

# Inngangur að netafræði

Bergur Snorrason

February 26, 2024

- ▶ Tvennd  $(V, E)$ , þar sem  $V$  er endanlegt mengi og  $E \subset V \times V$ , kallast *net* (e. *graph*).
- ▶ Stökin í  $V$  köllum við *hnúta* (e. *vertices*) og stökin í  $E$  köllum við *leggi* (e. *edges*).
- ▶ Ef venslin  $E$  eru samhverf, það er að segja ef

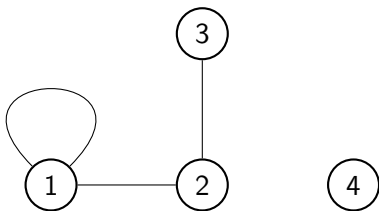
$$(u, v) \in E \Rightarrow (v, u) \in E,$$

þá segjum við að netið sé *óstefnt* (e. *undirected*).

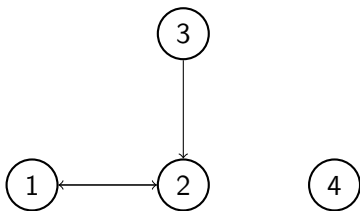
- ▶ Net sem er ekki óstefnt kallast *stefnt* (e. *directed*).
- ▶ Við segjum að hnúturinn  $v$  sé *nágranni* (e. *neighbour*) hnútsins  $u$  ef  $(u, v)$  er í  $E$ .
- ▶ Við segjum að hnútar  $u$  og  $v$  í óstefndu neti séu *nágrannar* ef  $(u, v)$  er í  $E$ .
- ▶ Við segjum einnig að það liggi *leggur* á milli  $u$  og  $v$ .

- ▶ Takið eftir að stefnda netið  $G = (V, E)$  hefur  $|V|$  fjölda hnúta og  $|E|$  fjölda leggja (ögn flóknara ef netið er óstefnt).
- ▶ Tímaflækjur reinkirita í netafræði eru því iðulega gefnar sem föll af  $|V|$  og  $|E|$  (hingað til höfum við aðalega notað  $n$ ).
- ▶ Þegar ég forrita netafræði dæmi læt ég yfirleitt  $n$  tákna fjölda hnúta og  $m$  tákna fjölda leggja.
- ▶ Því kemur fyrir að ég lýsa tímaflækjum reikniritana með þessum breytum.
- ▶ Ég mun einnig leyfa mér að nota  $V$  í stað  $|V|$  og  $E$  í stað  $|E|$ , því þetta ætti aldrei valda ruglingi.
- ▶ Sem dæmi skrifa ég  $\mathcal{O}(E + V)$  í stað  $\mathcal{O}(|E| + |V|)$ .

- ▶ Oft hjálpar að skilja tiltekið net með því að teikna það.
- ▶ Við byrjum á að teikna punkta fyrir hnútana.
- ▶ Ef netið er óstefnt teiknum við svo línu á milli nágranna (svo hver lína svarar til leggs).
- ▶ Ef netið er stefnt þá teiknum við ör í stað línu.
- ▶ Leggur  $(u, v)$  er þá táknaður með ör frá hnút  $u$  til hnúts  $v$ .



- Hér má sjá teikningu sem svarar til  $V = \{1, 2, 3, 4\}$  og  $E = \{(1, 1), (1, 2), (2, 1), (2, 3), (3, 2)\}$ .



- Hér má sjá teikningu sem svarar til  $V = \{1, 2, 3, 4\}$  og  $E = \{(1, 2), (2, 1), (3, 2)\}$ .

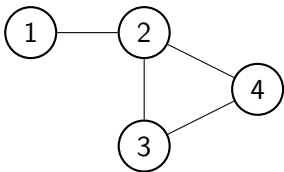
- ▶ Leggir af gerðinni  $(u, u)$  kallast *lykkjur* (e. *loop*) (ástæða nafngiftarinnar sést af fyrri myndinni).
- ▶ Net án lykkja kallast *einfalt* (e. *simple*).
- ▶ Í umfjöllun okkar gerum við ráð fyrir að öll net séu einföld nema annað sé tekið fram.
- ▶ Algenzt eru að skilgreina net á máta sem leyfir fleiri en einn legg milli tveggja tiltekinna hnúta.
- ▶ Einföld net þurfa þá líka að hafa í mesta lagi einn legg milli hnúta.

