Reiknirit Prims (1957)

Bergur Snorrason

1. mars 2021

- Gerum ráð fyrir að við séum með óstefnt vegið samanhangandi net $G = (V, E, w)$.

- Gerum ráð fyrir að við séum með óstefnt vegið samanhangandi net $G = (V, E, w)$.
- Ef við viljum finna spannandi tré nægir að framkvæma leit í trénu (til dæmis breiddarleit eða dýptarleit).

- Gerum ráð fyrir að við séum með óstefnt vegið samanhangandi net $G = (V, E, w)$.
- Ef við viljum finna spannandi tré nægir að framkvæma leit í trénu (til dæmis breiddarleit eða dýptarleit).
- Þar sem við komum aðeins við í hverjum hnút einu sinni ferðumst við aðeins eftir $|V| - 1$ legg.

- Gerum ráð fyrir að við séum með óstefnt vegið samanhangandi net $G = (V, E, w)$.
- Ef við viljum finna spannandi tré nægir að framkvæma leit í trénu (til dæmis breiddarleit eða dýptarleit).
- Þar sem við komum aðeins við í hverjum hnút einu sinni ferðumst við aðeins eftir $|V| - 1$ legg.
- Ef við viljum slembið spannandi tré getum við skipt biðröðinni í breiddarleit út fyrir einhverja gagnagrind sem skilar alltaf slembnu staki.

- Við getum líka beitt aðferð svipaðri reikniriti Dijkstras til að finna minnsta spannandi tré með því að ferðast í netinu.

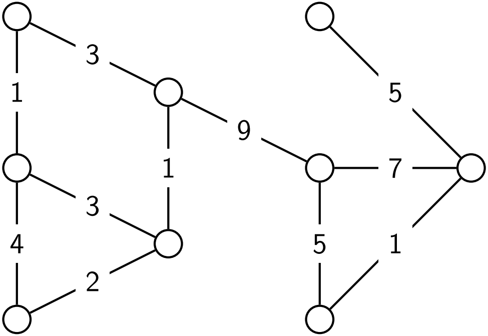
- Við getum líka beitt aðferð svipaðri reikniriti Dijkstras til að finna minnsta spannandi tré með því að ferðast í netinu.
- Við byrjum á að velja upphafshnút og merkjum hann sem „séðan".

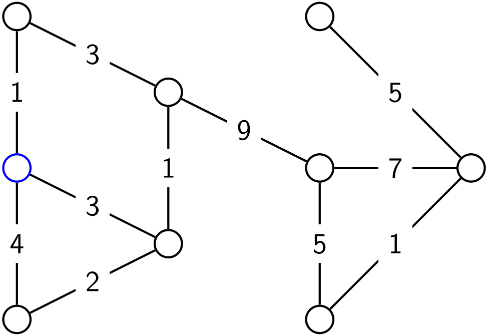
- Við getum líka beitt aðferð svipaðri reikniriti Dijkstras til að finna minnsta spannandi tré með því að ferðast í netinu.
- Við byrjum á að velja upphafshnút og merkjum hann sem „séðan".
- Þar sem allar hnútirnar munu vera í spannandi trénu skiptir ekki máli hvaða hnút við veljum.

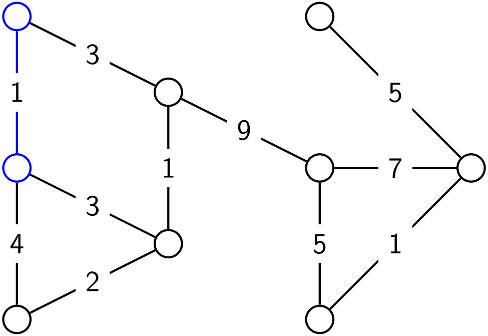
- Við getum líka beitt aðferð svipaðri reikniriti Dijkstras til að finna minnsta spannandi tré með því að ferðast í netinu.
- Við byrjum á að velja upphafshnút og merkjum hann sem „séðan".
- Þar sem allar hnútirnar munu vera í spannandi trénu skiptir ekki máli hvaða hnút við veljum.
- Við ferðumst svo alltaf eftir þeim legg sem hefur minnsta vigt og tengist nákvæmlega einum „séðum" hnút.

