Netafræði I

Óstefnd og óvigtuð net

Atli Fannar Franklín

25. febrúar 2019

Efnisyfirlit

Hvað er net?

- Þið ættuð að muna eftir netum úr stærðfræðimynstrum.

Hvað er net?

- Þið ættuð að muna eftir netum úr stærðfræðimynstrum.
- Þetta er mengi af *hnútum* sem tengdir eru með *leggjum* sem liggja frá einhverjum hnút í einhvern hnút.

Hvað er net?

- Þið ættuð að muna eftir netum úr stærðfræðimynstrum.
- Þetta er mengi af *hnútum* sem tengdir eru með *leggjum* sem liggja frá einhverjum hnút í einhvern hnút.
- Formlega séð er net þrennd (V, E, ϵ) þar sem V er mengi hnúta, E er mengi leggja og ϵ er vörpun úr E í $\mathcal{P}(V)$ þ.a. $|\epsilon(v)| = 1$ eða 2 sem úthlutar hverjum legg *endapunktum* sínum.

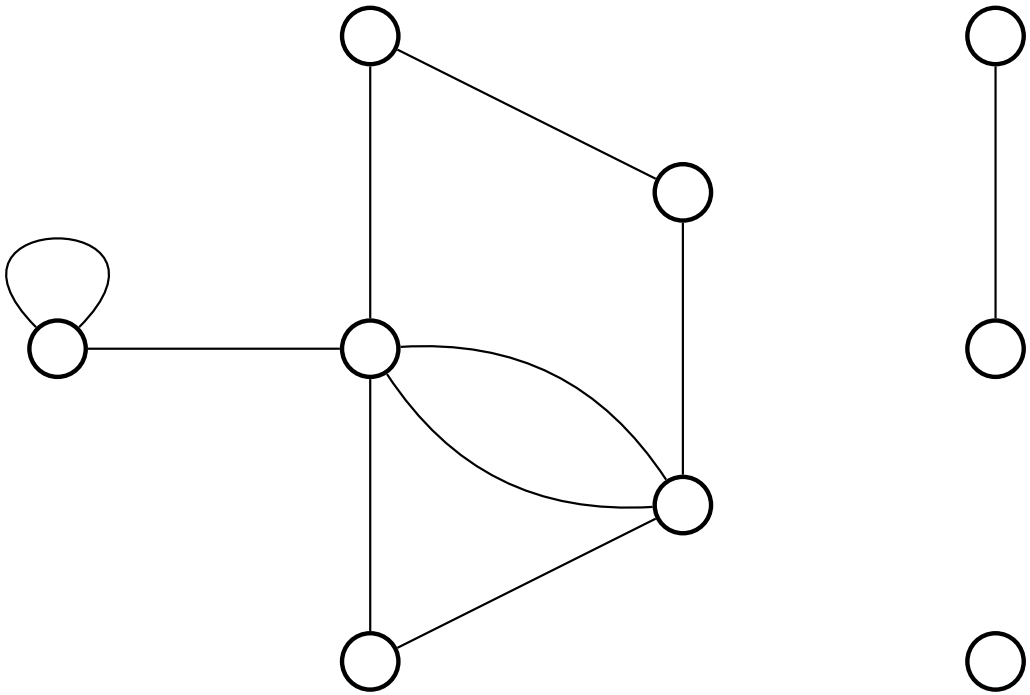
Hvað er net?

- Þið ættuð að muna eftir netum úr stærðfræðimynstrum.
- Þetta er mengi af *hnútum* sem tengdir eru með *leggjum* sem liggja frá einhverjum hnút í einhvern hnút.
- Formlega séð er net þrennd (V, E, ϵ) þar sem V er mengi hnúta, E er mengi leggja og ϵ er vörpun úr E í $\mathcal{P}(V)$ þ.a. $|\epsilon(v)| = 1$ eða 2 sem úthlutar hverjum legg *endapunktum* sínum.
- Ef hnútar hafa legg á milli sín köllum við þá nágranna. Ef tveir leggir hafa sameiginlegan endapunkt köllum við þá aðlæga. Einnig segjum að að endapunktur sé aðlægur legg sínum. Leggur með einn endapunkt kallast snara. Net þar sem engir tveir leggir hafa sömu endapunkta og hefur engar snörur kallast *einfalt*.

Hvað er net?

- Þið ættuð að muna eftir netum úr stærðfræðimynstrum.
- Þetta er mengi af *hnútum* sem tengdir eru með *leggjum* sem liggja frá einhverjum hnút í einhvern hnút.
- Formlega séð er net þrennd (V, E, ϵ) þar sem V er mengi hnúta, E er mengi leggja og ϵ er vörpun úr E í $\mathcal{P}(V)$ þ.a. $|\epsilon(v)| = 1$ eða 2 sem úthlutar hverjum legg *endapunktum* sínum.
- Ef hnútar hafa legg á milli sín köllum við þá nágranna. Ef tveir leggir hafa sameiginlegan endapunkt köllum við þá aðlæga. Einnig segjum að að endapunktur sé aðlægur legg sínum. Leggur með einn endapunkt kallast snara. Net þar sem engir tveir leggir hafa sömu endapunkta og hefur engar snörur kallast *einfalt*.
- Langoftast vinnum við með einföld net.

Dæmi um net



Vegir og rásir

- Runa leggja þar sem hver leggur hefur sameiginlegan endapunkt með þeim sem kom á undan kallast vegur. Ef vegurinn byrjar og endar á sama stað köllum við hann rás. Ef við endurtökum enga hnúta köllum við veginn einfaldan. Rás er einföld ef við endurtökum enga hnúta nema þann síðasta.

Vegir og rásir

- Runa leggja þar sem hver leggur hefur sameiginlegan endapunkt með þeim sem kom á undan kallast vegur. Ef vegurinn byrjar og endar á sama stað köllum við hann rás. Ef við endurtökum enga hnúta köllum við veginn einfaldan. Rás er einföld ef við endurtökum enga hnúta nema þann síðasta.
- Við segjum að tveir hnútar séu tengdir ef til er vegur milli þeirra. Samhengisþáttur í neti er þá óstækkanlegt hlutmengi hnúta sem eru allir innbyrðis tengdir.

Vegir og rásir

- Runa leggja þar sem hver leggur hefur sameiginlegan endapunkt með þeim sem kom á undan kallast vegur. Ef vegurinn byrjar og endar á sama stað köllum við hann rás. Ef við endurtökum enga hnúta köllum við veginn einfaldan. Rás er einföld ef við endurtökum enga hnúta nema þann síðasta.
- Við segjum að tveir hnútar séu tengdir ef til er vegur milli þeirra. Samhengisþáttur í neti er þá óstækkanlegt hlutmengi hnúta sem eru allir innbyrðis tengdir.
- Skilgreina má tré á hundrað vegu, en hér skulum við bara skilgreina það sem samanhangandi net án rása. Net án rása (sem er þá ekki nauðsynlega samanhangandi) er þá kallað skógur.

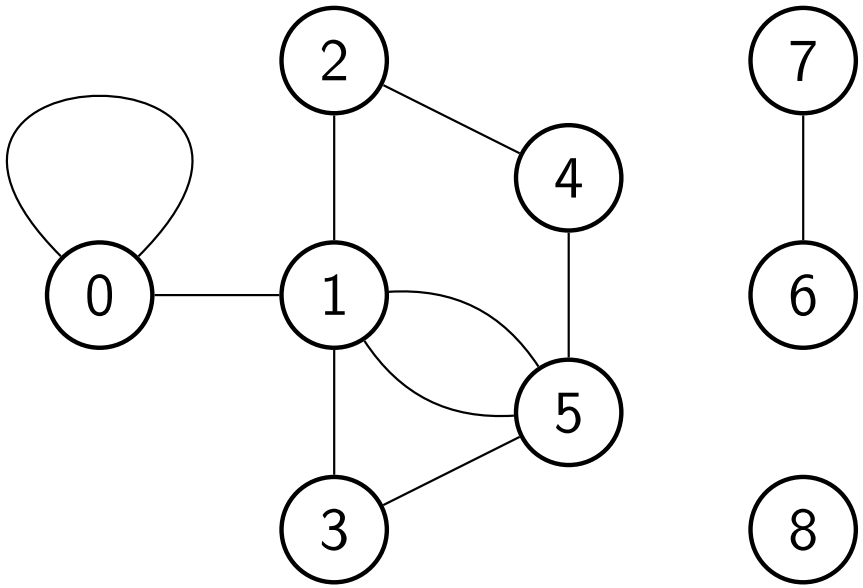
Framsetning í tölvu

- Við höfum þrjár leiðir til að tákna almenn net í tölvu (til eru einhverjar exótískari útgáfur eins og link-cut tree og heavy-light decomposition en við látum þessar þrjár duga)

Framsetning í tölvu

- Við höfum þrjár leiðir til að tákna almenn net í tölvu (til eru einhverjar exótískari útgáfur eins og link-cut tree og heavy-light decomposition en við látum þessar þrjár duga)
- Við höfum þá nágrannaframsetningu, fylkjaframsetningu og listaframsetningu fyrir almenn net. Notum nágrannaframsetninguna langmest. Sjáum nú þessar framsetningar fyrir netið sem við sáum áðan.

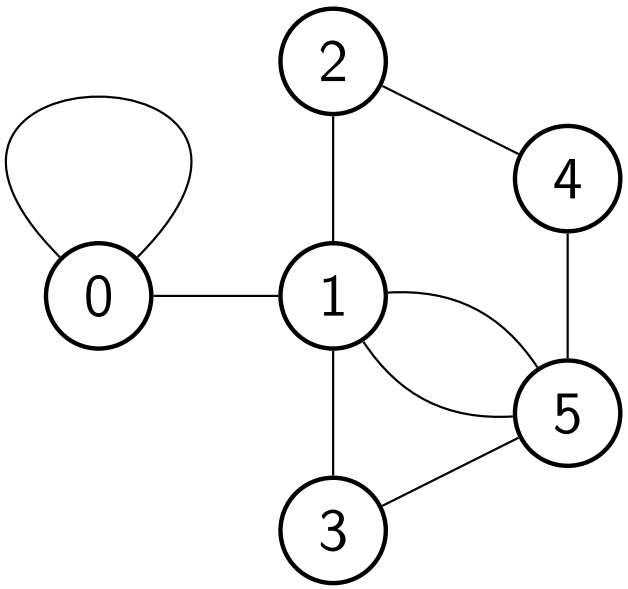
Nágrannaframsetning (adjacency list)

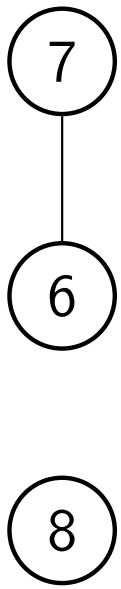



```
G = [
    [0, 1],
    [2, 3, 5, 5],
    [1, 4],
    [1, 5],
    [2, 5],
    [1, 1, 3, 4],
    [7],
    [6],
    []
]
```

- `G` er þá listi lista og `G[i]` er listi nágranna `i`.

Fylkjaframsetning (adjacency matrix)

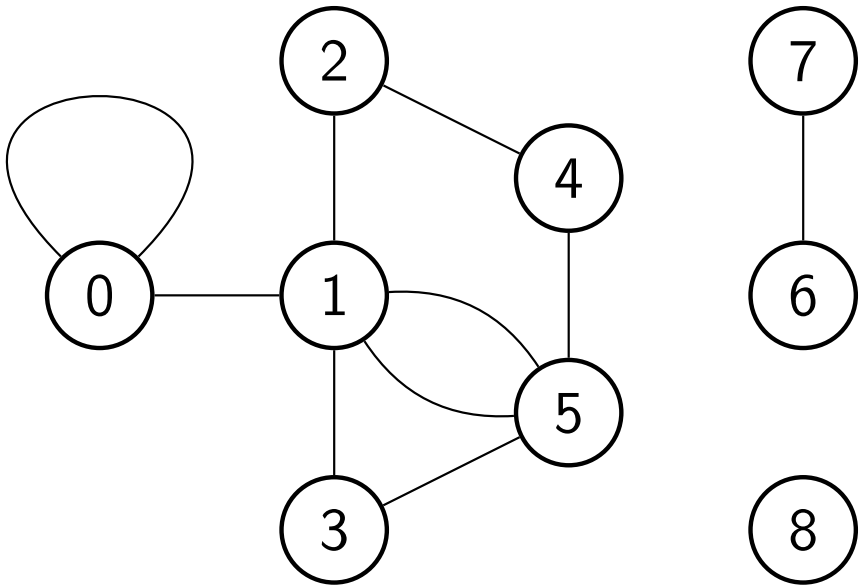





```
G = [
    [1, 1, 0, 0, 0, 0, 0, 0, 0],
    [1, 0, 1, 1, 0, 2, 0, 0, 0],
    [0, 1, 0, 0, 1, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 1, 0, 0, 0],
    [0, 2, 0, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0]
]
```

- `G` er þá listi lista og `G[i][j]` er fjöldi leggja milli `i` og `j`.

Listaframsetning (edge list)




```
G = [
    [0, 0],
    [0, 1],
    [1, 2],
    [1, 3],
    [1, 5],
    [1, 5],
    [2, 4],
    [3, 5],
    [4, 5],
    [6, 7]
]
```

- `G` er þá listi para og hvert par `[i, j]` segir að til sé leggur milli `i` og `j` í netinu.

Efnisyfirlit

Netaleitir

- Mörg dæmi fela það í sér að leita að einhverju í neti eða að ítra í gegnum hnúta nets. Hvernig má gera þetta?

Netaleitir

- Mörg dæmi fela það í sér að leita að einhverju í neti eða að ítra í gegnum hnúta nets. Hvernig má gera þetta?
- Fyrir svona net (sjáum meira í næstu viku fyrir aðrar tegundir af netum) þá höfum við tvær leitar aðferðir. Við köllum þær breiddarleit (Breadth-first search, BFS) og dýptarleit (Depth-first search, DFS).

Netaleitir

- Mörg dæmi fela það í sér að leita að einhverju í neti eða að ítra í gegnum hnúta nets. Hvernig má gera þetta?
- Fyrir svona net (sjáum meira í næstu viku fyrir aðrar tegundir af netum) þá höfum við tvær leitar aðferðir. Við köllum þær breiddarleit (Breadth-first search, BFS) og dýptarleit (Depth-first search, DFS).
- Bæði fela það í sér að byrja bara einhverstaðar og fara svo endurkvæmt í alla nágrannana nema þann sem maður er búinn með. Munurinn felst í því í hvaða röð þetta er gert. Í BFS hendum við nágrönnunum í biðröð en í DFS hendum við þeim á hlaða.