- ▶ Runa hnúta  $v_1, v_2, \dots, v_n$  kallast *vegur* (e. *path*) ef  $(v_j, v_{j+1}) \in E$ , fyrir  $j = 1, 2, \dots, n - 1$ .
- ▶ Vegurinn kallast *rás* (e. *cycle*) ef  $v_1 = v_n$ .
- ▶ Vegurinn kallast *einfaldur* (e. *simple*) ef engir tveir hnútar í  $v_1, v_2, \dots, v_n$  eru eins.
- ▶ Rás kallast *einföld* (e. *simple*) ef engir tveir hnútar í  $v_1, v_2, \dots, v_{n-1}$  eru eins.
- ▶ Við segjum að vegurinn  $v_1, v_2, \dots, v_n$  *liggi á milli hnútanna*  $v_1$  og  $v_n$ .
- ▶ Óstefnt net er sagt vera *samanhangandi* (e. *connected*) ef til er vegur milli sérhverja tveggja hnúta.
- ▶ Óstefnt net er sagt vera *tré* (e. *tree*) ef það er samanhagandi og inniheldur enga rás.



- ▶ Það er engin stöðluð leið til að geyma net í minni.
- ▶ Yfirleitt er notað eina af þremur gagnagrindum:
  - ▶ Leggjalista.
  - ▶ Nágrannafylki.
  - ▶ Nágrannalista (algengust).

- ▶ Látum  $G = (V, E)$  tákna netið okkar.
- ▶ Þar sem  $V$  er endanlegt megum við gera ráð fyrir að  $V = \{1, 2, \dots, n\}$ , þar sem  $n$  er fjöldi hnúta í  $G$ .
- ▶ Látum  $m$  vera fjölda leggja í  $G$ .
- ▶ Listi af tvenndum sem inniheldur nákvæmlega sömu stök og  $E$  kallast *leggjalisti* netsins  $G$ .
- ▶ Við notum leggjalist sjaldan, en það kemur fyrir (til dæmis í reikniriti Kruskals).
- ▶ Net í dæmum í keppnisforritun eru þó oftast gefin með leggjalista.
- ▶ Í óstefndum netum er hver leggur tvítekinn í  $E$  og við leyfum okkur að sleppa annari endurtekningunni í listanum.

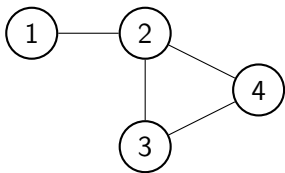

$$L = [$$

(1, 2),  
(2, 3),  
(2, 4),  
(3, 4)

$$]$$

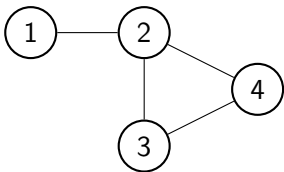
- ▶ Helsti galli leggjalistans er að það tekur  $\mathcal{O}(m)$  tíma að ákvarða hvort hnútar séu nágrannar eða finna nágranna tiltekins hnúts.

- ▶ Látum  $A$  vera  $V \times V$  fylki þannig að  $A_{uv} = 1$  ef  $(u, v)$  er í  $E$ , en  $A_{uv} = 0$  annars.
- ▶ Við köllum  $A$  *nágrannafylki* netsins  $G$ .
- ▶ Takið eftir að það tekur  $\mathcal{O}(V^2)$  tíma að upphafsstillast  $A$ .
- ▶ Svo þessi aðferð er oft of hæg ef, til dæmis ef  $V = 10^5$  (sem er oft raunin).
- ▶ Þegar  $V$  er nógu lítið eru nágrannafylki nytsamleg því við getum ákvarðað hvort tveir hnútar séu nágrannar í  $\mathcal{O}(1)$  tíma.
- ▶ Einnig hefur  $A^p$  (fylkjamargföldun) áhugaverða talningarfræðilega merkingu sem við skoðum síðar.



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

- ▶ Látum nú  $L$  tákna lista af  $n$  listum.
- ▶ Táknum  $j$ -ta lista  $L$  með  $L_j$ .
- ▶ Látum nú  $L_u$  innihalda alla nágranna hnútsins  $u$  í netinu  $G$ , án endurtekningar.
- ▶ Við köllum  $L$  *nágrannalista* (fleirtölu) netsins  $G$  og  $L_u$  *nágrannalista* (eintölu) hnútsins  $u$  í netinu  $G$ .
- ▶ Helsti kostur nágrannalistanna er að við getum skoðað alla nágranna tiltekins hnúts án þess að skoða neitt annað.
- ▶ Við getum því ítrað í gegnum alla nágranna allra hnúta í  $\mathcal{O}(E + V)$  tíma, óháð röð hnútanna.
- ▶ Þetta kemur að góðum notum þegar við erum að ferðast í gegnum netið.


