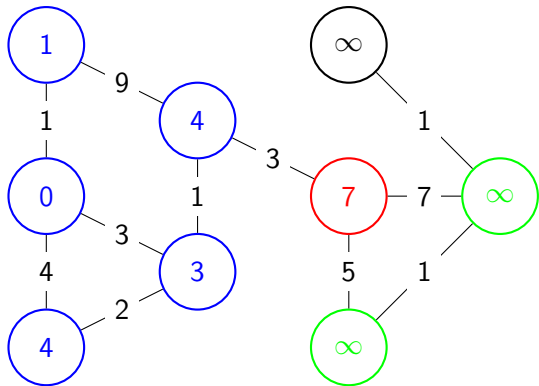


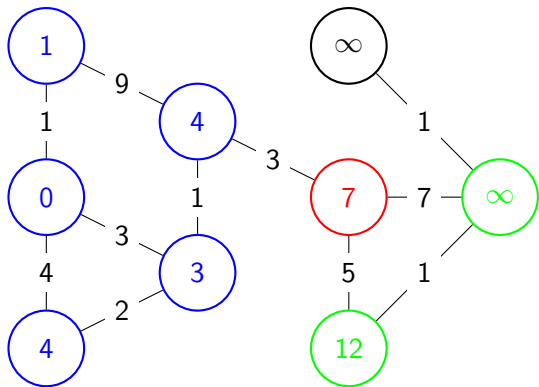


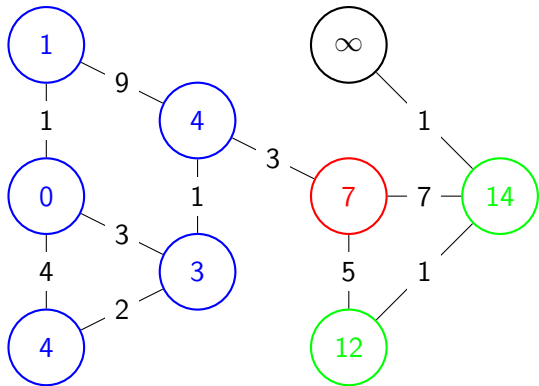


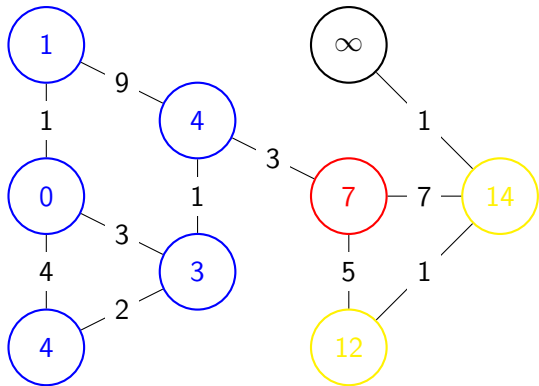


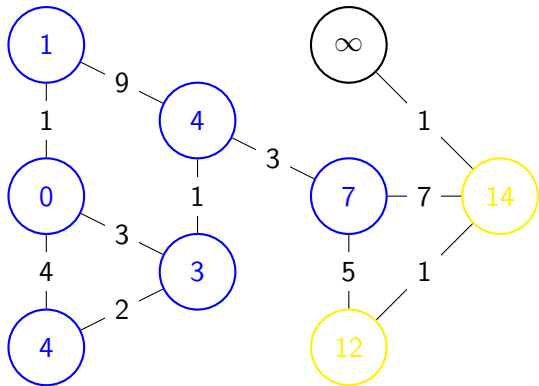


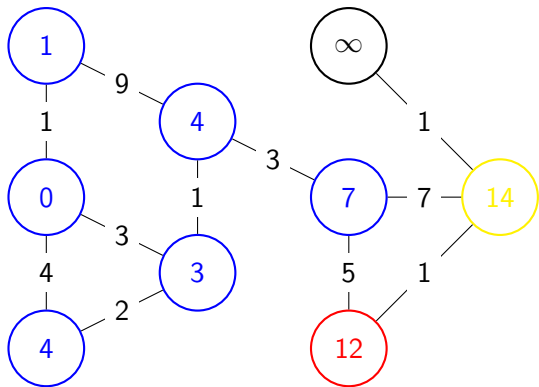


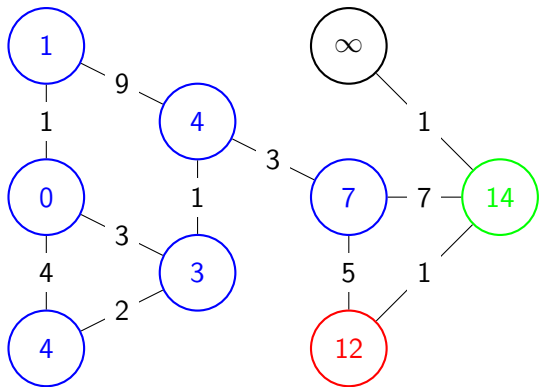


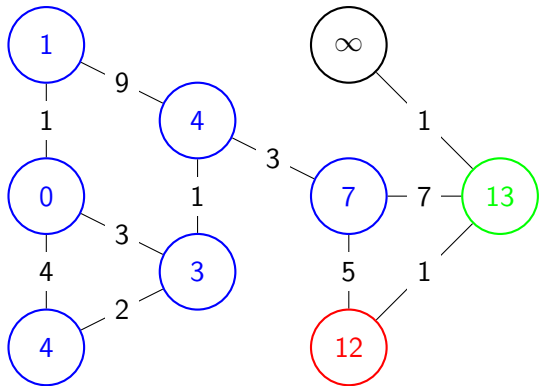


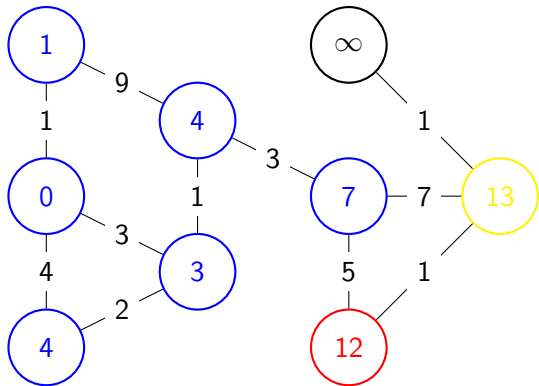


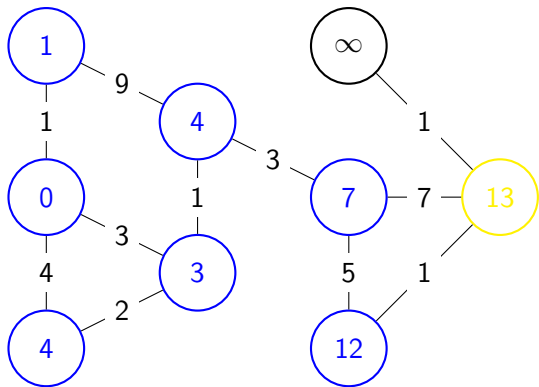


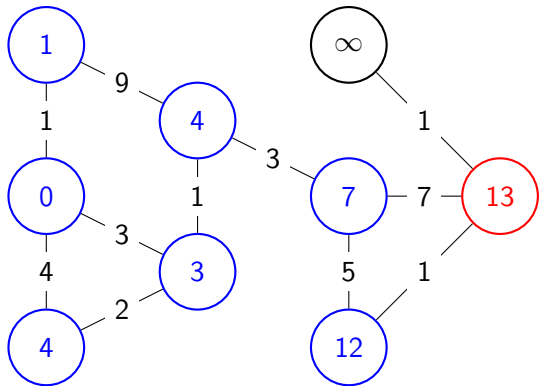


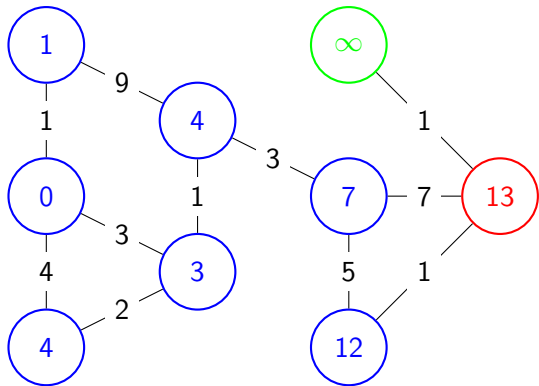


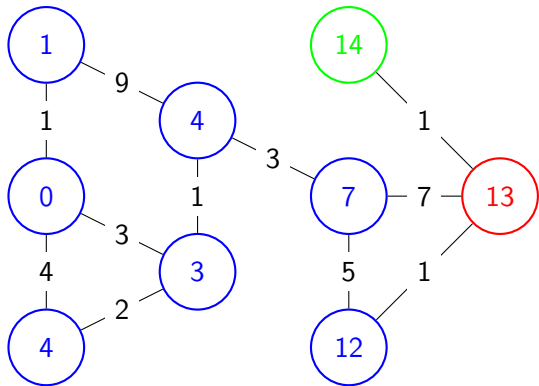


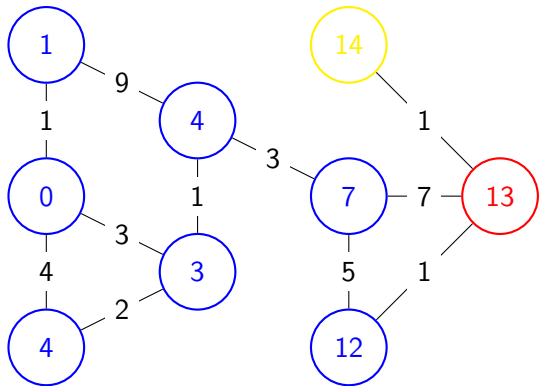


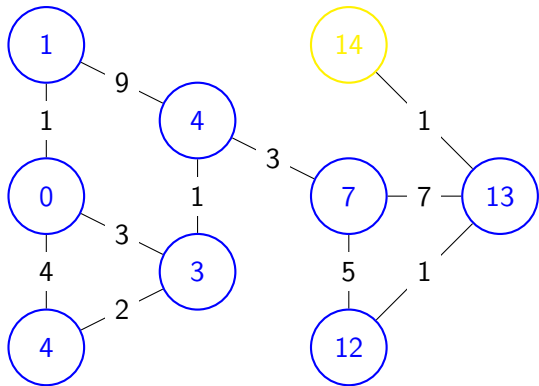


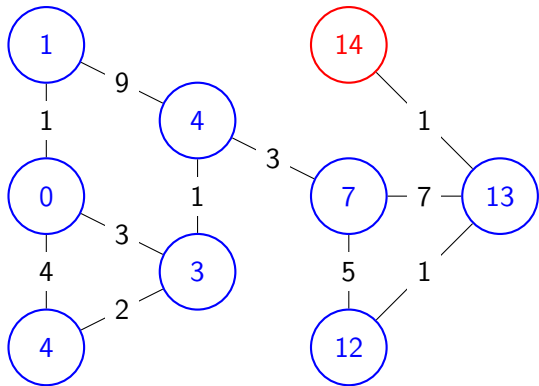


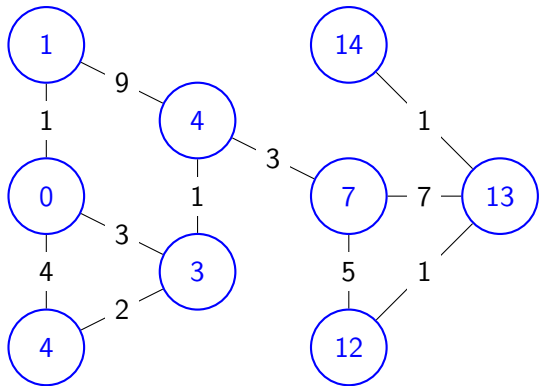