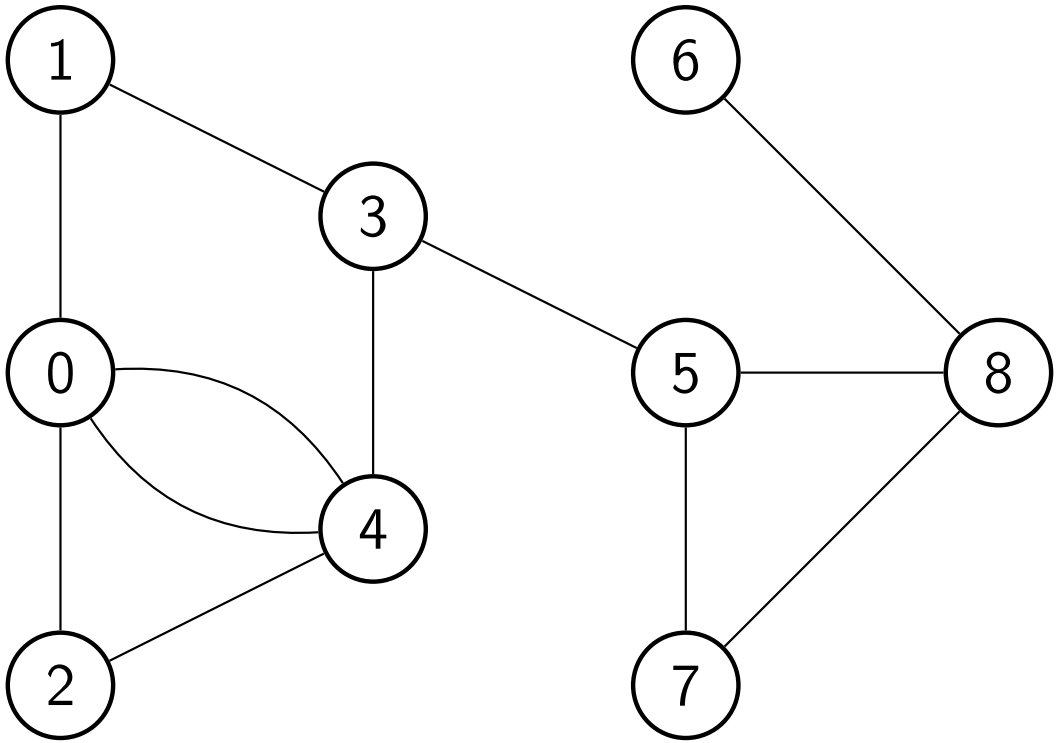
Netaleitir

- Mörg dæmi fela það í sér að leita að einhverju í neti eða að ítra í gegnum hnúta nets. Hvernig má gera þetta?
- Fyrir svona net (sjáum meira í næstu viku fyrir aðrar tegundir af netum) þá höfum við tvær leitar aðferðir. Við köllum þær breiddarleit (Breadth-first search, BFS) og dýptarleit (Depth-first search, DFS).
- Bæði fela það í sér að byrja bara einhverstaðar og fara svo endurkvæmt í alla nágrannana nema þann sem maður er búinn með. Munurinn felst í því í hvaða röð þetta er gert. Í BFS hendum við nágrönnunum í biðröð en í DFS hendum við þeim á hlaða.
- Tímaflækja reikniritanna beggja er $\mathcal{O}(V + E)$ þar sem V er fjöldi hnúta og E er fjöldi leggja.

Netaleitir

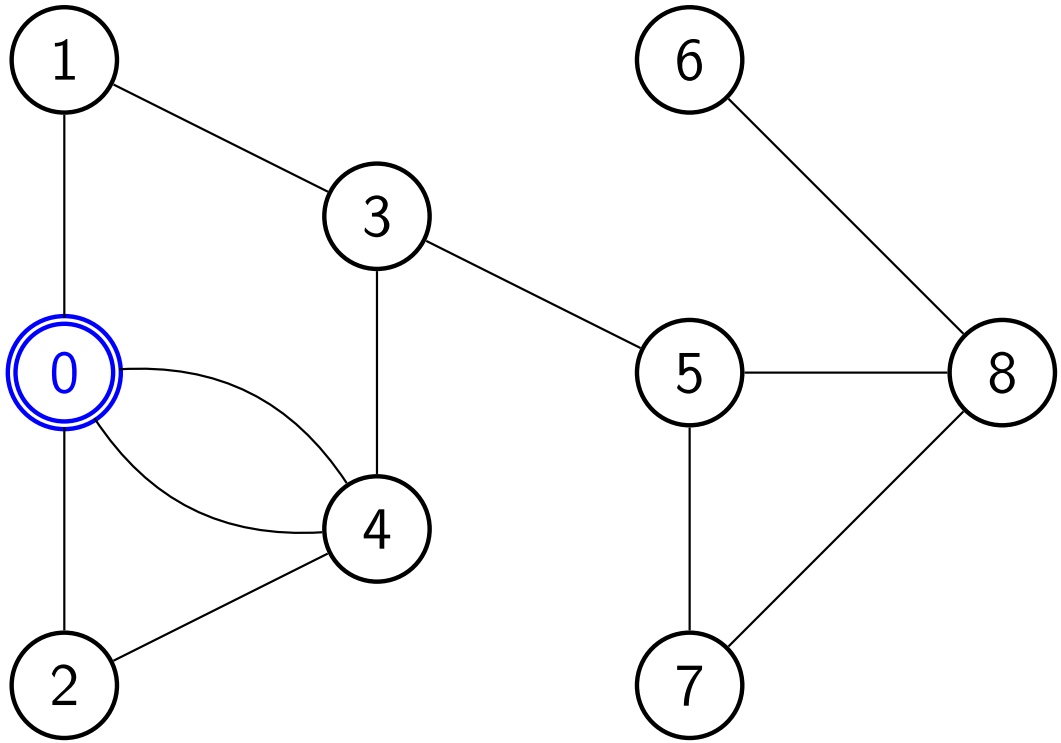
- Mörg dæmi fela það í sér að leita að einhverju í neti eða að ítra í gegnum hnúta nets. Hvernig má gera þetta?
- Fyrir svona net (sjáum meira í næstu viku fyrir aðrar tegundir af netum) þá höfum við tvær leitar aðferðir. Við köllum þær breiddarleit (Breadth-first search, BFS) og dýptarleit (Depth-first search, DFS).
- Bæði fela það í sér að byrja bara einhverstaðar og fara svo endurkvæmt í alla nágrannana nema þann sem maður er búinn með. Munurinn felst í því í hvaða röð þetta er gert. Í BFS hendum við nágrönnunum í biðröð en í DFS hendum við þeim á hlaða.
- Tímaflækja reikniritanna beggja er $\mathcal{O}(V + E)$ þar sem V er fjöldi hnúta og E er fjöldi leggja.
- Skoðum nú dæmi um hvernig BFS og DFS fara með ólíkum hætti í gegnum sama netið.

