Reiknirit Dijkstras (1959)

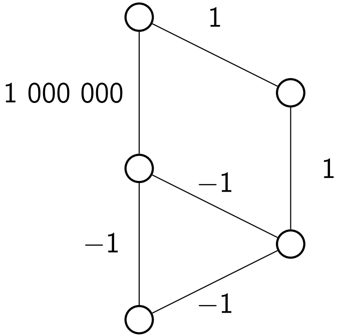
Bergur Snorrason

3. mars 2021

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

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

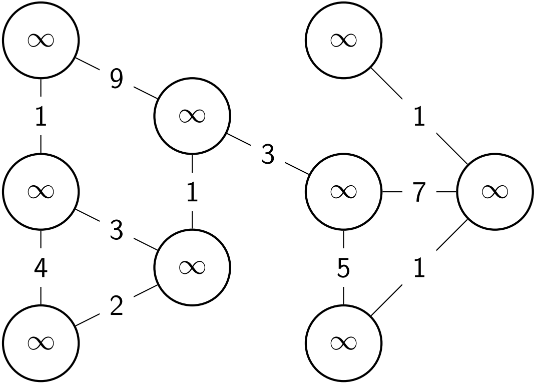
- Við höfum nú áhuga á að vita hvernig við finnum stysta veg milli tiltekinna hnúta.
- Tökum þó eftir einu.
- Þó svo að til sé vegur á milli hnúta þá þarf ekki að vera til stysti vegur.
- Tökum dæmi.

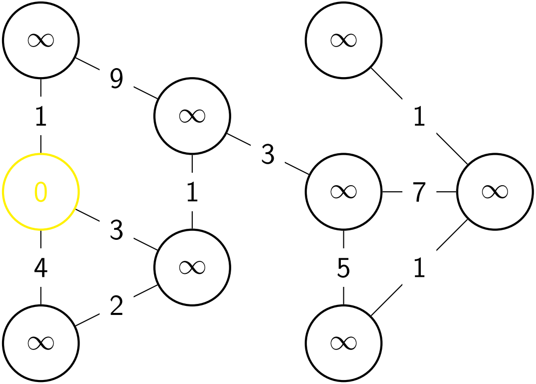

1
1 000 000
1
−1
−1
−1

- 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.

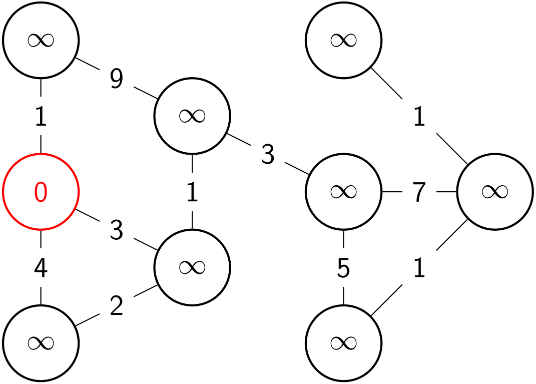
- Við merkjum alla hnút „óserta”, nema einn sem við merkjum „séðan”.
- Sá hnútur er kallaður *upphafsnúturinn*.
- Við gefum svo öllum hnútum gildi sem upphafstillt er sem ∞, nema upphafshnúturin hefur gildi 0.
- Þetta gildi svara í raun til stysta vegur sem við höfum fundið hingað til.
- Við endurtökum 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 geðinni $e_v = (u, v)$ 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 „kláraðan”.

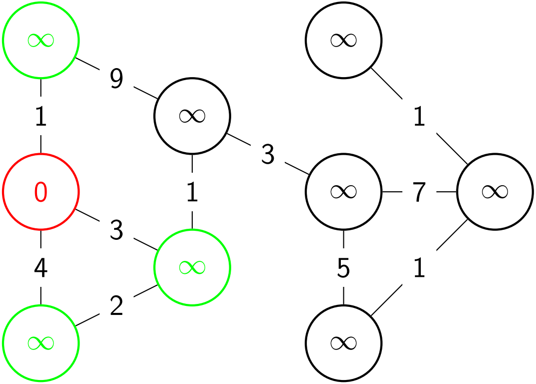
- Tökum eftir að ef $w(e) = 1$ fyrir alla leggi $e \in E$ þá er þetta breiddarleit.
- Þetta reiknirit er gráðugt og við munum ekki sanna að það skili alltaf réttum gildum.
- Reikniritið skilar í raun stysta veg frá öllum hnútum til upphafshnútsins.

