# Inngangur að netafræði

Bergur Snorrason

18. febrúar 2021

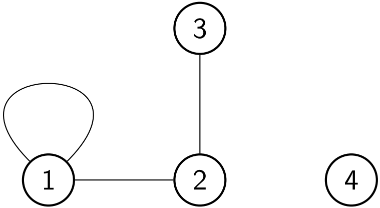
# Net

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

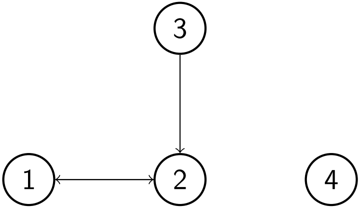
$$(u, v) \in E \Rightarrow (v, u) \in E,$$

  þá segjum við að netið sé *óstefnt*.
- Net sem er ekki óstefnt kallast *stefnt*.
- Við segjum að nóðan $v$ sé *nágranni* nóðunnar $u$ ef $(u, v)$ er í $E$.
- Við segjum að nóður $u$ og $v$ í óstefndu neti séu *nágrannar* ef $(u, v)$ er í $E$.
- Við segjum einnig að það liggi leggur á milli $u$ og $v$.

- Oft hjálpar að skilja tiltekið net með því að teikna það.
- Við byrjum á að teikna punkta fyrir nóðurnar.
- 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 nóðu $u$ til nóðu $v$.




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




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

- Leggir af gerðinni $(u, u)$ kallast *lykkjur* (ástæða nafngiftarinnar sést að fyrri myndinni).
- Net án leggja kallast *einfalt*.
- Í umfjöllun okkar gerum við ráð fyrir að öll net séu einföld nema annað sé tekið fram.

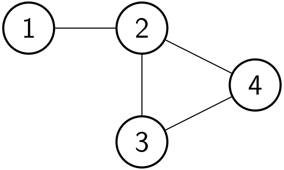
- Runa nóða $v_1, v_2, ..., v_n$ kallast *vegur* ef $(v_j, v_{j+1}) \in E$, fyrir $j = 1, 2, ..., n-1$.
- Vegur kallast *rás* ef $v_1 = v_n$.
- Vegur kallast *einfaldur* ef engar tvær nóður í $v_1, v_2, ..., v_n$ eru eins.
- Rás kallast *einföld* ef engar tvær nóður í $v_1, v_2, ..., v_{n-1}$ eru eins.
- Við segjum að vegurinn $v_1, v_2, ..., v_n$ *liggi á milli nóðanna* $v_1$ *og* $v_n$.
- Óstefnt net er sagt vera *samanhangandi* ef til er vegur milli sérhverja tveggja nóða.
- Óstefnt net er sagt vera *tré* ef það er samanhangandi og inniheldur enga rás.

# Framsetning neta í tölvum

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

# Leggjalisti

- Látum $G = (E, V)$ tákna netið okkar.
- Þar sem $E$ er endanlegt megum við gera ráð fyrir að $E = \{1, 2, ..., n\}$, þar sem $n$ er fjöldi nóða í $G$.
- Látum $m$ vera fjölda leggja í $G$.
- Listi af tvenndum sem inniheldur nákvæmlega sömu stök og $V$ 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 leggja lista.
- Í óstefndum netum er hver leggur tvítekinn í $E$ og við leyfum okkur að sleppa annari endurtekningunni í listanum.

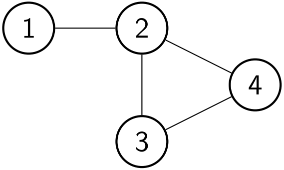



$$
\begin{aligned}
L = [\\
&(1,2),\\
&(2,3),\\
&(2,4),\\
&(3,4)\\
]
\end{aligned}
$$

- Helsti galli leggjalistans er að það tekur $\mathcal{O}(\, m\,)$ tíma að ákvarða hvort nóður séu nágrannar eða finna nágranna tilteknar nóðu.