BFS



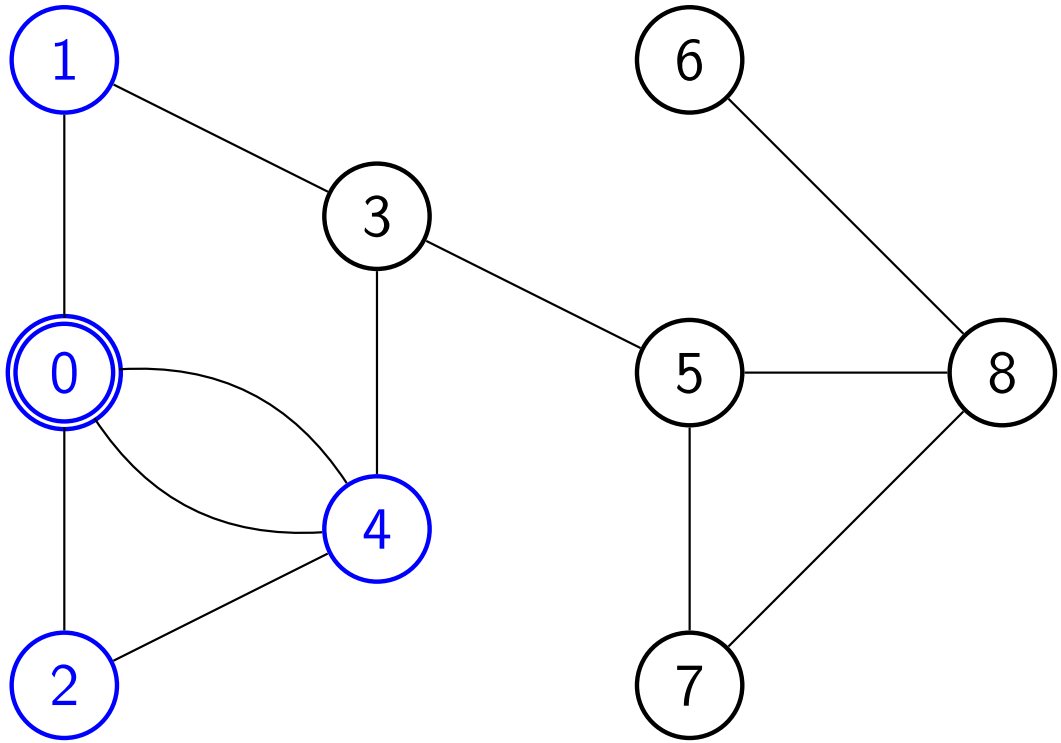

```
q = [
  0
]
```

BFS



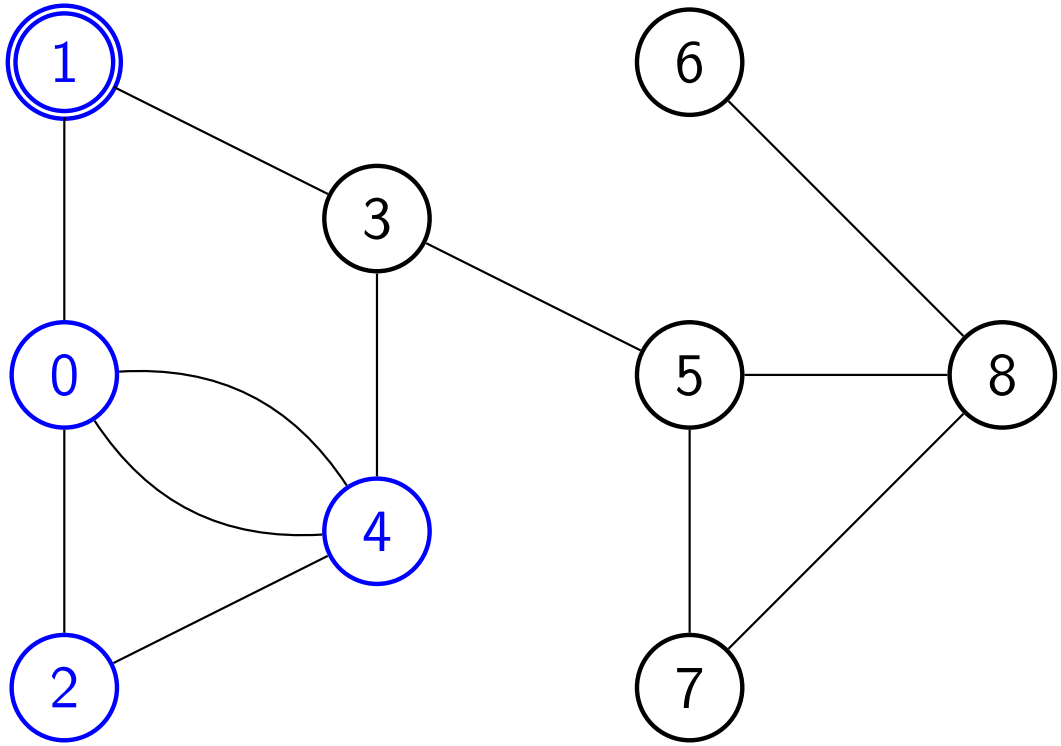

```
q = [
]
```

BFS



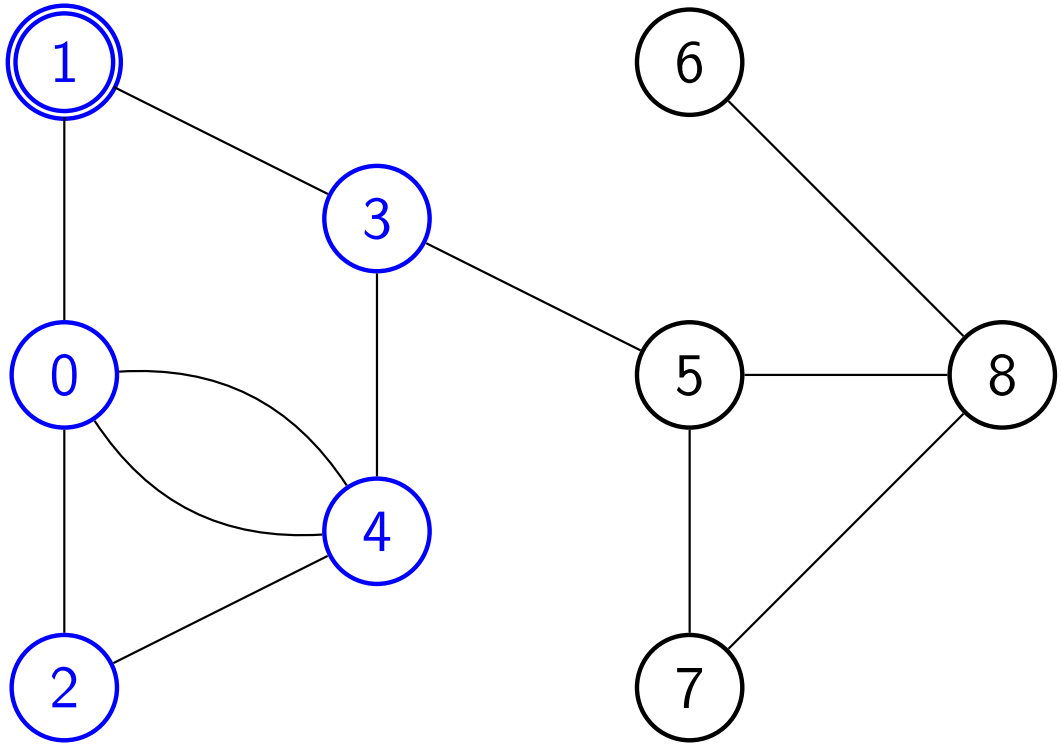

```
q = [
 1,
 2,
 4
]
```

BFS



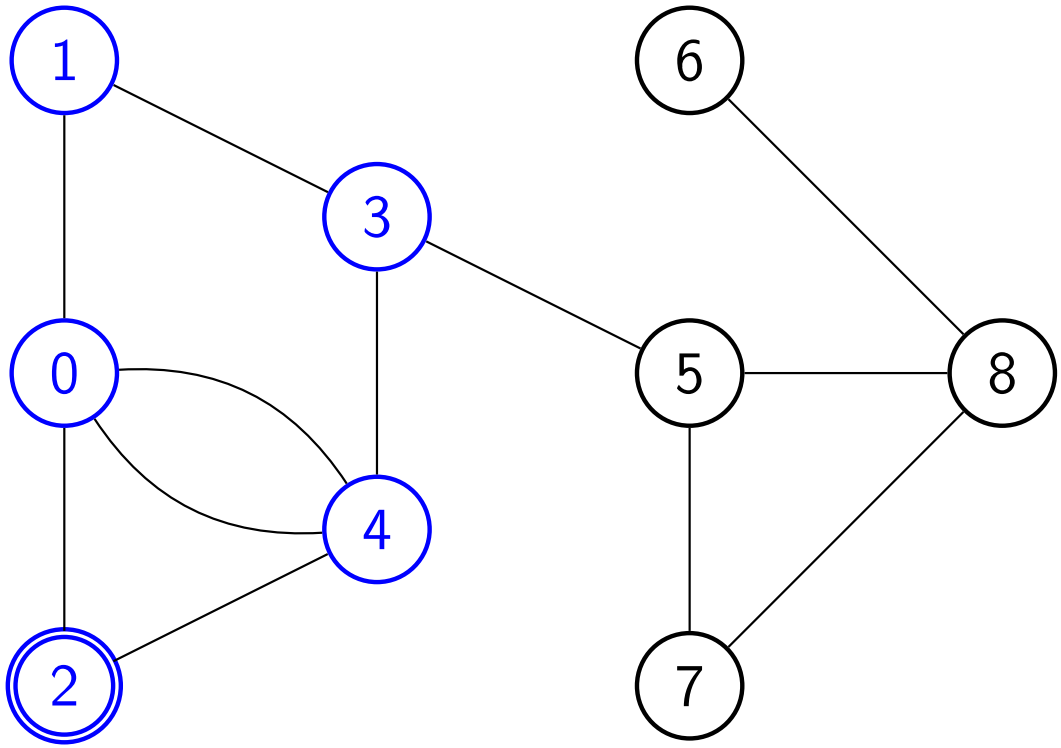

```
q = [
 2,
 4
]
```

BFS



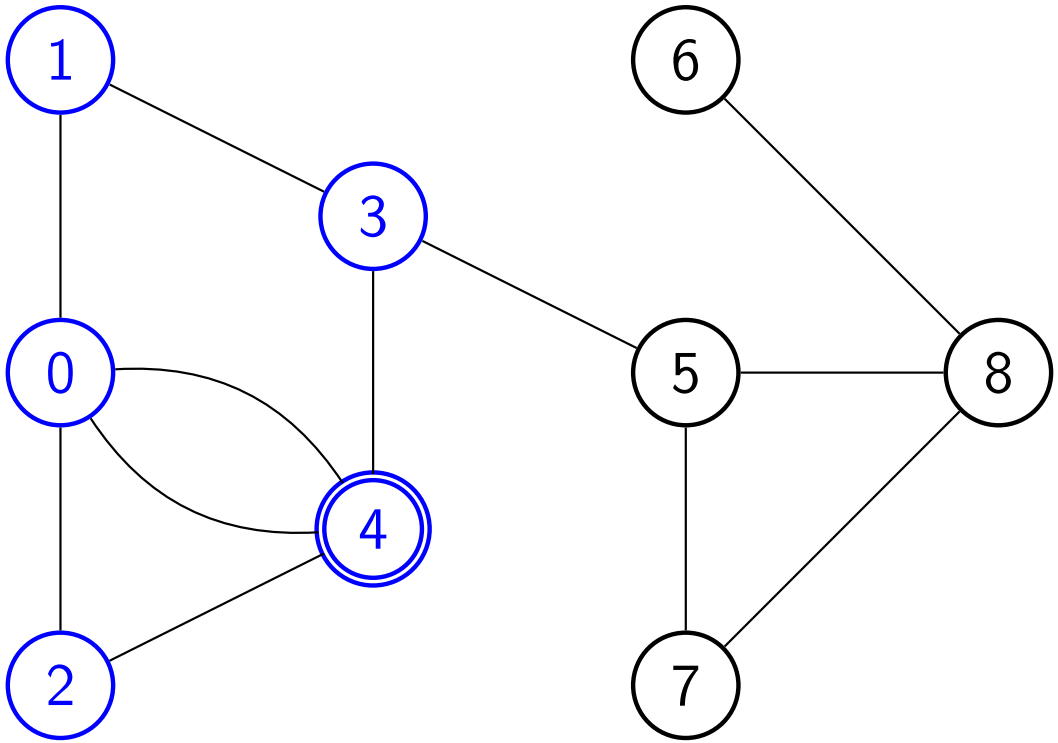

```
q = [
 2,
 4,
 3
]
```

BFS



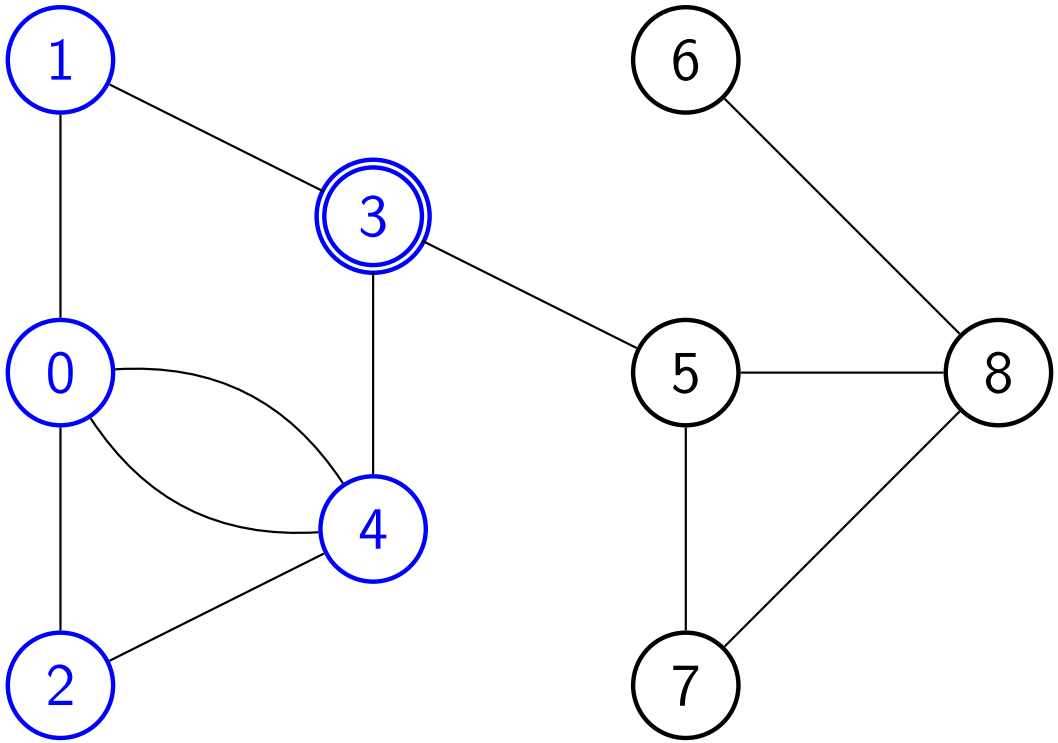

```
q = [
 4,
 3
]
```

BFS



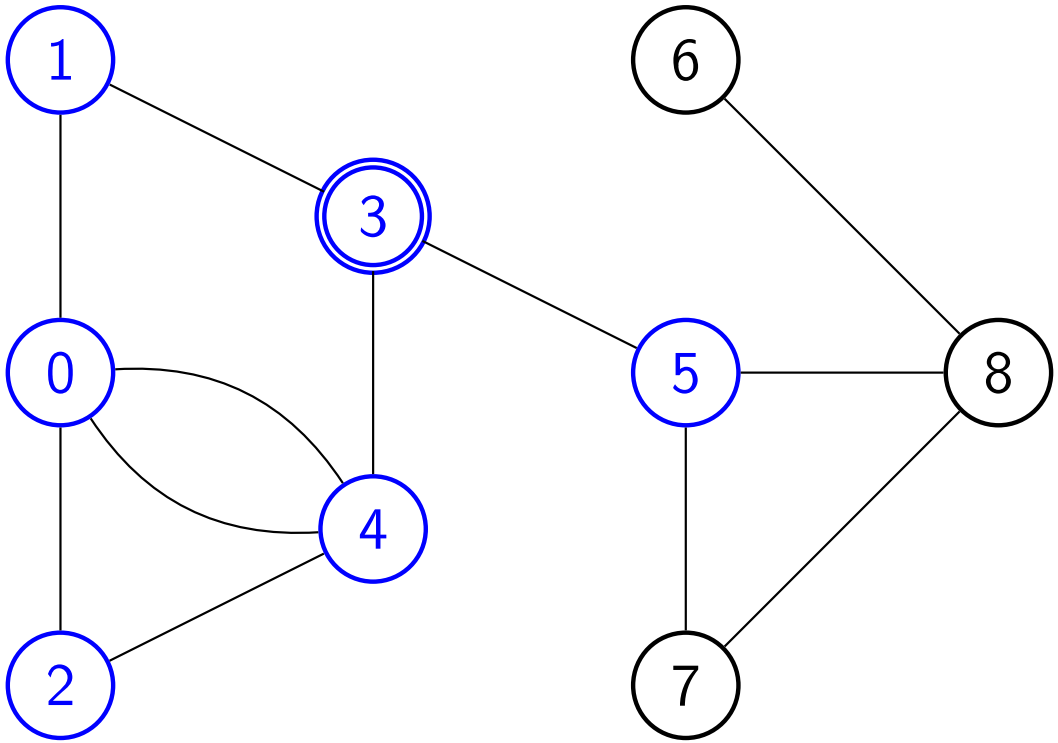

```
q = [
  3
]
```

BFS



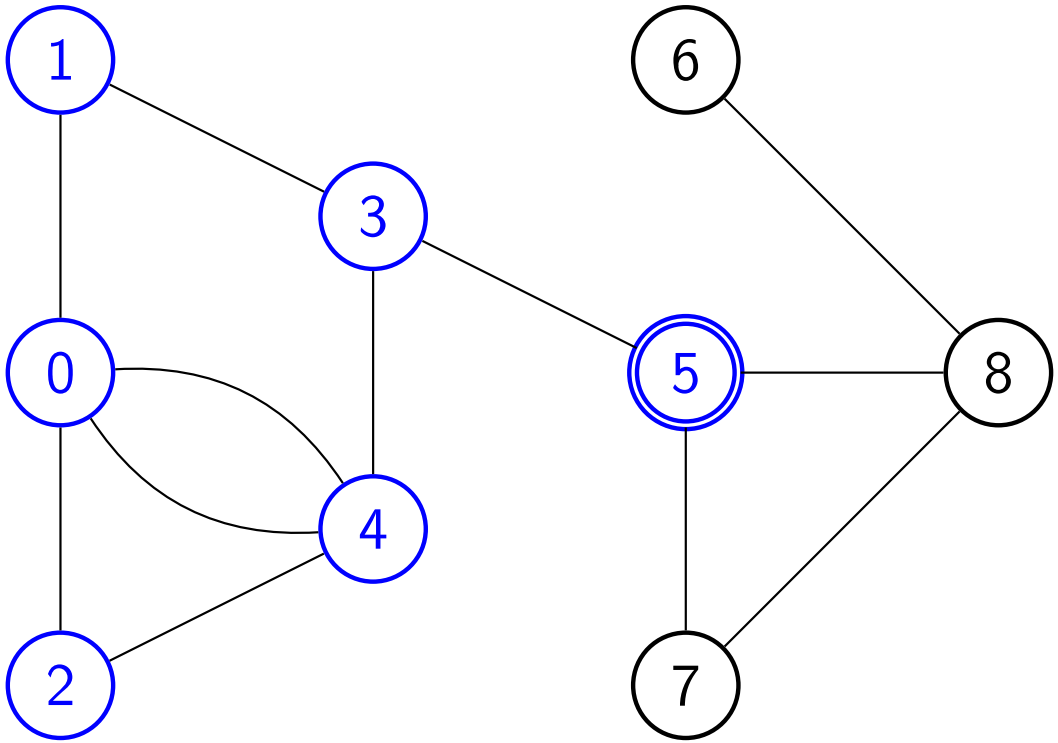

```
q = [
]
```

BFS



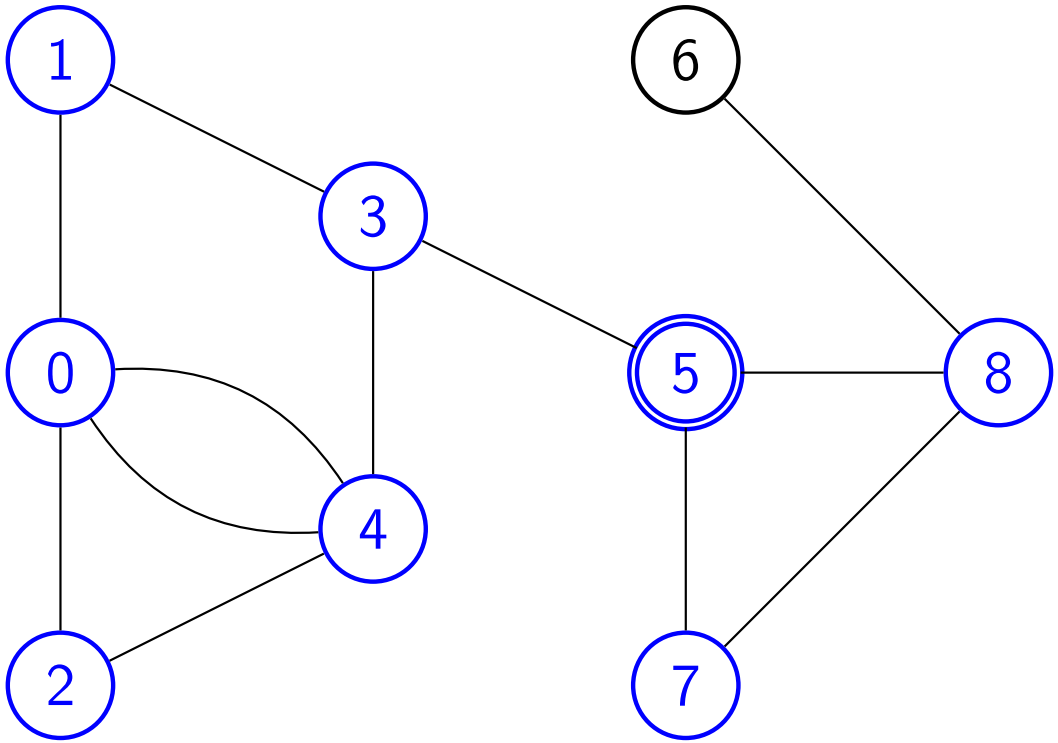

```
q = [
  5
]
```

BFS



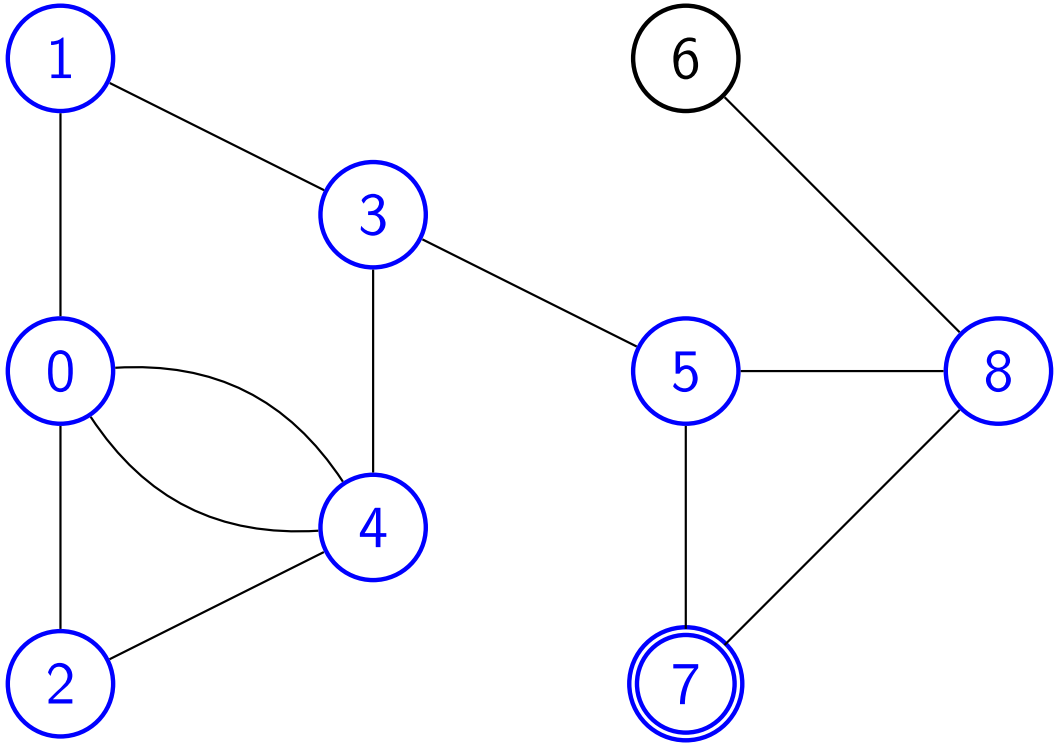

```
q = [
]
```

BFS



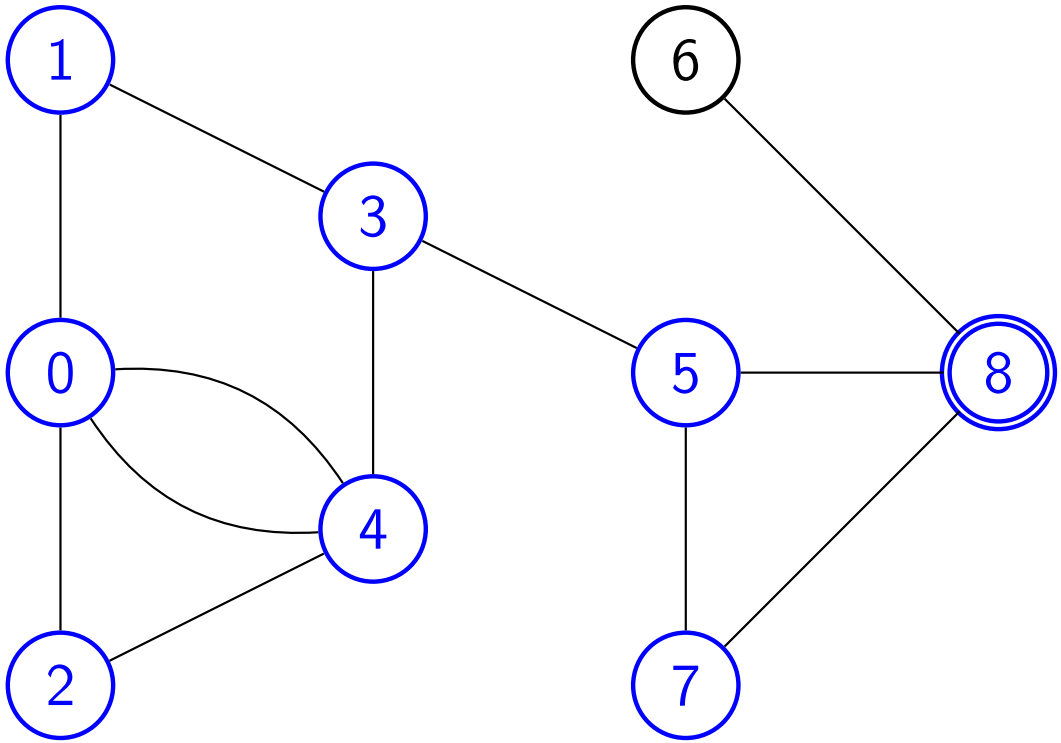

```
q = [
 7,
 8
]
```

BFS



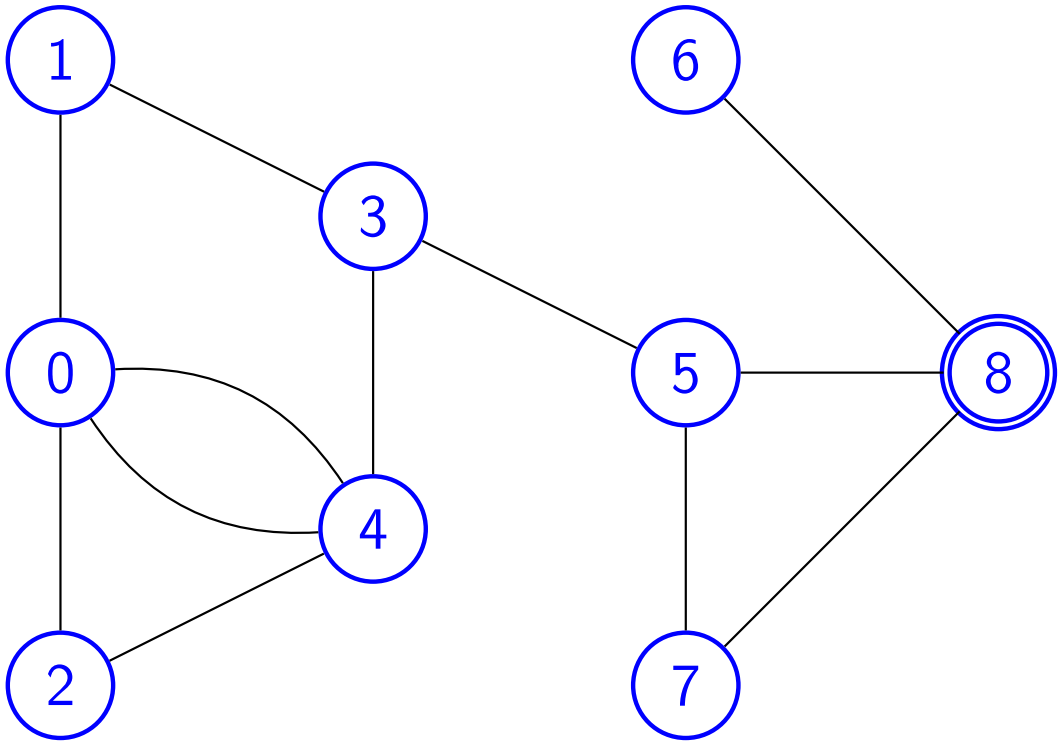

```
q = [
  8
]
```

BFS



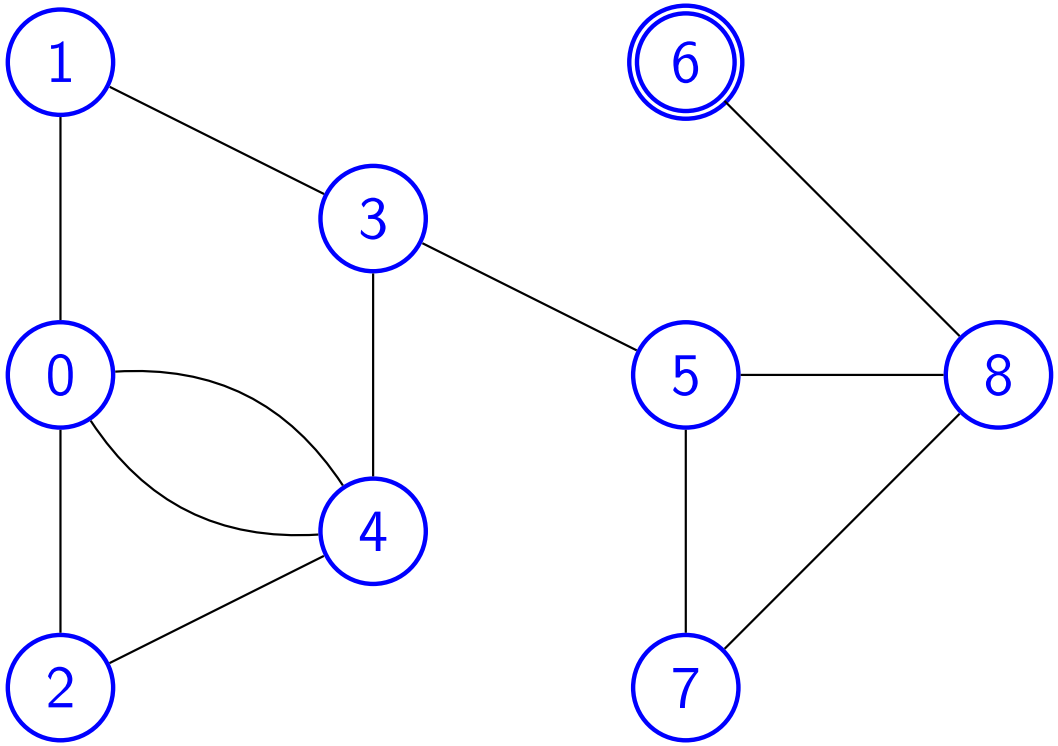

```
q = [
]
