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*).

- 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*).

- 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*).

- 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*).

- 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.

- 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.

- 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).

- 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).

- 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.

- 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.

- 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.

- 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 $\mathcal{O}(|E| + |V|)$.

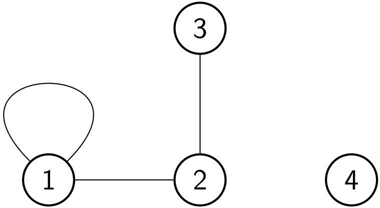
- Oft hjálpar að skilja tiltekið net með því að teikna það.

- Oft hjálpar að skilja tiltekið net með því að teikna það.
- Við byrjum á að teikna punkta fyrir hnútana.

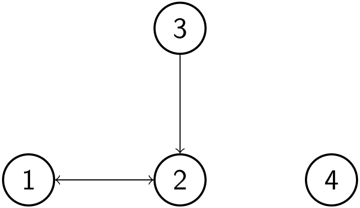
- Oft hjálpar að skilja tiltekið net með því að teikna það.
- Við byrjum á að teikna punkta fyrir hnútana.
- Ef netið er óstefnt teiknum við svo línu á milli nágranna (svo hver lína svarar til leggs).

- 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.

- 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).

- Leggir af gerðinni (u, u) kallast *lykkjur* (e. *loop*) (ástæða nafngiftarinnar sést af fyrri myndinni).
- Net án lykkja kallast *einfalt* (e. *simple*).

- 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.

- 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.
- Algengt eru að skilgreina net á máta sem leyfir fleiri en einn legg milli tveggja tiltekinna hnúta.

- 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.
- Algengt 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 $v_1, v_2, ..., v_n$ kallast *vegur* (e. *path*) ef $(v_j, v_{j+1}) \in E$, fyrir $j = 1, 2, ..., n-1$.

- Runa hnúta $v_1, v_2, ..., v_n$ kallast *vegur* (e. *path*) ef $(v_j, v_{j+1}) \in E$, fyrir $j = 1, 2, ..., n-1$.
- Vegurinn kallast *rás* (e. *cycle*) ef $v_1 = v_n$.

- Runa hnúta $v_1, v_2, ..., v_n$ kallast *vegur* (e. *path*) ef $(v_j, v_{j+1}) \in E$, fyrir $j = 1, 2, ..., n-1$.
- Vegurinn kallast *rás* (e. *cycle*) ef $v_1 = v_n$.
- Vegurinn kallast *einfaldur* (e. *simple*) ef engir tveir hnútar í $v_1, v_2, ..., v_n$ eru eins.

- Runa hnúta $v_1, v_2, ..., v_n$ kallast *vegur* (e. *path*) ef $(v_j, v_{j+1}) \in E$, fyrir $j = 1, 2, ..., n-1$.
- Vegurinn kallast *rás* (e. *cycle*) ef $v_1 = v_n$.
- Vegurinn kallast *einfaldur* (e. *simple*) ef engir tveir hnútar í $v_1, v_2, ..., v_n$ eru eins.
- Rás kallast *einföld* (e. *simple*) ef engir tveir hnútar í $v_1, v_2, ..., v_{n-1}$ eru eins.

- Runa hnúta $v_1, v_2, ..., v_n$ kallast *vegur* (e. *path*) ef $(v_j, v_{j+1}) \in E$, fyrir $j = 1, 2, ..., n-1$.
- Vegurinn kallast *rás* (e. *cycle*) ef $v_1 = v_n$.
- Vegurinn kallast *einfaldur* (e. *simple*) ef engir tveir hnútar í $v_1, v_2, ..., v_n$ eru eins.
- Rás kallast *einföld* (e. *simple*) ef engir tveir hnútar í $v_1, v_2, ..., v_{n-1}$ eru eins.
- Við segjum að vegurinn $v_1, v_2, ..., v_n$ *liggi á milli hnútanna* v_1 *og* v_n.