- ▶ Við útfærum þetta líkt og breiddarleit, nema í stað biðraðar notum við forgangsbiðröð.
- ▶ Við þurfum þó að passa okkur á einu.
- ▶ Þegar við uppfærum gildið á hnút v bætum við nýja gildinu í forgangsbiðröð.
- ▶ Gamla gildið er þó ennþá í biðröðinni svo við þurfum að passa okkur á ítra þá ekki í gegnum alla nágranna v aftur.
- ▶ Við gerum þetta með því að bera saman gildið sem er í forgangsbiðröðinni og besta gildið sem við höfum núþegar fundið.
- ▶ Forgangsbiðraðir í C++ skila líka alltaf stærsta gildinu.
- ▶ Við höfum þó áhuga á minnsta gildinu, svo við skiptum um formerki á tölunum sem við látum inn í forgangsbiðröðina.

```

9 vi dijkstra(vvii& g, int s)
10 {
11     int i, x, w, n = g.size();
12     vi d(n, INF);
13     priority_queue<ii> q;
14     q.push(ii(-0, s)), d[s] = 0;
15     while (q.size() > 0)
16     {
17         w = -q.top().first, x = q.top().second, q.pop();
18         if (w > d[x]) continue;
19         for (i = 0; i < g[x].size(); i++)
20         {
21             if (d[g[x][i].first] <= w + g[x][i].second) continue;
22             q.push(ii(-(w + g[x][i].second), g[x][i].first));
23             d[g[x][i].first] = w + g[x][i].second;
24         }
25     }
26     return d;
27 }

```

- ▶ Fyrir hvern legg í netinu gætum við þurft að bæta í forgangsbiðröðina.
- ▶ Við heimsækjum hver hnút að mestu einu sinni.
- ▶ Svo tímaflækjan er $\mathcal{O}((V + E) \log E)$.