```

BFS



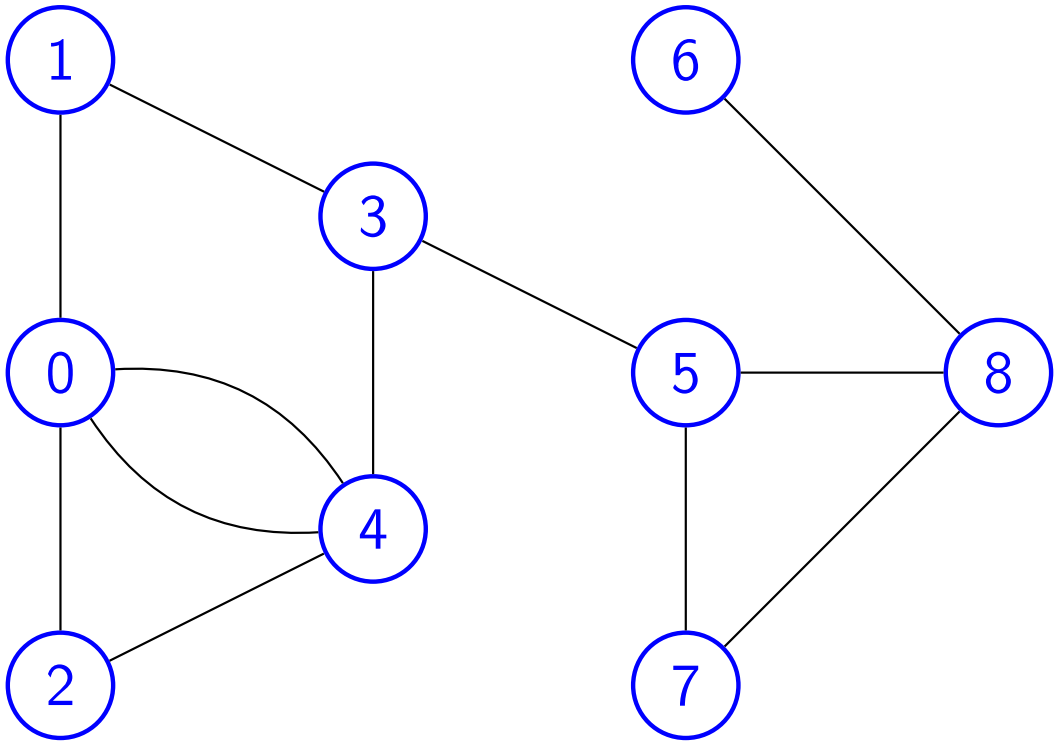

```
q = [
  6
]
```

BFS




```
q = [
]
```

BFS

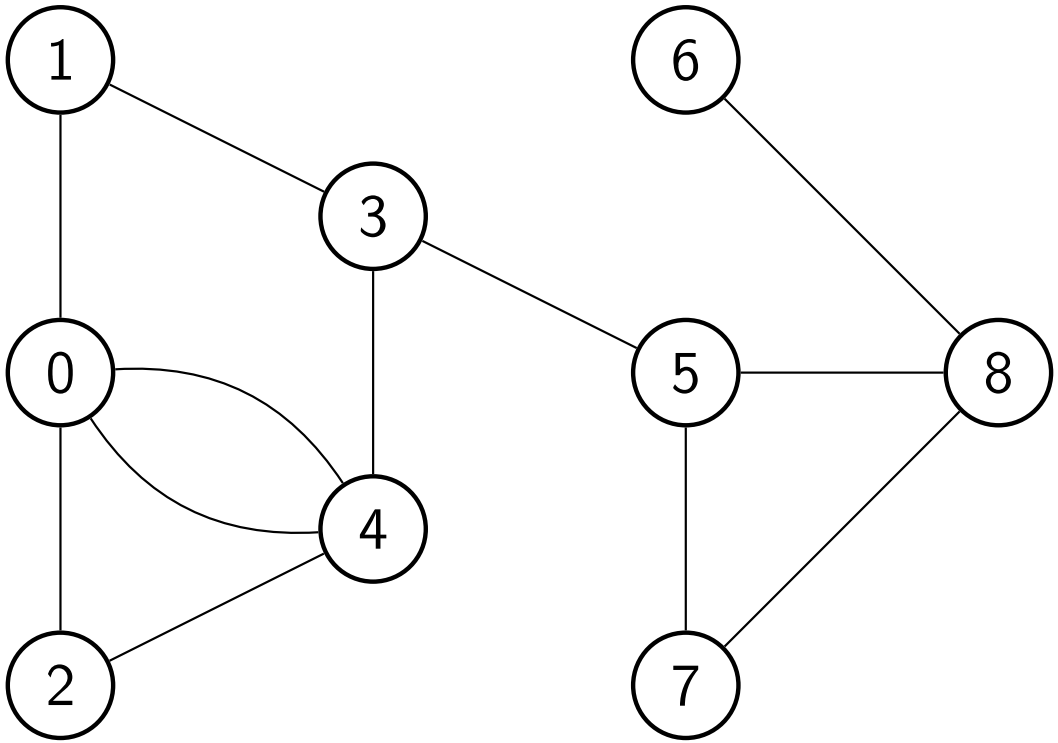



```
q = [
]
```

BFS útfærsla með nágrannalista

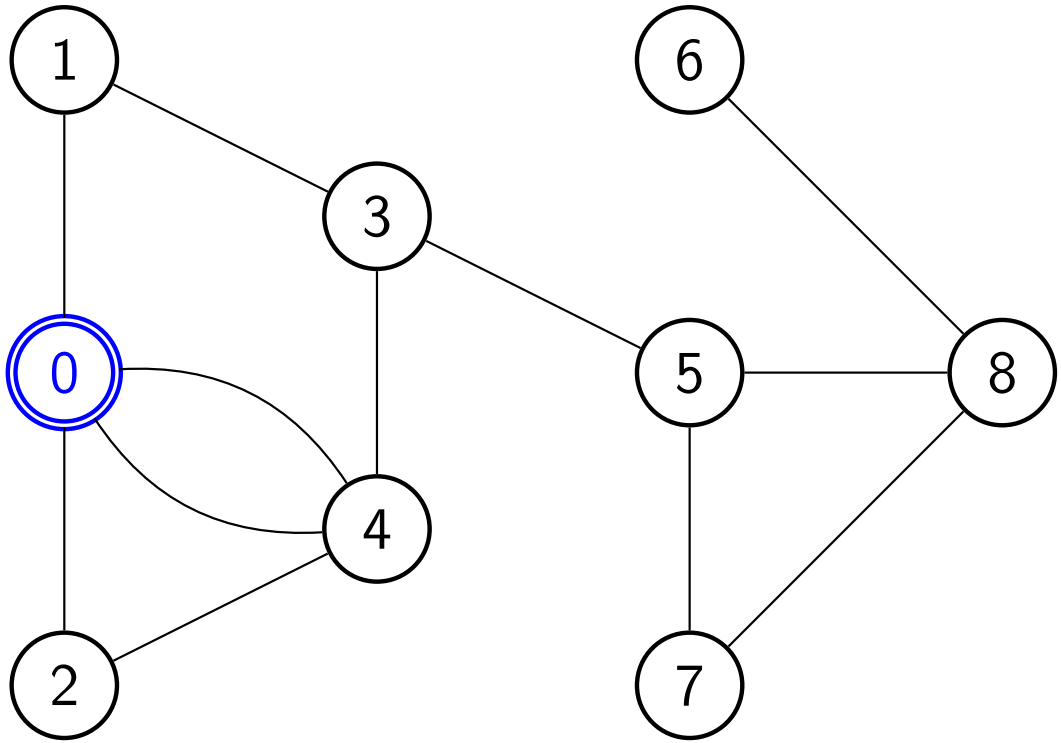
```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;
typedef vector<vi> vvi;

int main() {
    int n, m, a, b;
    cout << "Gefðu upp fjölda hnúta og leggja\n";
    cin >> n >> m;
    cout << "Gefðu upp m leggi (0-index)\n";
    vvi g(n, vi());
    for(int i = 0; i < m; ++i) {
        cin >> a >> b;
        g[a].push_back(b), g[b].push_back(a);
    }
    queue<int> q; vector<bool> d(n, false);
    q.push(0); d[0] = true;
    cout << "Í samhengisþætti 0 fundust:\n";
    while(!q.empty()) {
        int cur = q.front(); q.pop();
        cout << cur << '\n';
        for(int x : g[cur]) {
            if(d[x]) continue;
            d[x] = true; q.push(x); }}}
```