- Runa hnúta $v_1, v_2, ..., v_n$ kallast *vegur* (e. *path*) ef $(v_j, v_{j+1}) \in E$, fyrir $j = 1, 2, ..., n-1$.
- Vegurinn kallast *rás* (e. *cycle*) ef $v_1 = v_n$.
- Vegurinn kallast *einfaldur* (e. *simple*) ef engir tveir hnútar í $v_1, v_2, ..., v_n$ eru eins.
- Rás kallast *einföld* (e. *simple*) ef engir tveir hnútar í $v_1, v_2, ..., v_{n-1}$ eru eins.
- Við segjum að vegurinn $v_1, v_2, ..., v_n$ *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.

- Runa hnúta $v_1, v_2, ..., v_n$ kallast *vegur* (e. *path*) ef $(v_j, v_{j+1}) \in E$, fyrir $j = 1, 2, ..., n-1$.
- Vegurinn kallast *rás* (e. *cycle*) ef $v_1 = v_n$.
- Vegurinn kallast *einfaldur* (e. *simple*) ef engir tveir hnútar í $v_1, v_2, ..., v_n$ eru eins.
- Rás kallast *einföld* (e. *simple*) ef engir tveir hnútar í $v_1, v_2, ..., v_{n-1}$ eru eins.
- Við segjum að vegurinn $v_1, v_2, ..., v_n$ *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 samanhangandi og inniheldur enga rás.

- Það er engin stöðluð leið til að geyma net í minni.

- Það er engin stöðluð leið til að geyma net í minni.
- Yfirleitt er notað eina af þremur gagnagrindum:

- Það er engin stöðluð leið til að geyma net í minni.
- Yfirleitt er notað eina af þremur gagnagrindum:
  - Leggjalista.

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

- 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.

- Látum $G = (V, E)$ tákna netið okkar.
- Þar sem V er endanlegt megum við gera ráð fyrir að $V = \{1, 2, ..., n\}$, þar sem n er fjöldi hnúta í G.

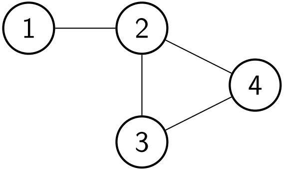
- Látum $G = (V, E)$ tákna netið okkar.
- Þar sem V er endanlegt megum við gera ráð fyrir að $V = \{1, 2, ..., n\}$, þar sem n er fjöldi hnúta í G.
- Látum m vera fjölda leggja í G.

- Látum $G = (V, E)$ tákna netið okkar.
- Þar sem V er endanlegt megum við gera ráð fyrir að $V = \{1, 2, ..., n\}$, 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.

- Látum $G = (V, E)$ tákna netið okkar.
- Þar sem V er endanlegt megum við gera ráð fyrir að $V = \{1, 2, ..., n\}$, 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).

- Látum $G = (V, E)$ tákna netið okkar.
- Þar sem V er endanlegt megum við gera ráð fyrir að $V = \{1, 2, ..., n\}$, 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.

- Látum $G = (V, E)$ tákna netið okkar.
- Þar sem V er endanlegt megum við gera ráð fyrir að $V = \{1, 2, ..., n\}$, 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.



$$
\begin{aligned}
L = [\ & \\
& (1,2), \\
& (2,3), \\
& (2,4), \\
& (3,4) \\
]\ &
\end{aligned}
$$

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

- 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.

- 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.

- 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 $\mathcal{O}(\quad)$ tíma að upphafsstilla A.

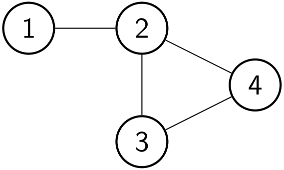
- Látum A vera $V \times V$ fylki þannig að $A_{uv} = 1$ ef (u, v) er í E, en $A_{uv} = 0$ annars.
- Við köllum A *nágrannafylki* netsins G.
- Takið eftir að það tekur $\mathcal{O}(V^2)$ tíma að upphafsstilla A.