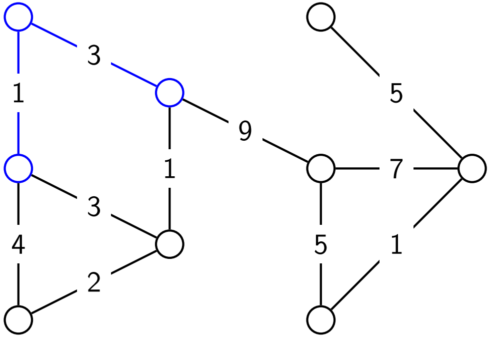
- Við getum líka beitt aðferð svipaðri reikniriti Dijkstras til að finna minnsta spannandi tré með því að ferðast í netinu.
- Við byrjum á að velja upphafshnút og merkjum hann sem „séðan".
- Þar sem allar hnútirnar munu vera í spannandi trénu skiptir ekki máli hvaða hnút við veljum.
- Við ferðumst svo alltaf eftir þeim legg sem hefur minnsta vigt og tengist nákvæmlega einum „séðum" hnút.
- Við merkjum svo hnútinn sem við ferðuðumst í sem „séðan".

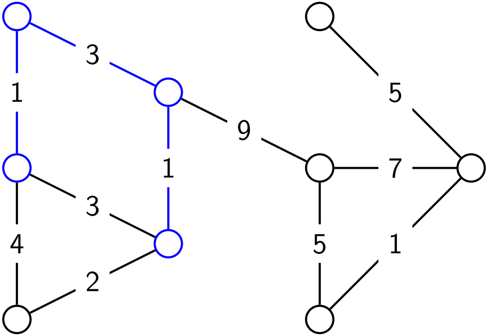
- Við getum líka beitt aðferð svipaðri reikniriti Dijkstras til að finna minnsta spannandi tré með því að ferðast í netinu.
- Við byrjum á að velja upphafshnút og merkjum hann sem „séðan".
- Þar sem allar hnútirnar munu vera í spannandi trénu skiptir ekki máli hvaða hnút við veljum.
- Við ferðumst svo alltaf eftir þeim legg sem hefur minnsta vigt og tengist nákvæmlega einum „séðum" hnút.
- Við merkjum svo hnútinn sem við ferðuðumst í sem „séðan".
- Þetta er gert þangað til allir hnútar eru „séðir".