DFS



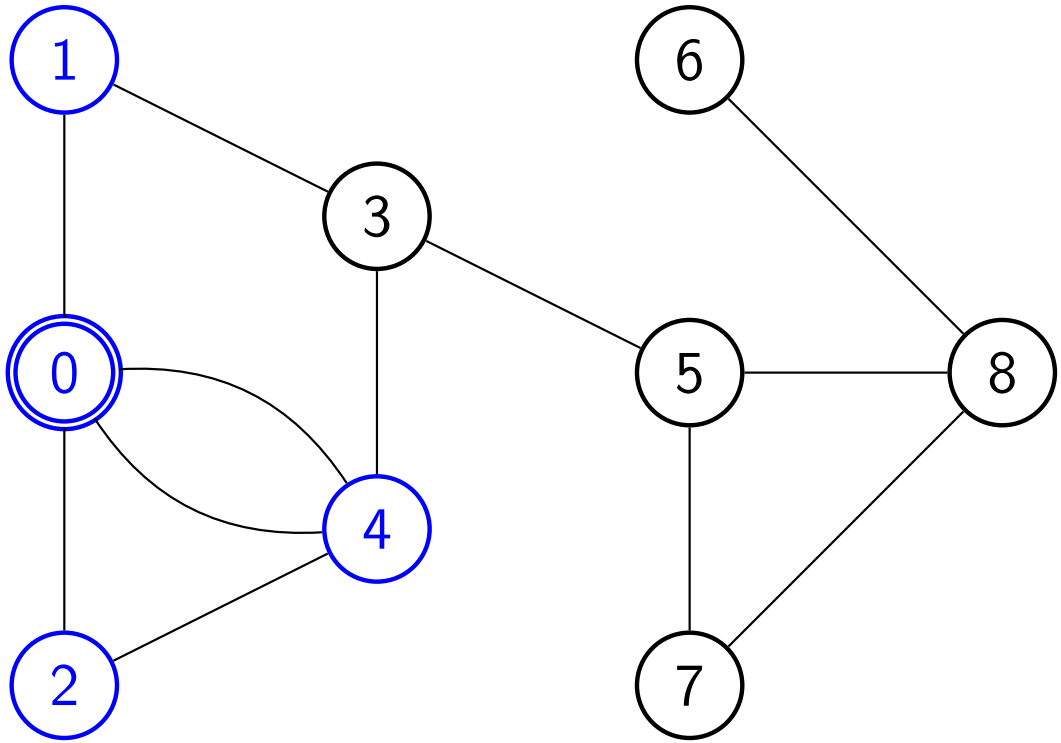

```
s = [
  0
]
```

DFS



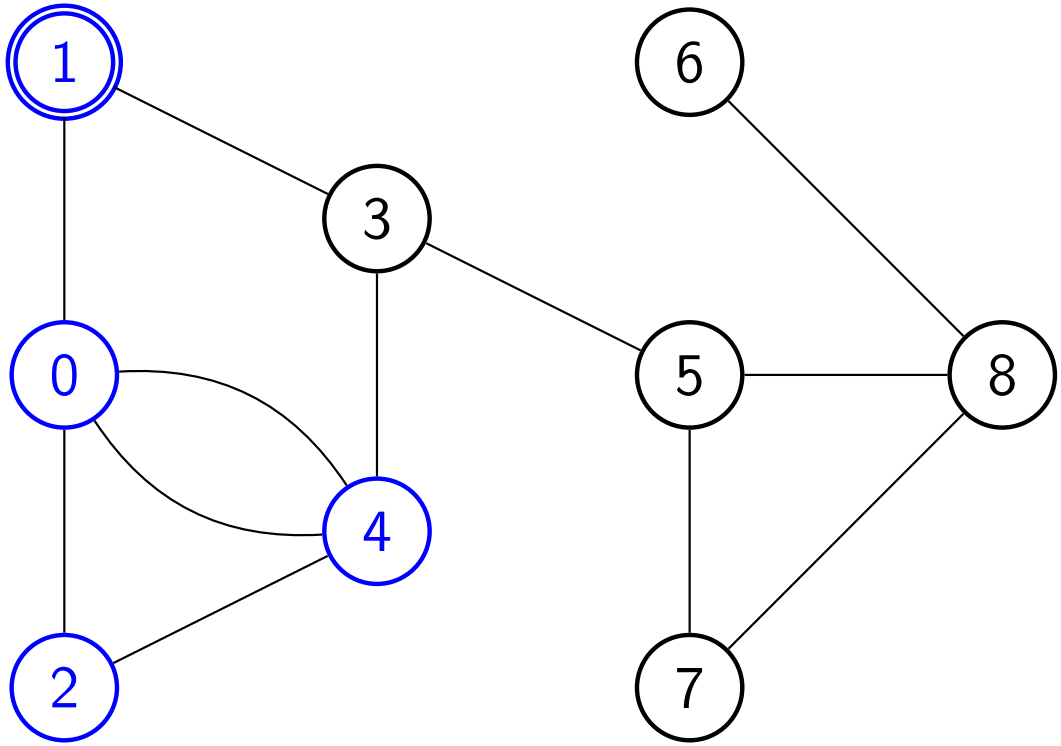

```
s = [
]
```

DFS



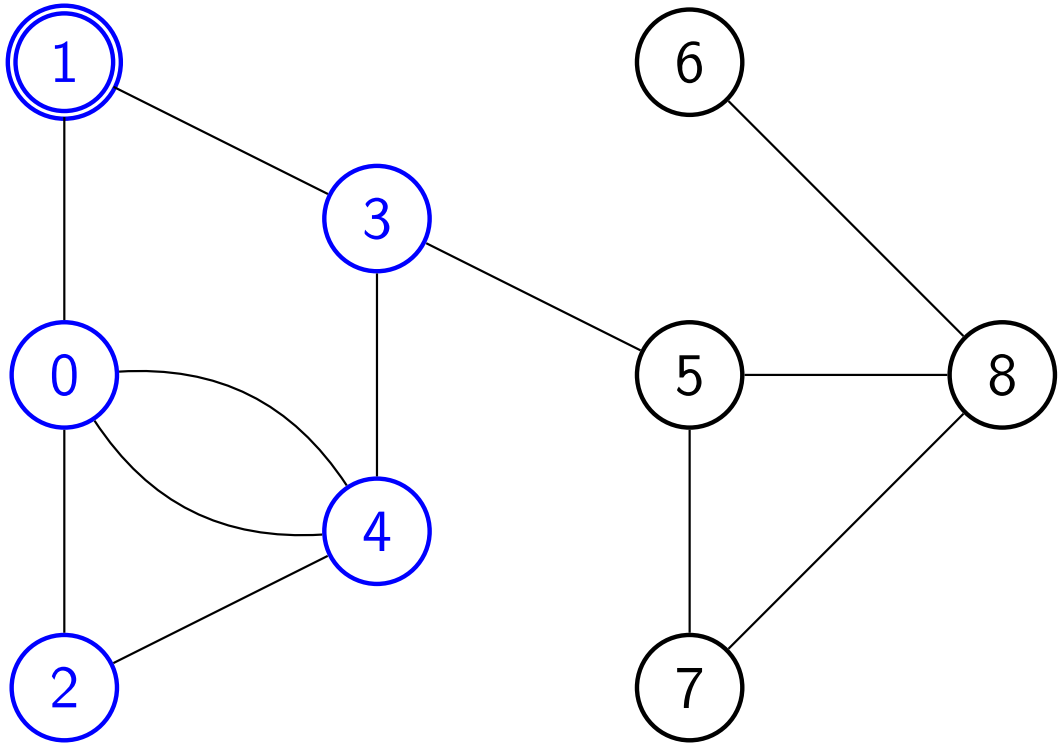

```
s = [
 1,
 2,
 4
]
```

DFS



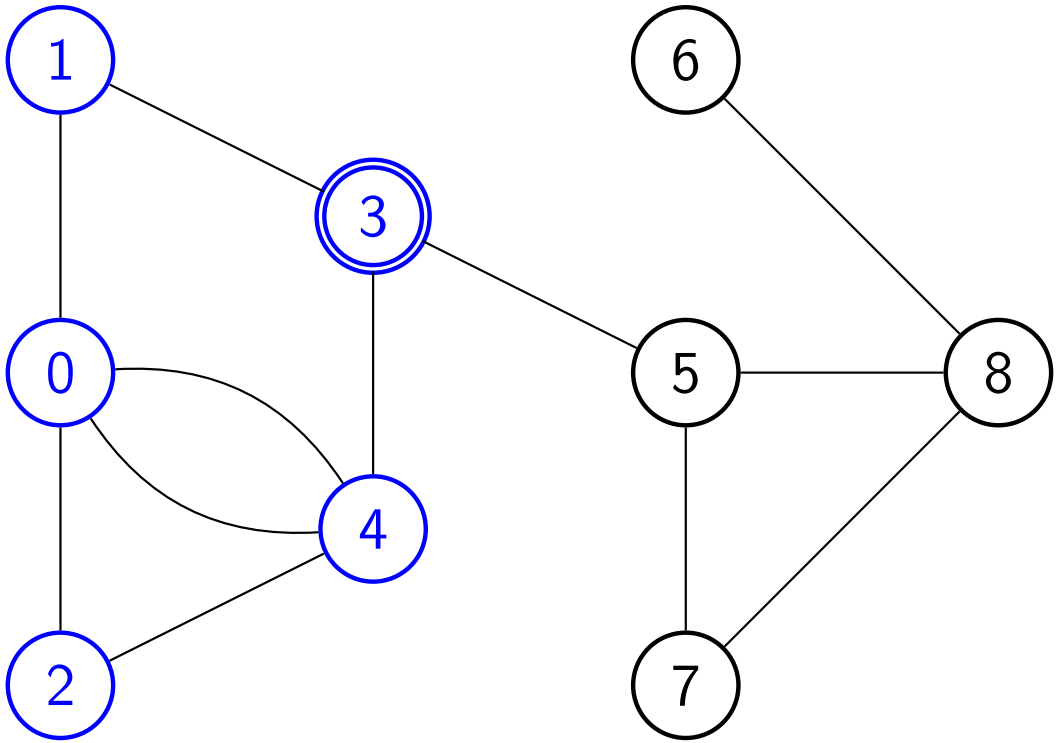

```
s = [
 2,
 4
]
```

DFS



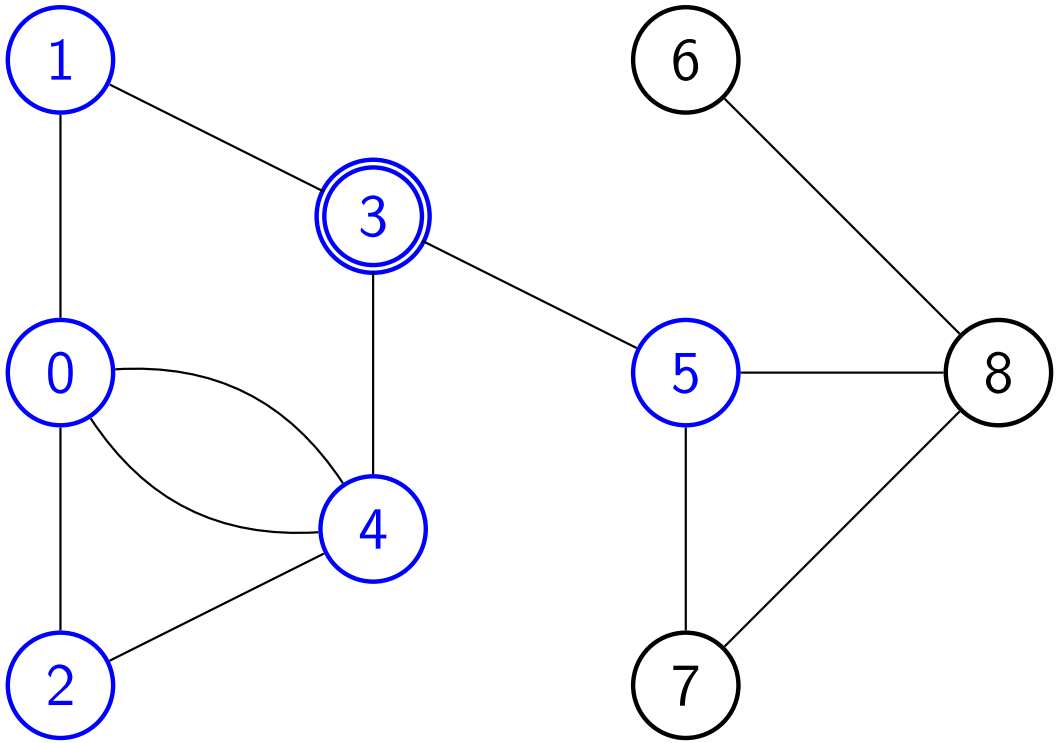

```
s = [
 3,
 2,
 4
]
```

DFS



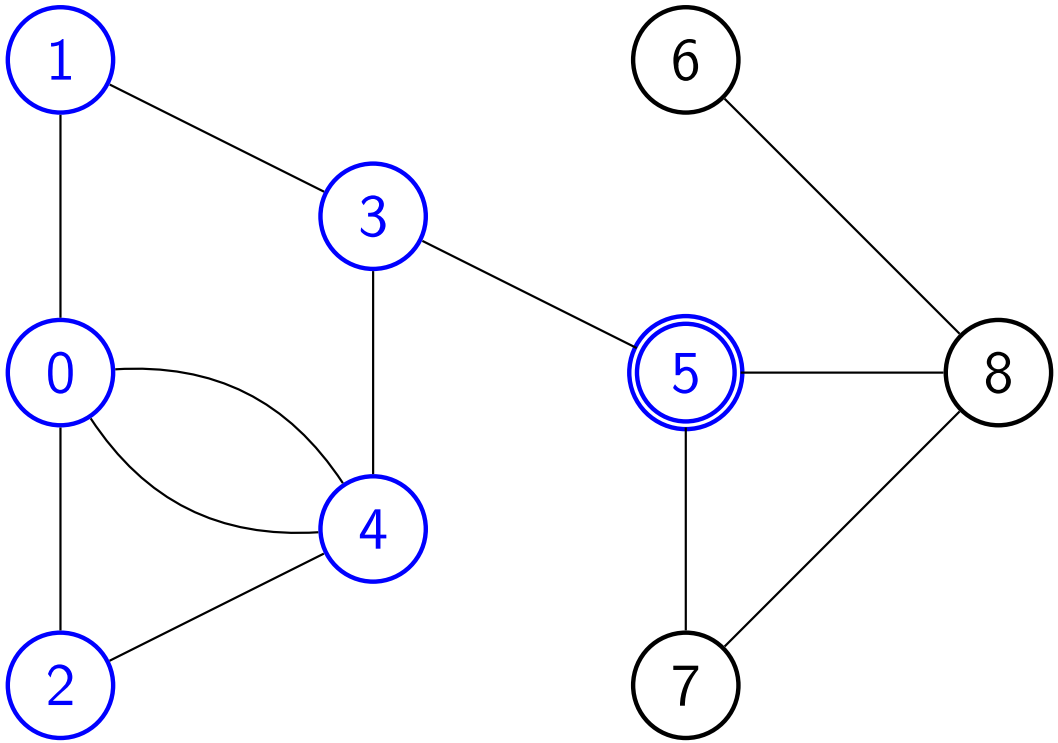

```
s = [
 2,
 4
]
```

DFS



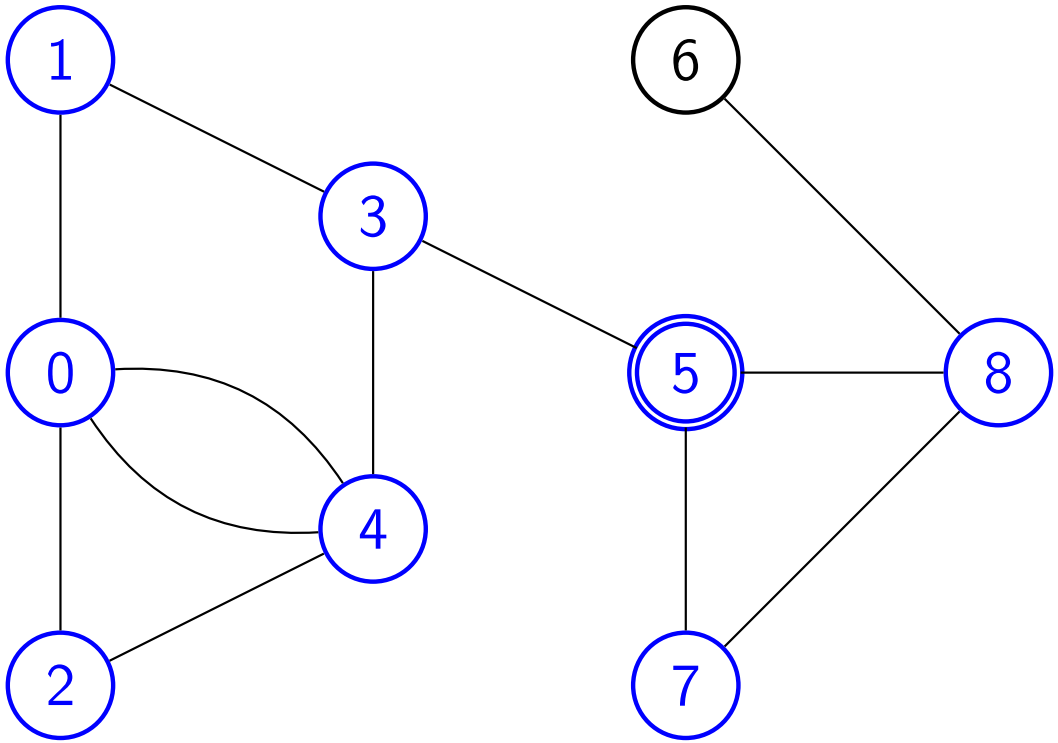

```
s = [
 5,
 2,
 4
]
```

DFS



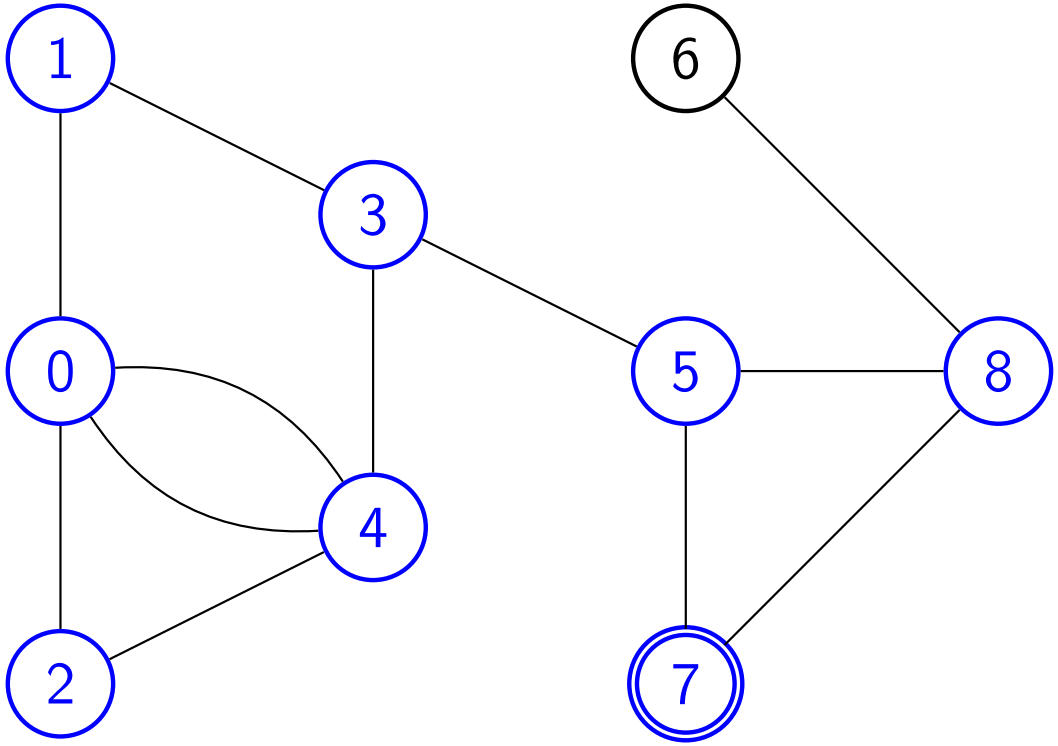

```
s = [
 2,
 4
]
```

DFS



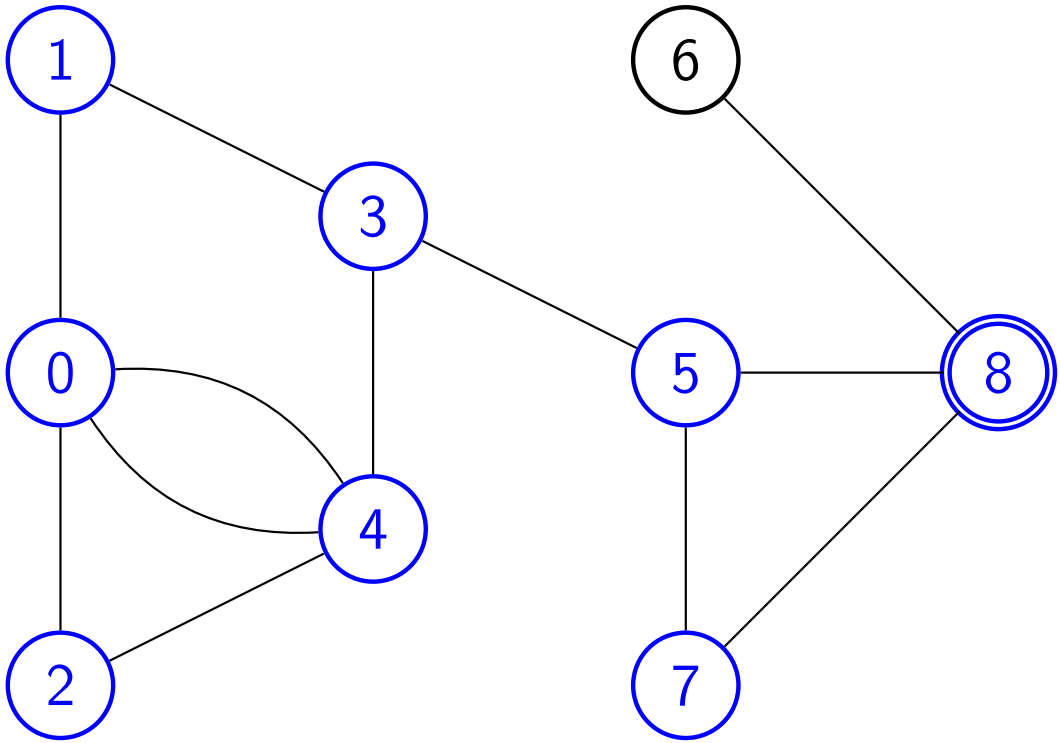

```
s = [
 7,
 8,
 2,
 4
]
```

DFS



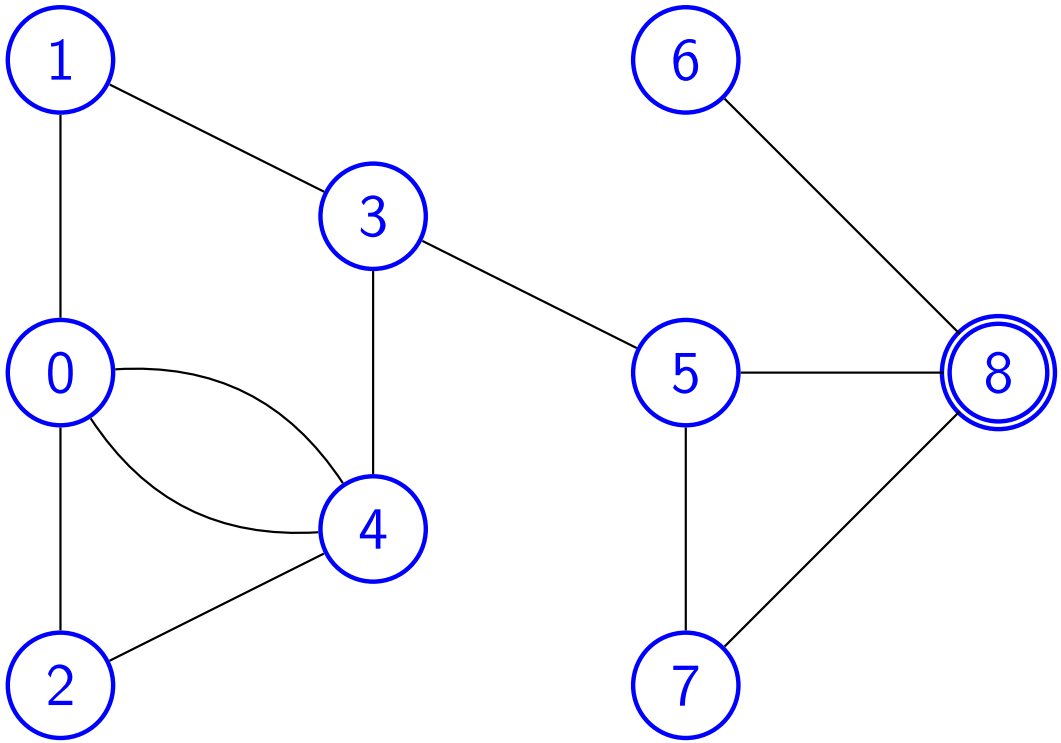

```
s = [
 8,
 2,
 4
]
```

DFS



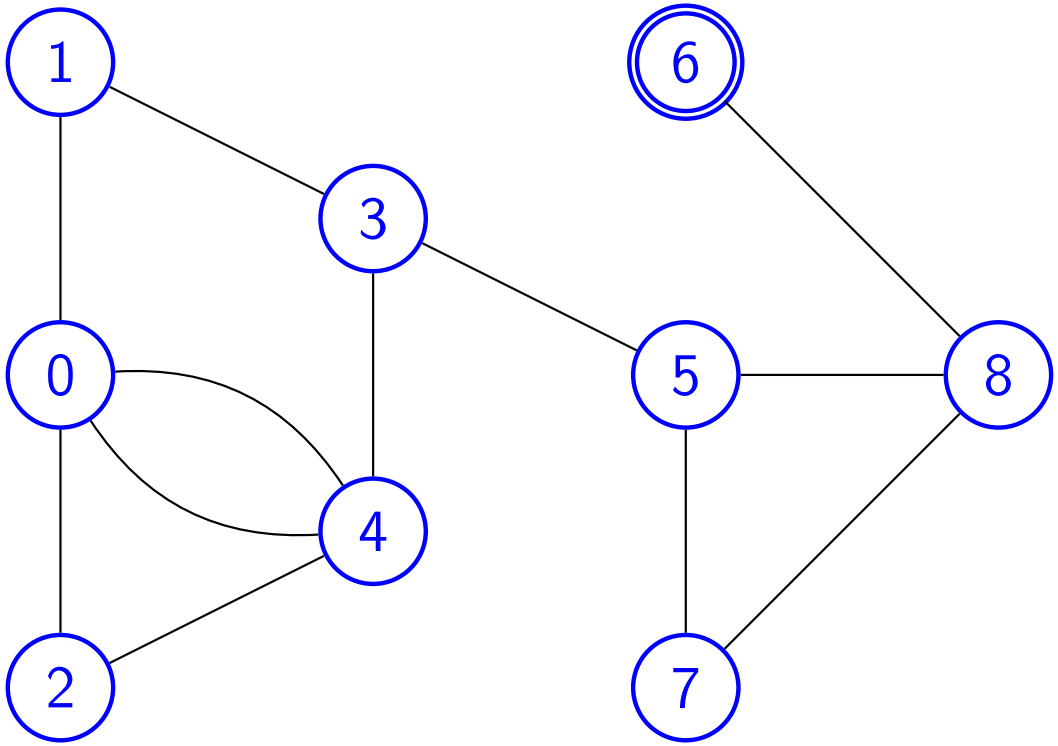

```
s = [
 2,
 4
]
```

DFS



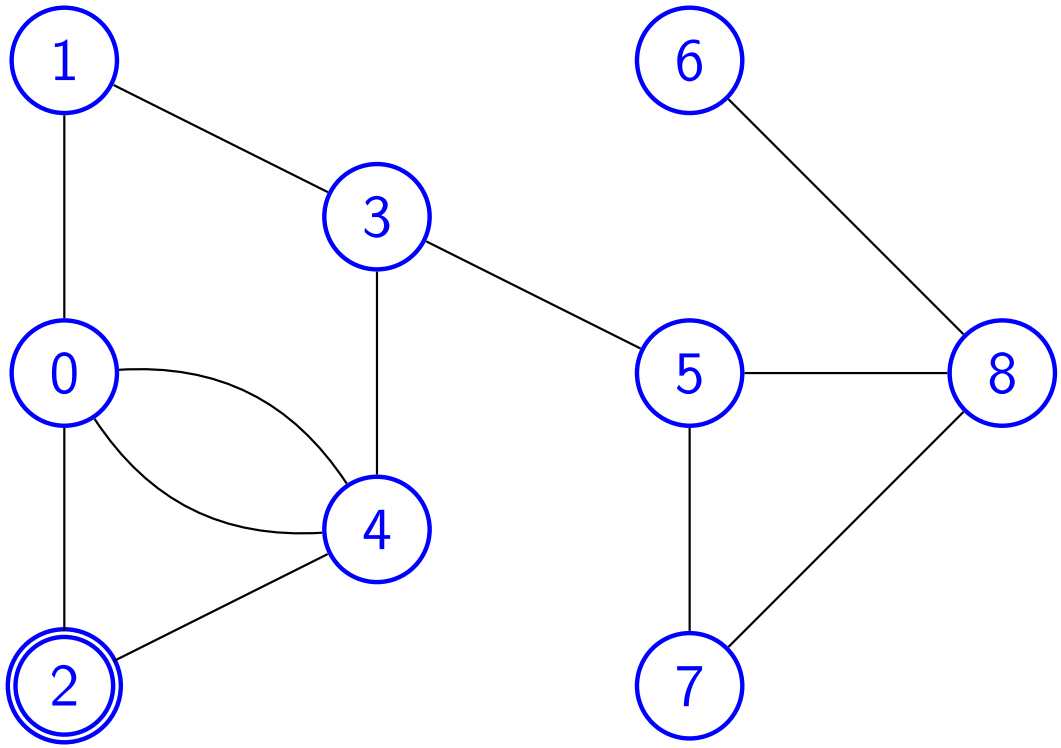

```
s = [
 6,
 2,
 4
]
```

DFS



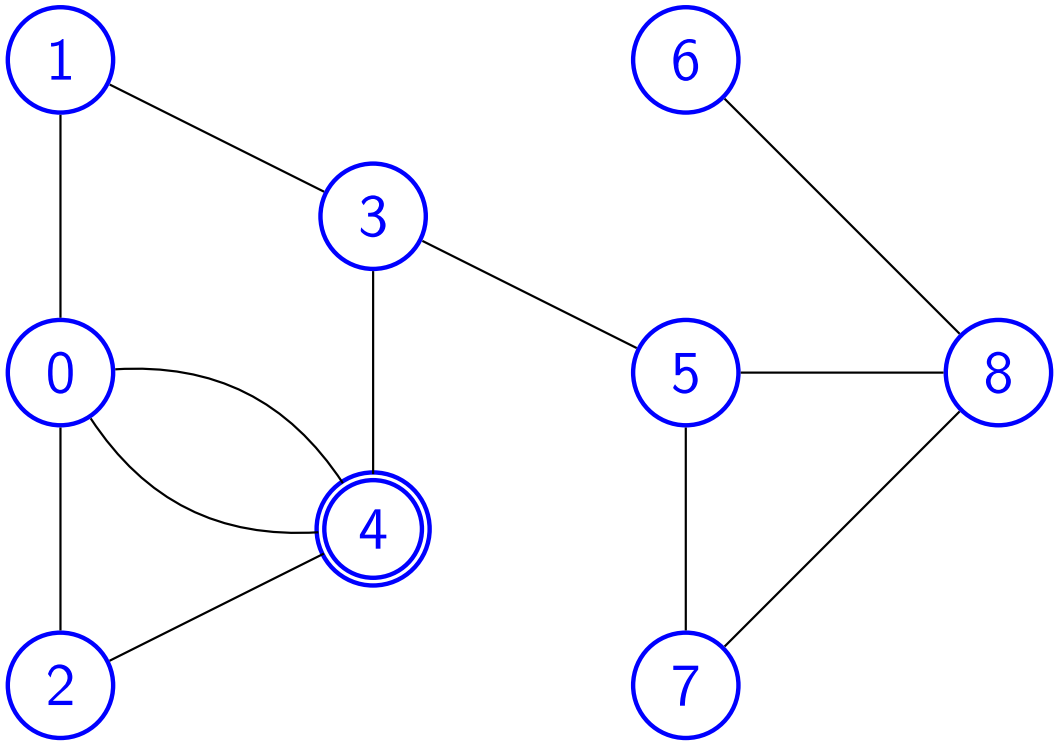

```
s = [
 2,
 4
]
```

DFS



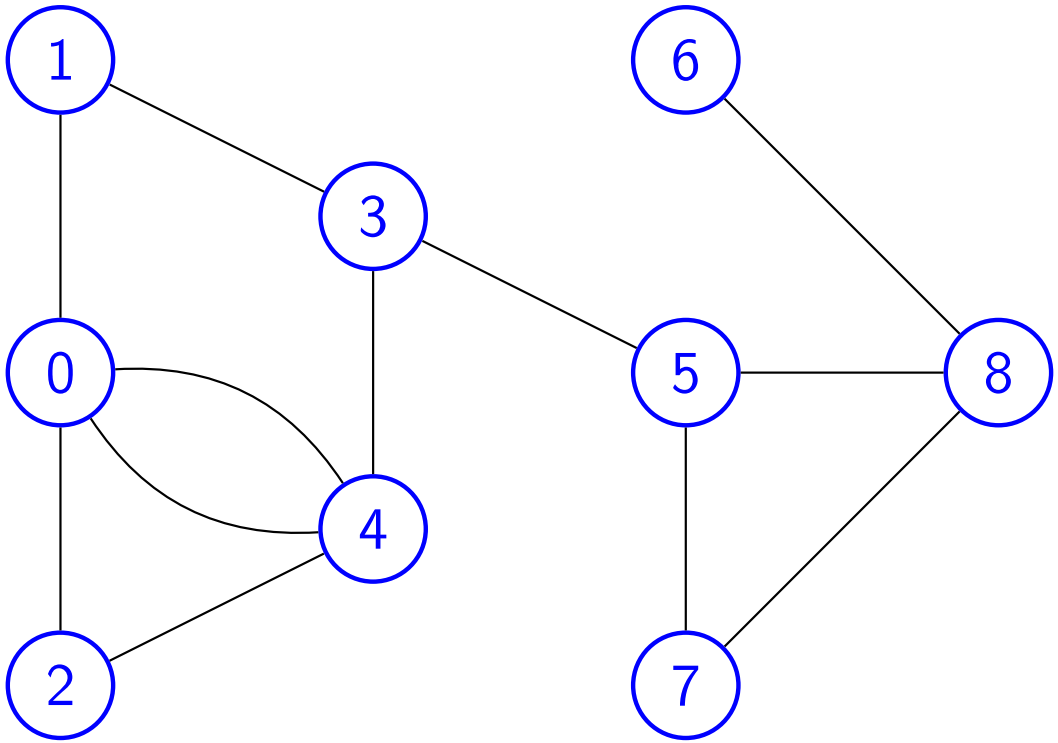

```
s = [
  4
]
```

DFS




```
s = [
]
```

DFS




```
s = [
]
```

DFS útfærsla með nágrannalista

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

int main() {
    int n, m, a, b;
    cout << "Gefðu upp fjölda hnúta og leggja\n";
    cin >> n >> m;
    cout << "Gefðu upp m leggi (0-index)\n";
    vvi g(n, vi());
    for(int i = 0; i < m; ++i) {
        cin >> a >> b;
        g[a].push_back(b), g[b].push_back(a);
    }
    stack<int> q; vector<bool> d(n, false);
    q.push(0); d[0] = true;
    cout << "Í samhengisþætti 0 fundust:\n";
    while(!q.empty()) {
        int cur = q.top(); q.pop();
        cout << cur << '\n';
        for(int x : g[cur]) {
            if(d[x]) continue;
            d[x] = true; q.push(x); }}}
```

Samhengisþættir

- Við sjáum að þetta fer bara í gegnum einn samhengisþátt netsins.

Samhengisþættir

- Við sjáum að þetta fer bara í gegnum einn samhengisþátt netsins.
- Ef netið er ekki samanhangandi (s.s. hefur fleiri en einn samhengisþátt) þá þarf að halda utan um samhengisþættina og leita í hverjum fyrir sig.

Samhengisþættir

- Við sjáum að þetta fer bara í gegnum einn samhengisþátt netsins.
- Ef netið er ekki samanhangandi (s.s. hefur fleiri en einn samhengisþátt) þá þarf að halda utan um samhengisþættina og leita í hverjum fyrir sig.
- Gera má það með union-find með því að join-a öllum aðlægum hnútum og gefur þá union-find-ið í hvaða samhengisþætti hver hnútur er.

Samhengisþættir