# Nágrannafylki

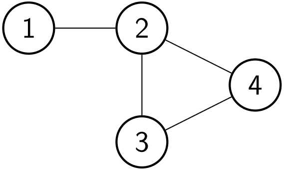
- Látum $A$ vera $n \times n$ fylki þannig að $A_{uv} = 1$ ef $(u, v)$ er í $E$, en $A_{uv} = 0$ annars.
- Við köllum $A$ nágrannafylki netsins $G$.
- Takið eftir að það tekur $\mathcal{O}(n^2)$ tíma að upphafsstilla $A$.
- Svo þessi aðferð er alferið of hæg ef, til dæmis, $n = 10^5$ (sem er oft raunin).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




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

# Nágrannalistar

- Látum nú $V$ tákna lista af $n$ listum.
- Táknum $j$-ta lista $V$ með $V_j$.
- Látum nú $V_u$ innihalda alla nágranna nóðunnar $u$ í netinu $G$, án endurtekningar.
- Við köllum $V$ *nágrannalista* (fleirtölu) netsins $G$ og $V_u$ *nágrannalista* (eintölu) nóðunnar $u$ í netinu $G$.
- Helsti kostur nágrannalistanna er að við getum skoðað alla nágranna tiltekinnar nóðu án þess að skoða neitt annað.
- Við getum því ítrað í gegnum alla nágranna alla nóðanna í $\mathcal{O}(\, m \,)$ tíma, óháð röð nóðanna.
- Þetta kemur að góðum notum þegar við erum að ferðast í gegnum netið.




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

- Eins minnst var á áðan eru net yfirleitt 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 í viðeigandi nóðum í tilheyrandi nágrannalista.
- Ef leggur $(u, v)$ er í leggjalista stefnts nets þá bætum við $v$ við $V_u$.
- Ef leggur $(u, v)$ er í leggjalista óstefnts nets þá bætum við $v$ við $V_u$ og $u$ við $V_v$.

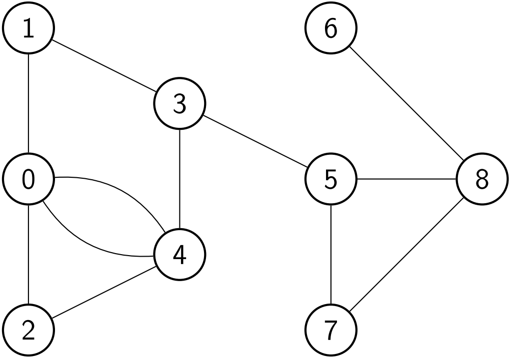
```cpp
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;
typedef vector<vi> vvi;

// Fyrsta lína inntaksins eru tvær heiltölur, fjöldi nóða og fjöldi leggja.
// Síðan koma m línur sem svara til leggjalistans.
int main()
{
    int i, j, n, m;
    cin >> n >> m;
    vvi g(n);
    for (i = 0; i < m; i++)
    {
        int x, y;
        cin >> x >> y;
        x--, y--;
        g[x].push_back(y);
        g[y].push_back(x); // Sleppa þesari línu ef netið er stefnt.
    }
    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");
    }
}
```

- Hvernig ítrum við í gegnum allar nóður netsins.
- Þetta má að sjálfsögðu gera á marga vegu, en algengt er að notast við annað að tvennu:
    - Dýptarleit (e. deapth-first search).
    - Breiddarleit (e. breadth-first search).
- Báðar byggja á því að byrja í einhverri nóðu og heimsækja svo nágranna hennar.

# Dýptarleit

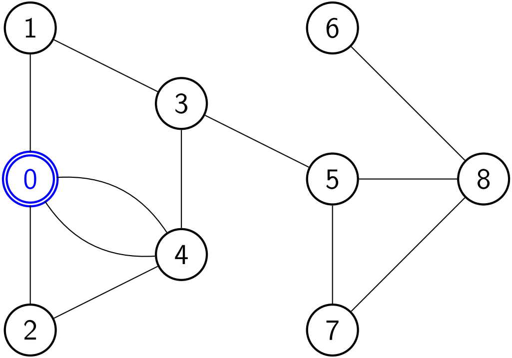
- Dýptarleit byrjar í einhverri nóðu.
- Í hverju skrefi heimsækir leitin nóðu sem hefur ekki verið heimsótt áður í leitinni.
- Ef allir nágrannar hafa verið heimsóttir þá er farið til baka og nágrannar síðustu nóðu eru skoðaðir.
- Tökum dæmi.