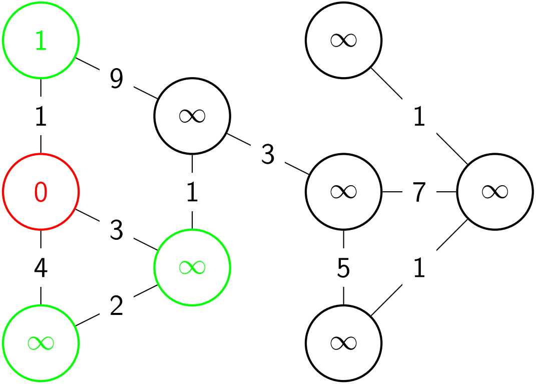

∞
∞
9
1
∞
1
3
∞
1
∞
7
∞
3
4
∞
5
1
2
∞
∞

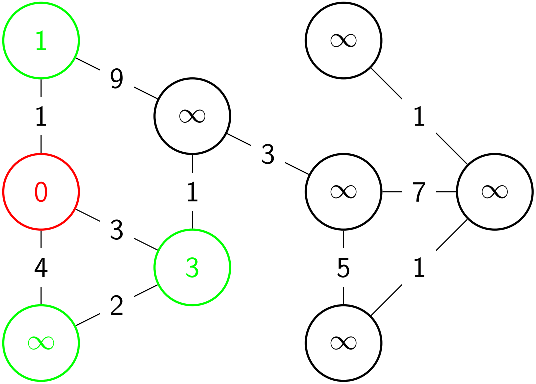


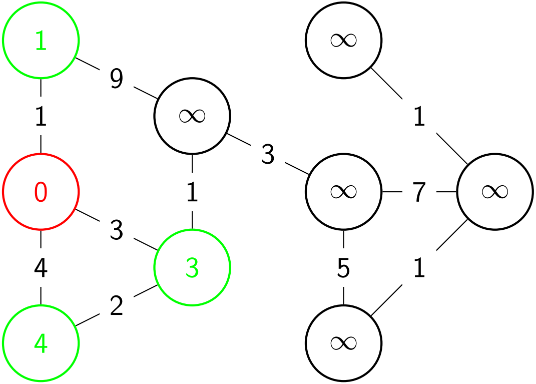
∞
9
1
∞
∞
1
3
0
∞
7
∞
1
3
∞
4
5
1
2
∞
∞

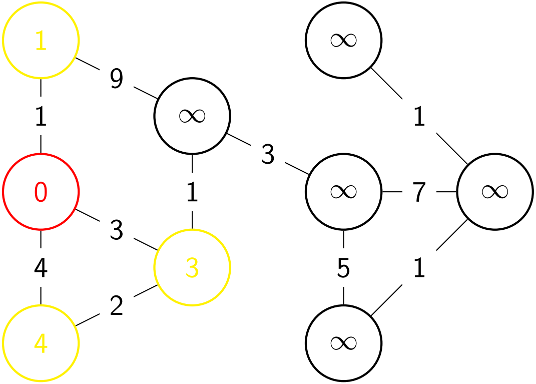

∞
9
1
∞
3
0
1
∞
7
∞
3
∞
4
5
1
2
∞
∞
∞
1
∞

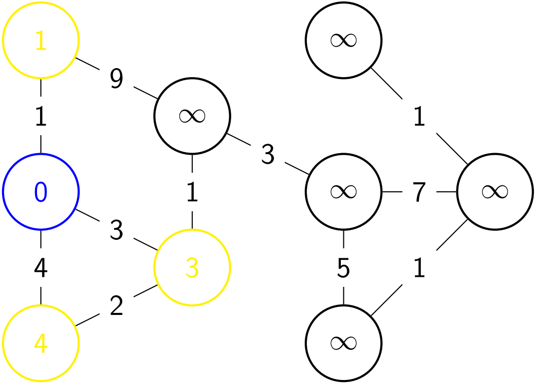

∞
9
1
∞
1
∞
3
0
1
∞
7
∞
3
4
5
1
2
∞
∞
∞

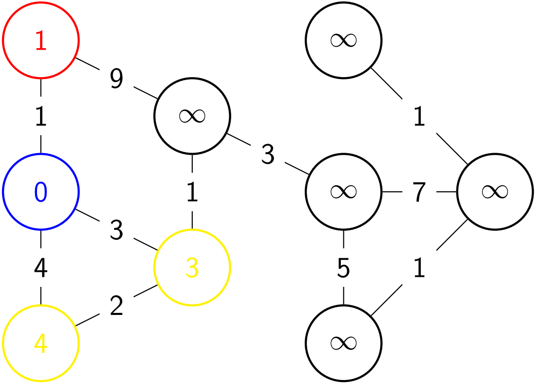

1
9
1
∞
3
0
1
∞
7
∞
3
4
∞
5
1
2
∞
∞
∞
1

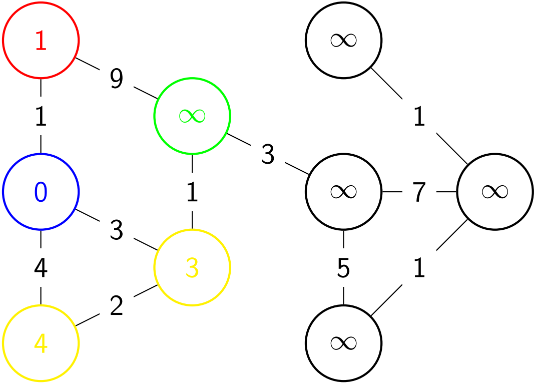