- Við sjáum að þetta fer bara í gegnum einn samhengisþátt netsins.
- Ef netið er ekki samanhangandi (s.s. hefur fleiri en einn samhengisþátt) þá þarf að halda utan um samhengisþættina og leita í hverjum fyrir sig.
- Gera má það með union-find með því að join-a öllum aðlægum hnútum og gefur þá union-find-ið í hvaða samhengisþætti hver hnútur er.
- Sjáum meira um þetta þegar við tölum um tengihnúta og brýr, en fyrst stutt innskot um tvíhlutanet.

Efnisyfirlit

Litun á neti

- Þegar talað er um litun á neti er átt við að lita hnúta netsins þannig að engir nágrannar hafi sama lit.

Litun á neti

- Þegar talað er um litun á neti er átt við að lita hnúta netsins þannig að engir nágrannar hafi sama lit.
- Að finna lágmarksfjölda lita í neti er NP-vandamál svo við munum ekki skoða það hér. Jafnvel bara að skoða hvort lita megi net með 3 litum er NP-vandamál.

Litun á neti

- Þegar talað er um litun á neti er átt við að lita hnúta netsins þannig að engir nágrannar hafi sama lit.
- Að finna lágmarksfjölda lita í neti er NP-vandamál svo við munum ekki skoða það hér. Jafnvel bara að skoða hvort lita megi net með 3 litum er NP-vandamál.
- Hvort lita megi net með einum lit er ekki spennandi því það er hægt þ.þ.a.a. netið hafi enga leggi. Því er tilvikið fyrir tvo liti það sem við munum skoða. Hvenær má lita net með tveimur litum?

Litun á neti

- Þegar talað er um litun á neti er átt við að lita hnúta netsins þannig að engir nágrannar hafi sama lit.
- Að finna lágmarksfjölda lita í neti er NP-vandamál svo við munum ekki skoða það hér. Jafnvel bara að skoða hvort lita megi net með 3 litum er NP-vandamál.
- Hvort lita megi net með einum lit er ekki spennandi því það er hægt þ.þ.a.a. netið hafi enga leggi. Því er tilvikið fyrir tvo liti það sem við munum skoða. Hvenær má lita net með tveimur litum?
- Þegar þetta er hægt má skipta hnútunum í tvo hópa svo allir leggirnir liggji milli hópa en ekki milli tveggja hnúta í sama hópnum.

Litun á neti

- Þegar talað er um litun á neti er átt við að lita hnúta netsins þannig að engir nágrannar hafi sama lit.
- Að finna lágmarksfjölda lita í neti er NP-vandamál svo við munum ekki skoða það hér. Jafnvel bara að skoða hvort lita megi net með 3 litum er NP-vandamál.
- Hvort lita megi net með einum lit er ekki spennandi því það er hægt þ.þ.a.a. netið hafi enga leggi. Því er tilvikið fyrir tvo liti það sem við munum skoða. Hvenær má lita net með tveimur litum?
- Þegar þetta er hægt má skipta hnútunum í tvo hópa svo allir leggirnir liggji milli hópa en ekki milli tveggja hnúta í sama hópnum.