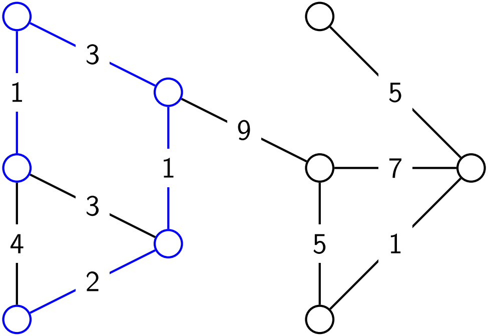

3
1
5
9
1
7
3
4
5
1
2


3
1
9
1
3
4
2
5
7
5
1

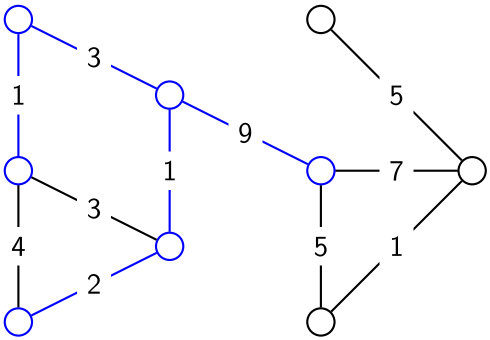

3
1
9
1
3
4
2
5
7
5
1

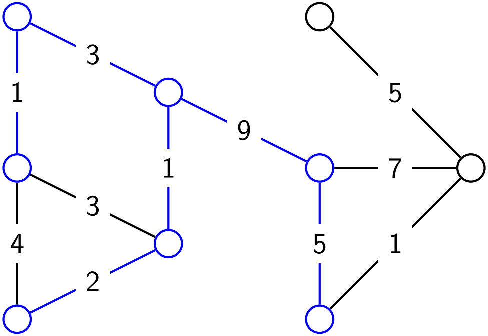

3
1
9
1
3
4
2
5
7
5
1

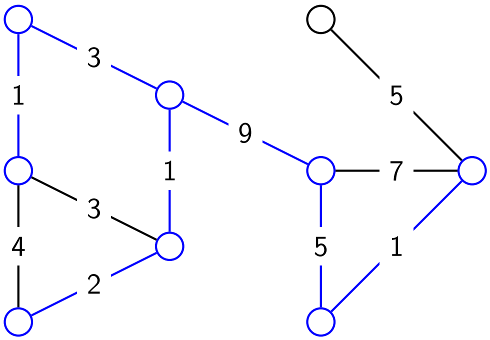

3
1
9
1
5
7
3
4
5
1
2

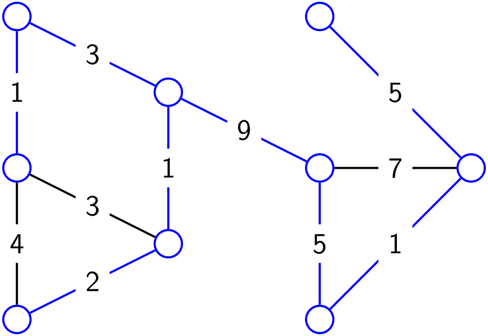


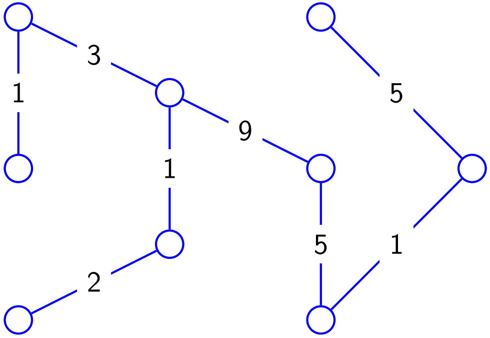
3
1
9
5
1
3
7
4
5
1
2


3
1
9
1
5
7
3
4
5
1
2


3
1
9
1
7
5
3
4
2
5
1


3
1
9
5
1
7
3
4
5
1
2


3
1
9
1
3
4
2
5
7
5
1


3
1
5
9
1
5
1
2

```
int prim(vvii& g, vii& mst)
{
    int i, x, y, w, n = g.size(), r = 0;
    vi v(n);
    mst = vii();
    priority_queue<iii> q;
    q.push(iii(-0, ii(0, -1)));
    while (q.size() > 0)
    {
        iii p = q.top(); q.pop();
        w = -p.first, x = p.second.first, y = p.second.second;
        if (v[x] == 1) continue;
        if (y != -1) mst.push_back(ii(x, y));
        r += w;
        v[x] = 1;
        rep(i, g[x].size()) if (v[g[x][i].first] == 0)
            q.push(iii(-g[x][i].second, ii(g[x][i].first, x)));
    }
    return r;
}
```

- Burt séð frá nokkrum smáatriðum er þetta reiknirit að gera það sama og reiknirit Dijkstras.

- Burt séð frá nokkrum smáatriðum er þetta reiknirit að gera það sama og reiknirit Dijkstras.
- Svo tímaflækjan er $\mathcal{O}($ $)$.

- Burt séð frá nokkrum smáatriðum er þetta reiknirit að gera það sama og reiknirit Dijkstras.
- Svo tímaflækjan er $\mathcal{O}((V + E)\log E)$.

- Burt séð frá nokkrum smáatriðum er þetta reiknirit að gera það sama og reiknirit Dijkstras.
- Svo tímaflækjan er $\mathcal{O}((V + E) \log E)$.
- Það er algengt að kenna reiknirit Prims, frekar en reiknirit Kruskals, þar sem það notast við forgangsbiðröð.

- Burt séð frá nokkrum smáatriðum er þetta reiknirit að gera það sama og reiknirit Dijkstras.
- Svo tímaflækjan er $\mathcal{O}((V + E) \log E)$.
- Það er algengt að kenna reiknirit Prims, frekar en reiknirit Kruskals, þar sem það notast við forgangsbiðröð.
- Það þekkja mun fleiri forgangsbiðraðir en sammengisleit.