1
9
1
∞
3
0
1
∞
7
∞
3
3
4
5
1
∞
2
∞
∞
∞

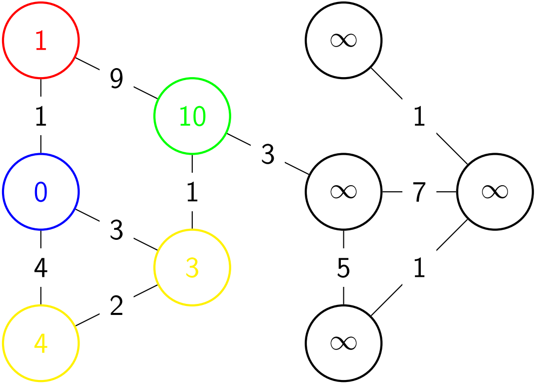

1
0
4
3
∞
∞
∞
∞
∞
9
1
3
4
2
1
3
7
5
1
1

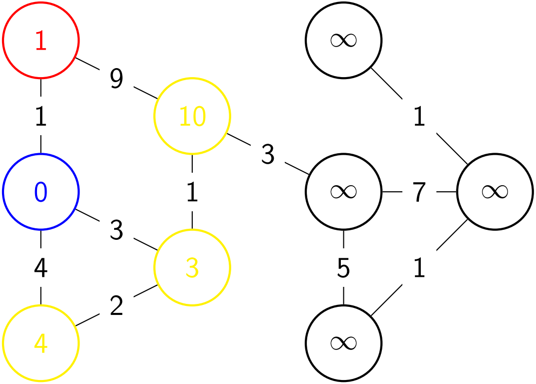

1
9
1
∞
3
0
1
∞
3
4
3
2
4
∞
1
∞
7
∞
5
1
∞

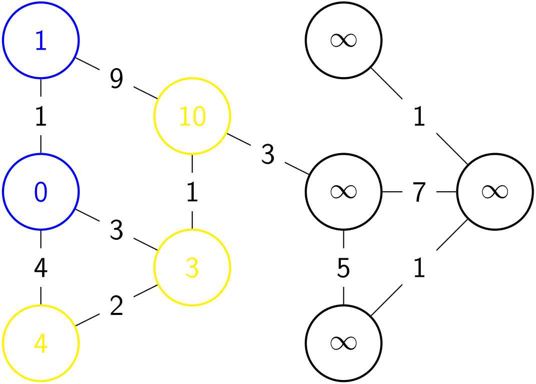

1
9
1
∞
3
0
1
3
∞
7
∞
4
3
2
4
5
1
∞
∞
∞
1

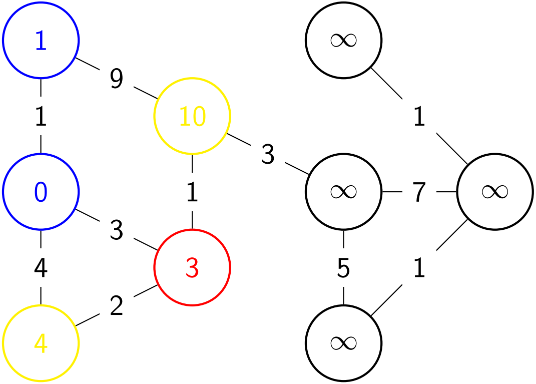

1
9
1
∞
0
3
3
1
3
4
2
4
∞
1
∞
7
∞
5
1
∞

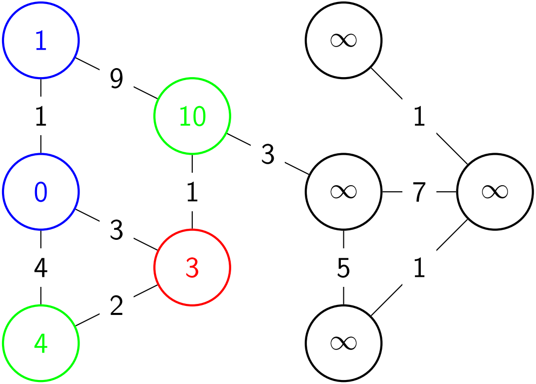

1
9
1
∞
3
0
1
∞
∞
7
∞
3
4
3
5
1
2
4
∞