- Net þar sem þetta er hægt kallast tvíhlutanet (bipartite graph).

Tvíhlutanet

- Tvíhlutanet hafa marga þægilega eiginleika. Þið getið til dæmis reynt að sannfæra ykkur um að tvíhlutanet hafi engar rásir af odda lengd.

Tvíhlutanet

- Tvíhlutanet hafa marga þægilega eiginleika. Þið getið til dæmis reynt að sannfæra ykkur um að tvíhlutanet hafi engar rásir af odda lengd.
- En hvernig ákvörðum við hvort net sé tvíhlutanet?

Tvíhlutanet

- Tvíhlutanet hafa marga þægilega eiginleika. Þið getið til dæmis reynt að sannfæra ykkur um að tvíhlutanet hafi engar rásir af odda lengd.
- En hvernig ákvörðum við hvort net sé tvíhlutanet?
- Við getum reynt að lita það í tveimur litum gráðugt og séð hvort við litum okkur út í horn!

Tvíhlutanet

- Tvíhlutanet hafa marga þægilega eiginleika. Þið getið til dæmis reynt að sannfæra ykkur um að tvíhlutanet hafi engar rásir af odda lengd.
- En hvernig ákvörðum við hvort net sé tvíhlutanet?
- Við getum reynt að lita það í tveimur litum gráðugt og séð hvort við litum okkur út í horn!
- Litum upphafspunkt bláan, nágranna hans rauða, nágranna þeirra bláa o.s.frv. og sjáum hvort það gangi upp. Göngum s.s. í gegnum netið með BFS og litum jafnóðum. Gera þarf þetta fyrir hvern samhengisþátt og er netið tvíhlutanet ef sérhver samhengisþáttur þess er það.

Tvíhlutanet

- Tvíhlutanet hafa marga þægilega eiginleika. Þið getið til dæmis reynt að sannfæra ykkur um að tvíhlutanet hafi engar rásir af odda lengd.
- En hvernig ákvörðum við hvort net sé tvíhlutanet?
- Við getum reynt að lita það í tveimur litum gráðugt og séð hvort við litum okkur út í horn!
- Litum upphafspunkt bláan, nágranna hans rauða, nágranna þeirra bláa o.s.frv. og sjáum hvort það gangi upp. Göngum s.s. í gegnum netið með BFS og litum jafnóðum. Gera þarf þetta fyrir hvern samhengisþátt og er netið tvíhlutanet ef sérhver samhengisþáttur þess er það.
- Sjáum hér útfærslu fyrir einn samhengisþátt (eða samanhangandi net)

Tvíhlutatékk

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

bool bipartite(vvi& g) {
    queue<int> q;
    q.push(0);
    vi color(g.size(), -1);
    color[0] = 0;
    while(!q.empty()) {
        int cur = q.front();
        q.pop();
        for(int x : g[cur]) {
            if(color[x] == -1) {
                color[x] = 1 - color[cur];
                q.push(x);
            } else if(color[x] == color[cur]) {
                return false;
            }
        }
    }
    return true;
}
```