$$L = \begin{bmatrix} [2] \\ [1, 3, 4] \\ [2, 4] \\ [2, 3] \\ ] \end{bmatrix}$$



- ▶ Eins og minnst var á áðan eru net oftast gefin með leggjalista, en við vinnum yfirleitt með nágrannalista.
- ▶ Til að breyta á milli látum nágrannalistann okkar vera af tagi `vector<vector<int>>`.
- ▶ Við upphafsstillum hann með  $n$  tómum listum.
- ▶ Við lesum svo í gegnum alla leggina og bætum viðeigandi hnútum í tilheyrandi nágrannalista.
- ▶ Ef leggur  $(u, v)$  er í leggjalista stefnds nets þá bætum við  $v$  við  $V_u$ .
- ▶ Ef leggur  $(u, v)$  er í leggjalista óstefnds nets þá bætum við  $v$  við  $V_u$  og  $u$  við  $V_v$ .

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef vector<int> vi;
4 typedef vector<vi> vvi;
5
6 int main()
7 {
8     int i, j, n, m, x, y;
9     cin >> n >> m;
10    vvi g(n);
11    for (i = 0; i < m; i++)
12    {
13        cin >> x >> y;
14        x--, y--;
15        g[x].push_back(y);
16        g[y].push_back(x);
17    }
18    for (i = 0; i < n; i++)
19    {
20        printf("%d: ", i + 1);
21        for (j = 0; j < g[i].size(); j++) printf("%d ", g[i][j] + 1);
22        printf("\n");
23    }
24    return 0;
25 }

```

- ▶ Við segjum að þrenndin  $(V, E, w)$ , þar sem  $(V, E)$  er net og  $w: E \rightarrow \mathbb{R} \cup \{-\infty, \infty\}$ , sé *vegið net* (e. *weighted graph*).
- ▶ Það sem við erum í rauninni að gera er að gefa hverjum legg í netinu vigt.
- ▶ Ef við erum, til dæmis, með net sem lýsir gönguleiðum í Mosfellsbæ þá gætu vigtirnar verið lengd hvers leggs gönguleiðanna eða tíminn sem það tekur að labba legginn.
- ▶ Þegar við teiknum vegin net þá teiknum við netið líkt og áður og bætum vigtunum við hliðina á tilheyrandi leggjum.

- ▶ Þegar við geymum vegin net í einhverri gagnagrind þá bætum við yfirleitt bara vigtinni við.
- ▶ Leggjalisti vegins nets er því listi af þrenndum.
- ▶ Þrennd  $(u, v, w)$  þýðir þá að það liggi leggur frá hnút  $u$  til hnúts  $v$  með vigt  $w$ .
- ▶ Nágrannafylki vegins nets er gefið með  $A_{uv} = \infty$  ef enginn leggur liggur á frá hnút  $u$  til hnúts  $v$  og  $A_{uv} = w(e)$  er leggurinn  $e$  liggur frá hnút  $u$  til hnúts  $v$ .
- ▶ Nágrannalistar vegins nets eru listar af tvenndum, fyrra stak tvenndarinnar er nágranninn og seinna stakið er vigtin á leggnum til nágrannans.
- ▶ Nágrannalistarnir eru því geymdir með `vector<vector<pair<int, int>>>`.
- ▶ Við notum `typedef` til að styttu þetta niður í `vvii`.

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef pair<int, int> ii;
4 typedef vector<ii> vii;
5 typedef vector<vii> vvii;
6
7 int main()
8 {
9     int i, j, n, m, x, y, w;
10    cin >> n >> m;
11    vvii g(n);
12    for (i = 0; i < m; i++)
13    {
14        cin >> x >> y >> w;
15        x--, y--;
16        g[x].push_back(ii(y, w));
17        g[y].push_back(ii(x, w));
18    }
19    printf("Nágrannar:\n");
20    for (i = 0; i < n; i++)
21    {
22        printf("%d: ", i + 1);
23        for (j = 0; j < g[i].size(); j++) printf("%2d ", g[i][j].first + 1);
24        printf("\n");
25    }
26    printf("Vigtir:\n");
27    for (i = 0; i < n; i++)
28    {
29        printf("%d: ", i + 1);
30        for (j = 0; j < g[i].size(); j++) printf("%2d ", g[i][j].second);
31        printf("\n");
32    }
33    return 0;
34 }

```