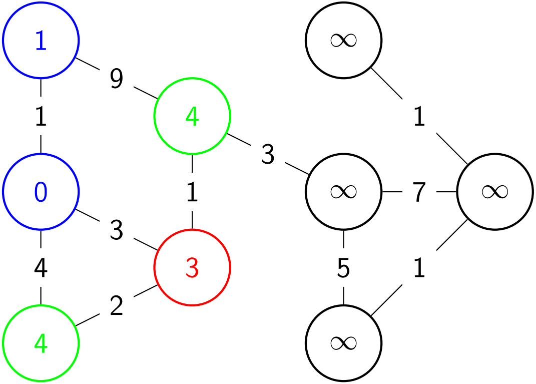

1
9
10
1
0
3
1
3
4
2
4
∞
1
∞
7
∞
5
1
∞

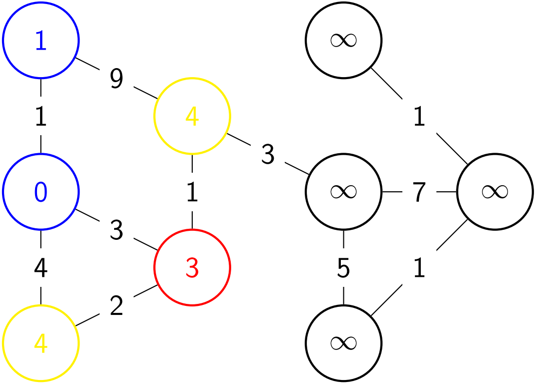

1
9
1
10
3
0
1
3
3
4
4
2
∞
1
∞
7
∞
5
1
∞

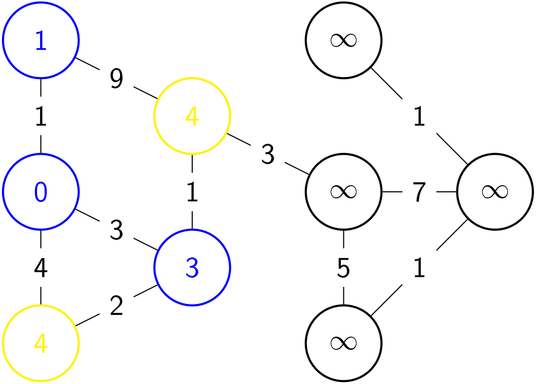

1
9
1
10
3
0
1
3
3
4
2
4
∞
1
∞
7
∞
5
1
∞

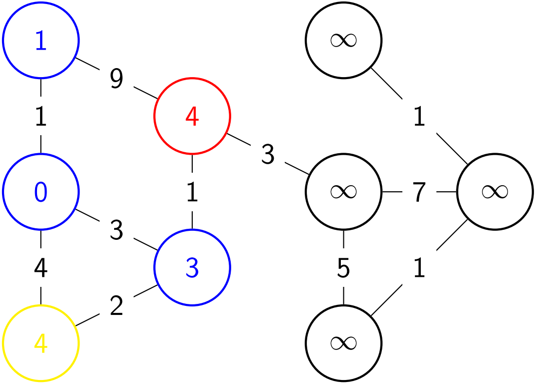

1
9
10
1
0
3
1
3
3
4
4
2
∞
1
∞
7
∞
5
1
∞

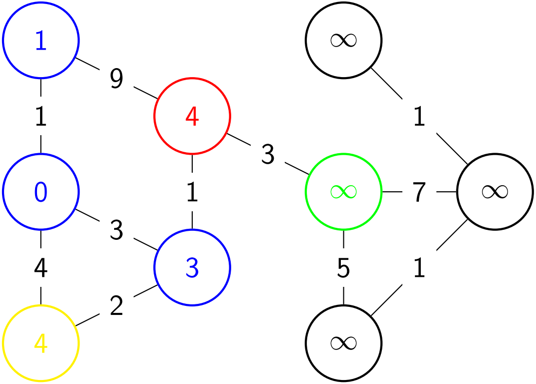

1
9
1
10
3
0
1
3
4
3
2
4
∞
1
∞
7
∞
5
1
∞

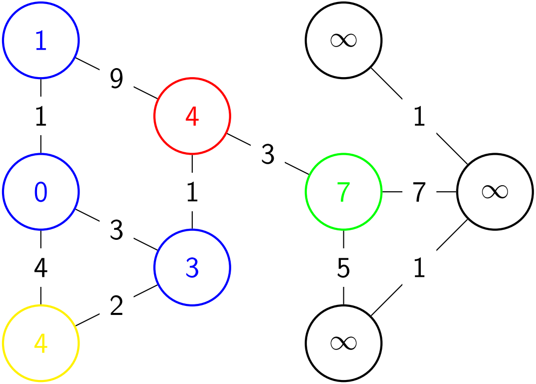

1
9
4
1
3
0
1
3
4
3
2
4
∞