Efnisyfirlit

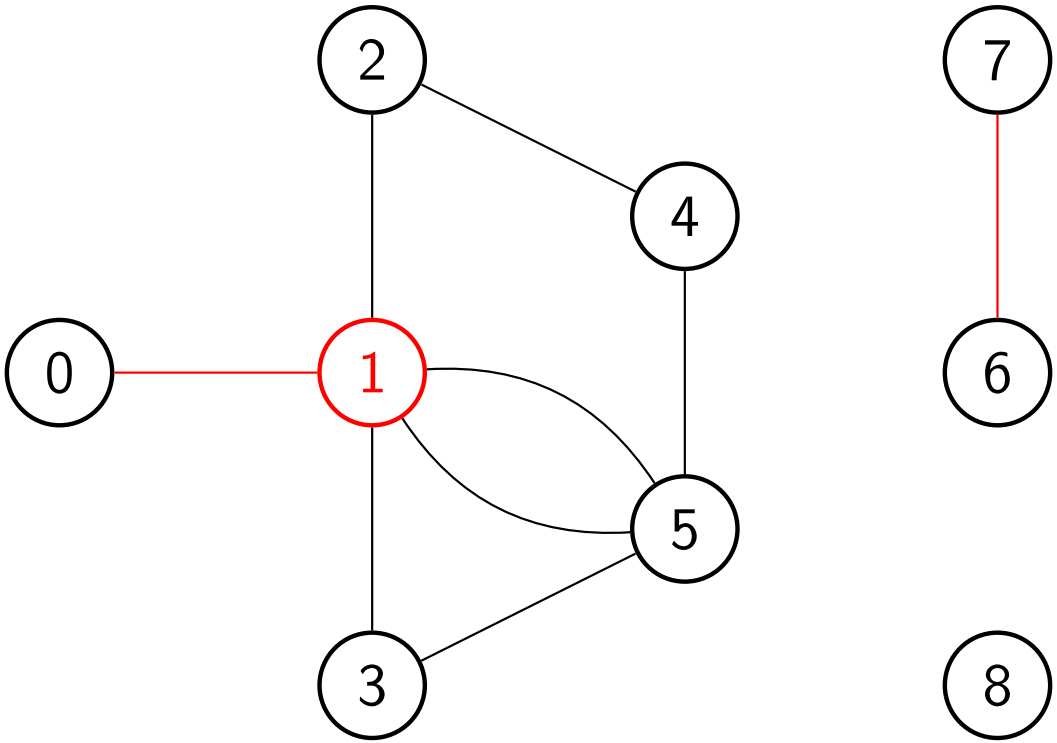
Tengihnútar og brýr

- Tengihnútar (cut points) eru hnútar sem hafa þann eiginleika að ef þeir eru fjarlægðir fjölgar samhengisþáttum netsins. Brýr (bridges) eru leggir með sama eiginleika. Þeir eru því hnútar/leggir sem eru nauðsynlegir til að halda netinu samanhangandi.

Tengihnútar og brýr

- Tengihnútar (cut points) eru hnútar sem hafa þann eiginleika að ef þeir eru fjarlægðir fjölgar samhengisþáttum netsins. Brýr (bridges) eru leggir með sama eiginleika. Þeir eru því hnútar/leggir sem eru nauðsynlegir til að halda netinu samanhangandi.
- Ef maður horfir á mynd af neti er yfirleitt frekar auðvelt að 'sjá út' hvaða hnútar eru tengihnútar og hvaða leggir eru brýr. Skoðum dæmi um þetta.

Dæmi um tengihnúta og brýr



Að finna tengihnúta og brýr

- En hvernig má finna þetta með tölvu?

Að finna tengihnúta og brýr

- En hvernig má finna þetta með tölvu?
- Kemur í ljós að finna má þetta allt með því að fara í gegnum netið með einu DFS!

Að finna tengihnúta og brýr

- En hvernig má finna þetta með tölvu?
- Kemur í ljós að finna má þetta allt með því að fara í gegnum netið með einu DFS!
- Þegar við löbbum í gegn með DFS geymum við fyrir hvern hnút tvær tölur, oftast kallaðar `low` og `num`.

Að finna tengihnúta og brýr

- En hvernig má finna þetta með tölvu?
- Kemur í ljós að finna má þetta allt með því að fara í gegnum netið með einu DFS!
- Þegar við löbbum í gegn með DFS geymum við fyrir hvern hnút tvær tölur, oftast kallaðar `low` og `num`.
- `num` geymir hvað það tók okkur mörg skref að komast í þann hnút með DFS leitinni og `low` segir hvert er lægsta `num` gildi sem er hægt að komast í úr þessum hnút án þess að endurtaka leggi sem við notuðum í DFS leitinni sjálfri.

Að finna tengihnúta og brýr

- En hvernig má finna þetta með tölvu?
- Kemur í ljós að finna má þetta allt með því að fara í gegnum netið með einu DFS!
- Þegar við löbbum í gegn með DFS geymum við fyrir hvern hnút tvær tölur, oftast kallaðar `low` og `num`.
- `num` geymir hvað það tók okkur mörg skref að komast í þann hnút með DFS leitinni og `low` segir hvert er lægsta `num` gildi sem er hægt að komast í úr þessum hnút án þess að endurtaka leggi sem við notuðum í DFS leitinni sjálfri.
- Ef við förum í hnút v á eftir hnút u og við endum með að `low` fyrir v sé stærra en `num` fyrir u þá er leggurinn frá u til v brú því við komumst bara úr v í u með þeim eina legg.

Að finna tengihnúta og brýr

- En hvernig má finna þetta með tölvu?
- Kemur í ljós að finna má þetta allt með því að fara í gegnum netið með einu DFS!
- Þegar við löbbum í gegn með DFS geymum við fyrir hvern hnút tvær tölur, oftast kallaðar `low` og `num`.
- `num` geymir hvað það tók okkur mörg skref að komast í þann hnút með DFS leitinni og `low` segir hvert er lægsta `num` gildi sem er hægt að komast í úr þessum hnút án þess að endurtaka leggi sem við notuðum í DFS leitinni sjálfri.
- Ef við förum í hnút v á eftir hnút u og við endum með að `low` fyrir v sé stærra en `num` fyrir u þá er leggurinn frá u til v brú því við komumst bara úr v í u með þeim eina legg.
- Ef við förum í hnút v á eftir hnút u og við endum með að `low` fyrir v sé stærra en eða jafnt `num` fyrir u þá er u tengihnútur því þá er enginn önnur leið til baka úr v sem fer ekki um u.

Að reikna `num` og `low`

- En hvernig reiknum við þessar tölur?

Að reikna `num` og `low`

- En hvernig reiknum við þessar tölur?
- Í byrjun förum við bara í gegnum netið með DFS og setjum `num` útfrá því. Við látum í byrjun bara `low` vera jafnt `num` í hverjum hnút.

Að reikna `num` og `low`

- En hvernig reiknum við þessar tölur?
- Í byrjun förum við bara í gegnum netið með DFS og setjum `num` útfrá því. Við látum í byrjun bara `low` vera jafnt `num` í hverjum hnút.
- Svo ef við rekumst á legg sem við notuðum ekki í DFS-leitinni til að komast í hnútinn sem við erum stödd í uppfærum við `low` fyrir alla hnútana sem komast í þann legg.

Að reikna `num` og `low`

- En hvernig reiknum við þessar tölur?
- Í byrjun förum við bara í gegnum netið með DFS og setjum `num` útfrá því. Við látum í byrjun bara `low` vera jafnt `num` í hverjum hnút.
- Svo ef við rekumst á legg sem við notuðum ekki í DFS-leitinni til að komast í hnútinn sem við erum stödd í uppfærum við `low` fyrir alla hnútana sem komast í þann legg.
- Þetta virkar fyrir alla hnútana nema fyrir þann sem við byrjum í, en við tékkum bara á honum sérstaklega.

Að reikna `num` og `low`

- En hvernig reiknum við þessar tölur?
- Í byrjun förum við bara í gegnum netið með DFS og setjum `num` útfrá því. Við látum í byrjun bara `low` vera jafnt `num` í hverjum hnút.
- Svo ef við rekumst á legg sem við notuðum ekki í DFS-leitinni til að komast í hnútinn sem við erum stödd í uppfærum við `low` fyrir alla hnútana sem komast í þann legg.
- Þetta virkar fyrir alla hnútana nema fyrir þann sem við byrjum í, en við tékkum bara á honum sérstaklega.
- Skoðum nú útfærslu á þessu

Tengihnúta- og brúarleit útfærð

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

vi low, num;
int curnum;

void dfs(vvi &g, vi &cp, vii &bri, int u, int p) {
    low[u] = num[u] = curnum++;
    int cnt = 0; bool found = false;
    for(int v : g[u]) {
        if(num[v] == -1) {
            dfs(g, cp, bri, v, u);
            low[u] = min(low[u], low[v]);
            cnt++; found |= low[v] >= num[u];
            if(low[v] > num[u]) bri.push_back(ii(u, v));
        } else if(p != v) low[u] = min(low[u], num[v]); }
  if(found && (p != -1 || cnt > 1)) cp.push_back(u); }

pair<vi,vii> cutpointsbridges(vvi &g) {
    vi cp; vii bri;
    num = vi(g.size(), -1);
    low.resize(g.size());
    curnum = 0;
    for(int i = 0; i < g.size(); ++i) {
        if(num[i] == -1) {
            dfs(g, cp, bri, i, -1);
        }
    }
    return pair<vi,vii>(cp, bri); }
```

Efnisyfirlit

Fleiri NP-dæmi

- Gott er að vita hvaða dæmi eru tölvunarfræðilega erfið að leysa því þá veit maður að ef þau koma upp í keppni er eitthvað skrýtið í gangi. Yfirleitt þýðir það að einhversstaðar í dæminu er eitthvað auka skilyrði gefið sem gerir dæmið leysanlegt. Gott er þá að þekkja venjulegu útgáfur dæmanna til að geta séð hvaða skilyrði eru mikilvæg.

Fleiri NP-dæmi

- Gott er að vita hvaða dæmi eru tölvunarfræðilega erfið að leysa því þá veit maður að ef þau koma upp í keppni er eitthvað skrýtið í gangi. Yfirleitt þýðir það að einhversstaðar í dæminu er eitthvað auka skilyrði gefið sem gerir dæmið leysanlegt. Gott er þá að þekkja venjulegu útgáfur dæmanna til að geta séð hvaða skilyrði eru mikilvæg.
- Erfitt er að finna stærsta mengi hnúta sem eru allir tengdir með legg tveir og tveir (maximum clique problem)

Fleiri NP-dæmi

- Gott er að vita hvaða dæmi eru tölvunarfræðilega erfið að leysa því þá veit maður að ef þau koma upp í keppni er eitthvað skrýtið í gangi. Yfirleitt þýðir það að einhversstaðar í dæminu er eitthvað auka skilyrði gefið sem gerir dæmið leysanlegt. Gott er þá að þekkja venjulegu útgáfur dæmanna til að geta séð hvaða skilyrði eru mikilvæg.
- Erfitt er að finna stærsta mengi hnúta sem eru allir tengdir með legg tveir og tveir (maximum clique problem)
- Erfitt er að finna stærsta mengi hnúta sem hafa enga leggi á milli sín (maximum independent set problem), hins vegar má leysa það í tvíhlutanetum frekar hratt (sjá bók)

Fleiri NP-dæmi

- Gott er að vita hvaða dæmi eru tölvunarfræðilega erfið að leysa því þá veit maður að ef þau koma upp í keppni er eitthvað skrýtið í gangi. Yfirleitt þýðir það að einhversstaðar í dæminu er eitthvað auka skilyrði gefið sem gerir dæmið leysanlegt. Gott er þá að þekkja venjulegu útgáfur dæmanna til að geta séð hvaða skilyrði eru mikilvæg.
- Erfitt er að finna stærsta mengi hnúta sem eru allir tengdir með legg tveir og tveir (maximum clique problem)
- Erfitt er að finna stærsta mengi hnúta sem hafa enga leggi á milli sín (maximum independent set problem), hins vegar má leysa það í tvíhlutanetum frekar hratt (sjá bók)
- Erfitt er að finna minnsta mengi hnúta sem eru saman aðlægir öllum leggjum netsins (minimum vertex cover problem), hins vegar má leysa það í tvíhlutanetum frekar hratt (sjá bók)

Fleiri NP-dæmi

- Gott er að vita hvaða dæmi eru tölvunarfræðilega erfið að leysa því þá veit maður að ef þau koma upp í keppni er eitthvað skrýtið í gangi. Yfirleitt þýðir það að einhversstaðar í dæminu er eitthvað auka skilyrði gefið sem gerir dæmið leysanlegt. Gott er þá að þekkja venjulegu útgáfur dæmanna til að geta séð hvaða skilyrði eru mikilvæg.
- Erfitt er að finna stærsta mengi hnúta sem eru allir tengdir með legg tveir og tveir (maximum clique problem)
- Erfitt er að finna stærsta mengi hnúta sem hafa enga leggi á milli sín (maximum independent set problem), hins vegar má leysa það í tvíhlutanetum frekar hratt (sjá bók)
- Erfitt er að finna minnsta mengi hnúta sem eru saman aðlægir öllum leggjum netsins (minimum vertex cover problem), hins vegar má leysa það í tvíhlutanetum frekar hratt (sjá bók)
- Erfitt er að finna hvort net hafi rás/veg sem fer í gegnum hvern hnút nákvæmlega einu sinni. Hins vegar er töluvert auðveldara að finna hvort net hafi rás/veg sem fer í gegnum hvern legg nákvæmlega einu sinni (sjá bók)