- 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 upphafsstilla A.
- Svo þessi aðferð er oft of hæg ef, til dæmis ef $V = 10^5$ (sem er oft raunin).

- 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 upphafsstilla 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 $\mathcal{O}(\quad)$ tíma.

- 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 upphafsstilla 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.

- 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 upphafsstilla 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.

- Látum nú L tákna lista af n listum.
- Táknum j-ta lista L með L_j.

- 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.

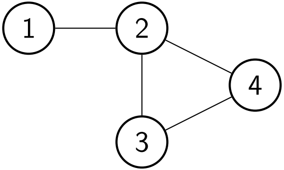
- Látum nú L tákna lista af n listum.
- Táknum j-ta lista L með L_j.
- Látum nú L_u innihalda alla nágranna hnútsins u í netinu G, án endurtekningar.
- Við köllum L *nágrannalista* (fleirtölu) netsins G og L_u *nágrannalista* (eintölu) hnútsins u í netinu G.

- 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.

- 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 $\mathcal{O}(\qquad)$ tíma, óháð röð hnútanna.

- 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.

- 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.



$$L = [$$
$$[2]$$
$$[1, 3, 4]$$
$$[2, 4]$$
$$[2, 3]$$
$$]$$

- Eins og minnst var á áðan eru net oftast gefin með leggjalista, en við vinnum yfirleitt með nágrannalista.

- 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>>`.

- 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.

- 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.

- 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.

- 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.

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;
typedef vector<vi> vvi;

int main()
{
    int i, j, n, m, x, y;
    cin >> n >> m;
    vvi g(n);
    for (i = 0; i < m; i++)
    {
        cin >> x >> y;
        x--, y--;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    for (i = 0; i < n; i++)
    {
        printf("%d: ", i + 1);
        for (j = 0; j < g[i].size(); j++) printf("%d ", g[i][j] + 1);
        printf("\n");
    }
    return 0;
}
```

- Við segjum að þrenndin (V, E, w), þar sem (V, E) er net og $w\colon E \to \mathbb{R} \cup \{-\infty, \infty\}$, sé *vegið net* (e. *weighted graph*).

- Við segjum að þrenndin (V, E, w), þar sem (V, E) er net og $w\colon E \to \mathbb{R} \cup \{-\infty, \infty\}$, sé *vegið net* (e. *weighted graph*).
- Það sem við erum í rauninni að gera er að gefa hverjum legg í netinu vigt.

- Við segjum að þrenndin (V, E, w), þar sem (V, E) er net og $w\colon E \to \mathbb{R} \cup \{-\infty, \infty\}$, 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.

- Við segjum að þrenndin (V, E, w), þar sem (V, E) er net og $w\colon E \to \mathbb{R} \cup \{-\infty, \infty\}$, 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.

- Þegar við geymum vegin net í einhverri gagnagrind þá bætum við yfirleitt bara vigtinni við.
- Leggjalisti vegins nets er því listi af þrenndum.

- 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.

- 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.

- 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.

- 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>>>`.

- 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 stytta þetta niður í `vvii`.

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef pair<int, int> ii;
typedef vector<ii> vii;
typedef vector<vii> vvii;

int main()
{
    int i, j, n, m, x, y, w;
    cin >> n >> m;
    vvii g(n);
    for (i = 0; i < m; i++)
    {
        cin >> x >> y >> w;
        x--, y--;
        g[x].push_back(ii(y, w));
        g[y].push_back(ii(x, w));
    }
    printf("Nágrannar:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d: ", i + 1);
        for (j = 0; j < g[i].size(); j++) printf("%2d ", g[i][j].first + 1);
        printf("\n");
    }
    printf("Vigtir:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d: ", i + 1);
        for (j = 0; j < g[i].size(); j++) printf("%2d ", g[i][j].second);
        printf("\n");
    }
    return 0;
}
```