1
∞
7
∞
5
1
∞


1
9
4
1
3
0
1
∞
1
3
∞
7
∞
3
4
5
1
4
2
∞
∞

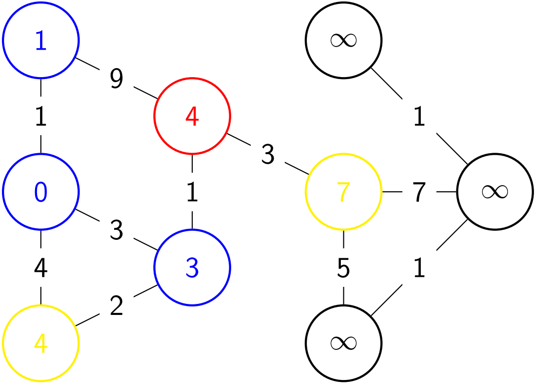

1
9
4
1
3
0
1
∞
3
3
4
2
4
∞
1
∞
7
∞
5
1
∞

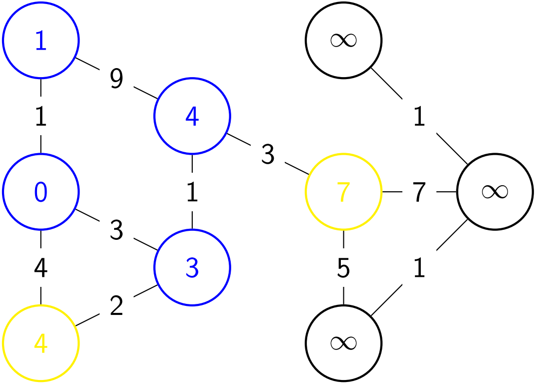

1
9
4
1
3
0
1
7
3
3
4
5
1
4
2
∞
∞
∞
∞
1

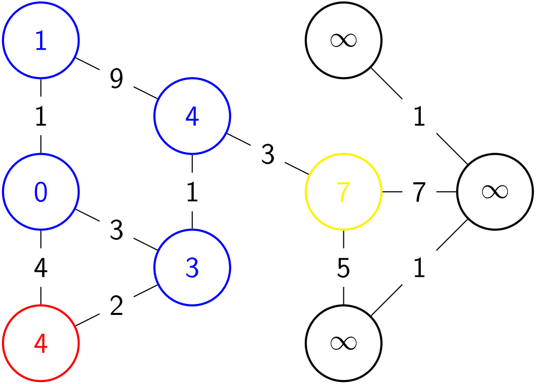

1
9
4
1
∞
1
3
0
1
∞
7
∞
3
4
3
5
1
4
2
∞

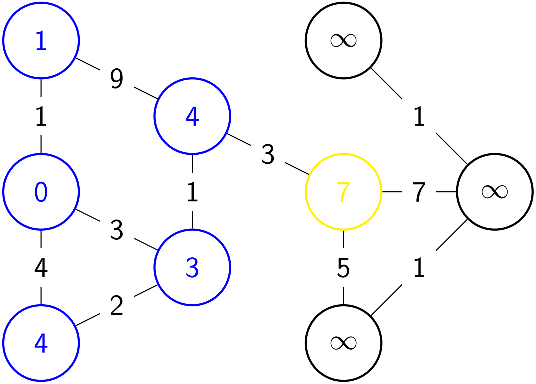


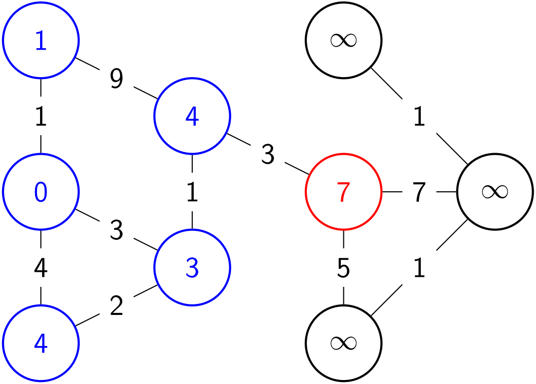
1
9
1
4
3
0
1
3
7
7
∞
∞
∞
3
4
5
1
2
4
1

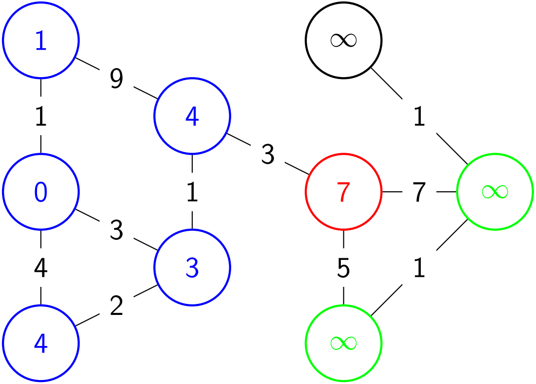

1
9
4
1
0
1
3
3
4
4
2
3
7
∞
1
7
∞
5
1
∞


1
9
4
1
3
7
0
1
3
3
4
2
4
∞
1
∞
7
5
1
∞

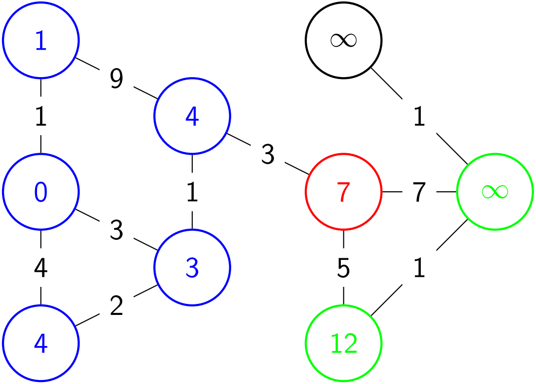

1
9
4
1
∞
1
3
0
1
7
7
∞
3
3
4
5
1
2
4
∞

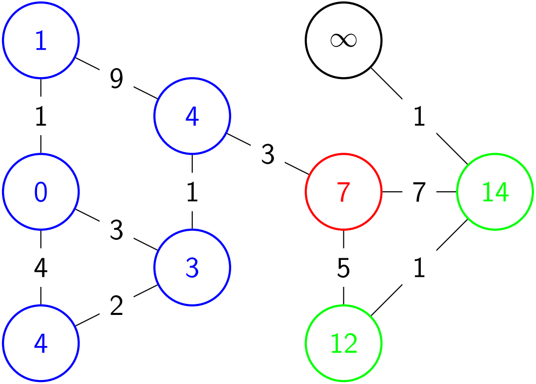

1
9
1
4
3
0
1
3
3
4
2
∞
1
7
7
∞
5
1
∞

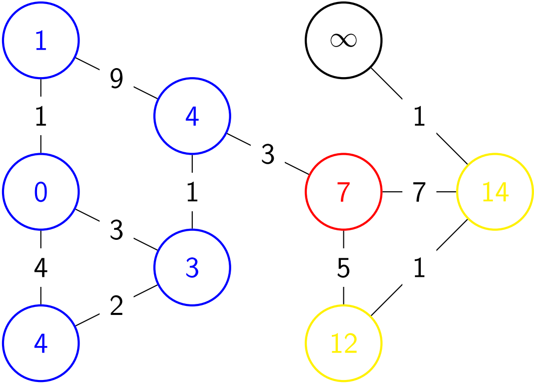

1
9
1
4
3
0
1
7
7
∞
∞
∞
3
3
4
2
5
1
4
1

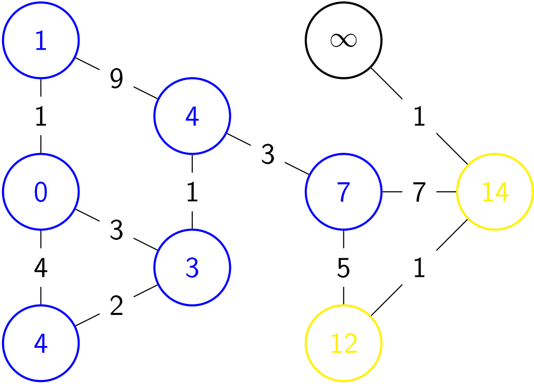


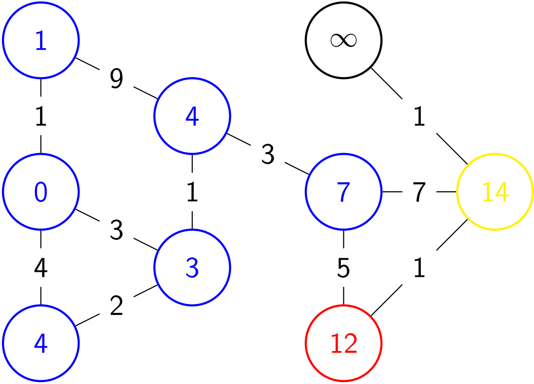
1
9
1
4
∞
1
3
0
1
7
7
∞
3
3
4
5
1
4
2
∞

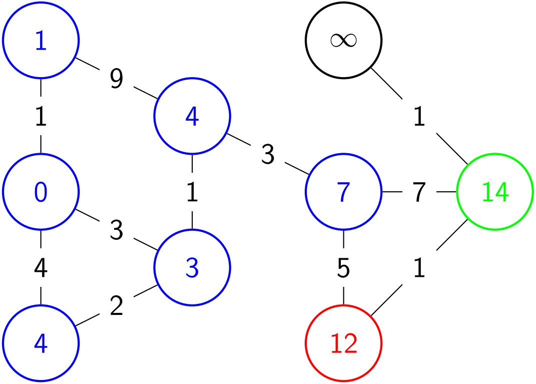

1
∞
9
4
1
1
3
0
1
7
7
∞
3
3
4
5
1
2
4
12


1
9
∞
1
4
1
3
0
1
7
7
14
3
3
4
5
1
2
4
12

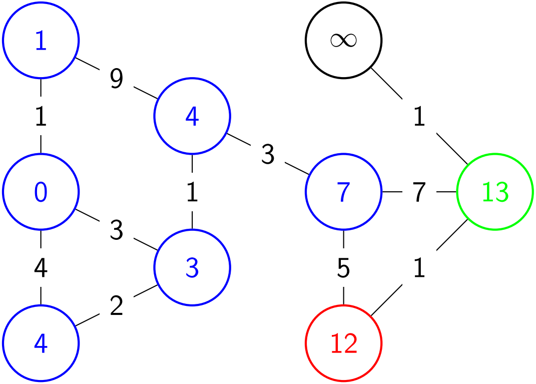

1
9
4
1
∞
1
3
0
1
7
7
14
3
3
4
5
1
2
4
12

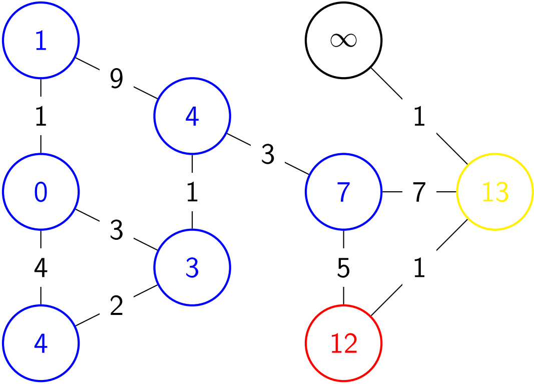

1
9
1
4
3
0
1
7
7
14
3
3
4
5
1
4
2
12
∞
1

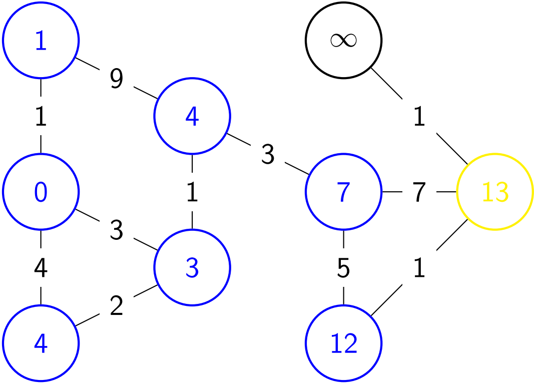

1
9
∞
1
1
4
3
0
1
7
7
14
3
3
4
5
1
2
4
12

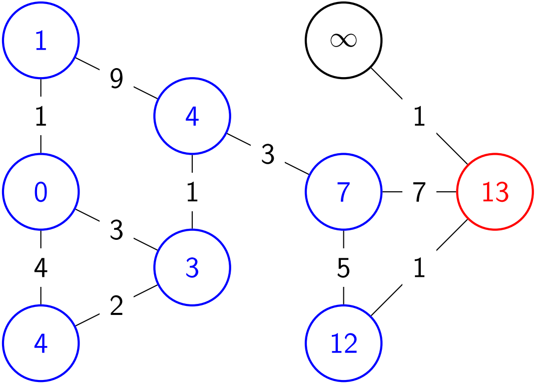


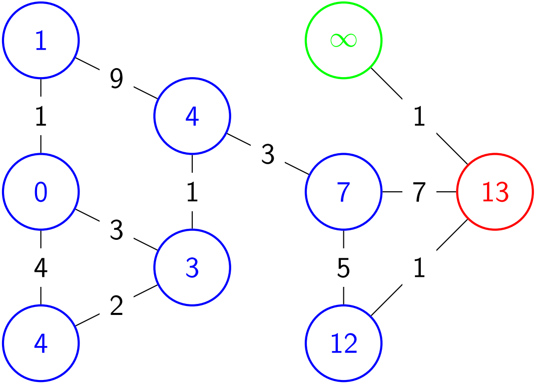
1
9
4
1
∞
1
1
3
0
1
7
7
14
3
3
4
5
1
4
2
12

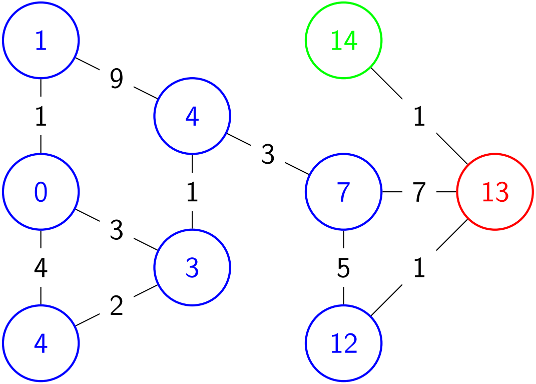

1
9
4
1
3
0
1
3
7
7
13
4
3
3
5
1
4
2
12
∞
1


1
∞
9
4
1
1
3
0
1
7
7
13
3
3
4
5
1
2
4
12

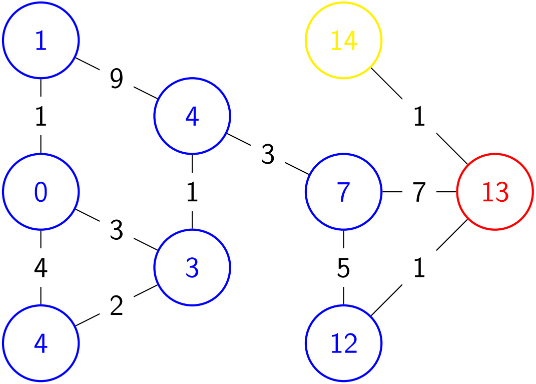

1
9
4
1
0
3
1
3
7
7
13
4
3
5
1
4
2
12
∞
1

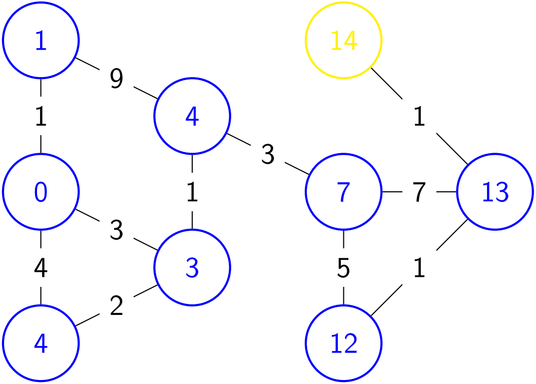

1
9
4
1
3
∞
1
0
1
7
7
13
3
4
3
5
1
4
2
12

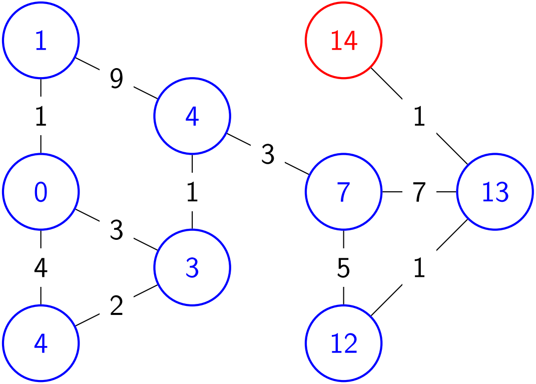

1
∞
9
4
1
1
3
0
1
7
7
13
3
3
4
5
1
2
4
12

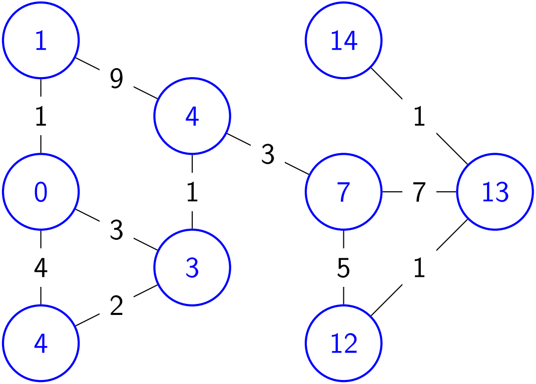


1
9
1
4
14
1
3
0
1
7
7
13
3
4
3
5
1
2
4
12


1
9
1
4
14
1
3
0
7
13
3
1
4
3
5
1
2
4
12
7


1
9
4
1
3
0
7
1
3
4
2
3
4
14
1
7
13
5
1
12


1
14
9
1
4
1
3
0
1
7
7
13
3
4
3
5
1
2
4
12



1
14
9
1
4
1
1
3
0
1
7
7
13
3
3
4
5
1
2
4
12

- 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 a minnsta gildinu, svo við skiptum um formerki á tölunum sem við látum inn í forgangsbiðröðina.

```
vi dijkstra(vvii& g, int s)
{
    int i, x, w, n = g.size();
    vi d(n, INF); // Bestu gildin hingad til
    priority_queue<ii> q; // Geymir tvenndirnar (gildi, hnutur)
    q.push(ii(-0, s)); // Upphafshnutur
    d[s] = 0;
    while (q.size() > 0)
    {
        w = -q.top().first, x = q.top().second; q.pop();
        if (w > d[x]) continue; // thetta er urelt gildi
        rep(i, g[x].size()) if (d[g[x][i].first] > w + g[x][i].second)
        {
            q.push(ii(-(w + g[x][i].second), g[x][i].first));
            d[g[x][i].first] = w + g[x][i].second;
        }
    }
    return d;
}
```

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