# Dýptarleit



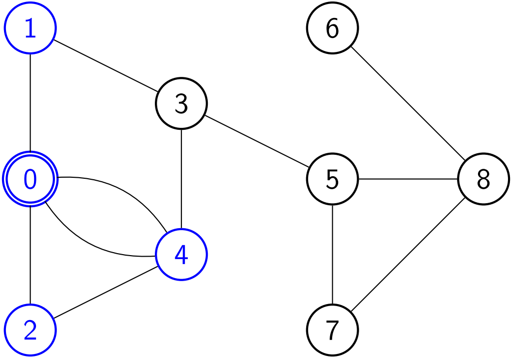

```
s = [
  0
]
```

# Dýptarleit



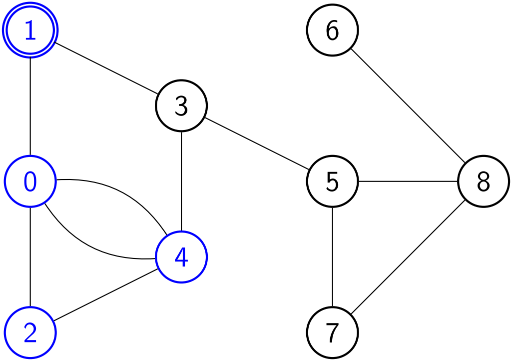

```
s = [
]
```

# Dýptarleit



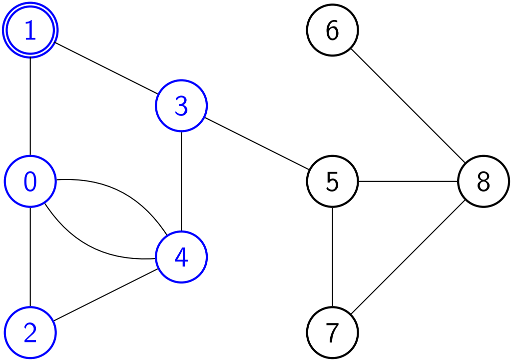

```
s = [
 1,
 2,
 4
]
```

# Dýptarleit



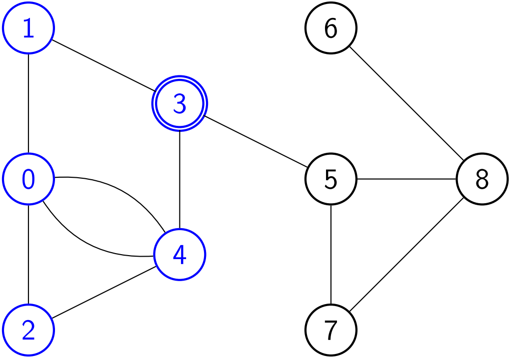

```
s = [
 2,
 4
]
```

# Dýptarleit



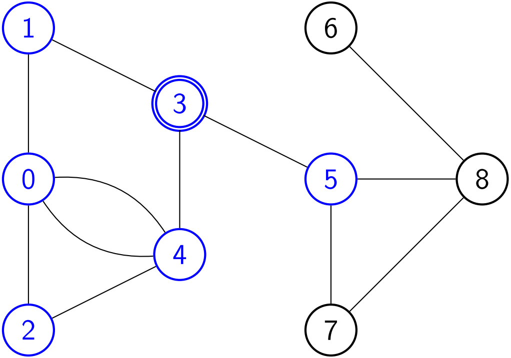

```
s = [
 3,
 2,
 4
]
```

# Dýptarleit



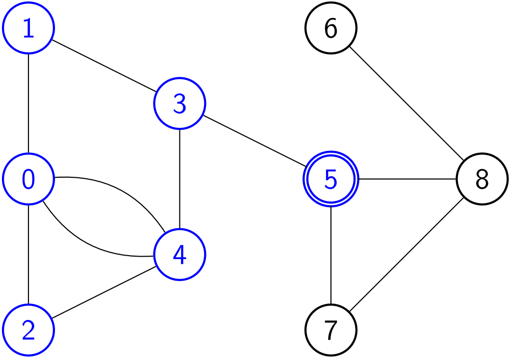

```
s = [
 2,
 4
]
```

# Dýptarleit



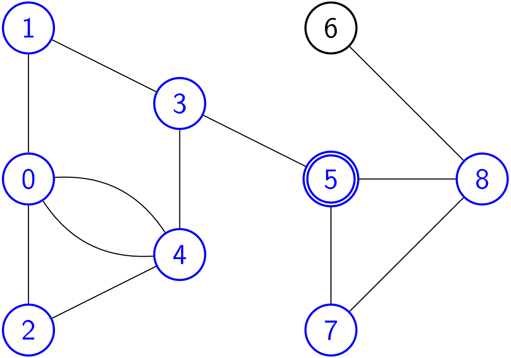

```
s = [
  5,
  2,
  4
]
```

# Dýptarleit



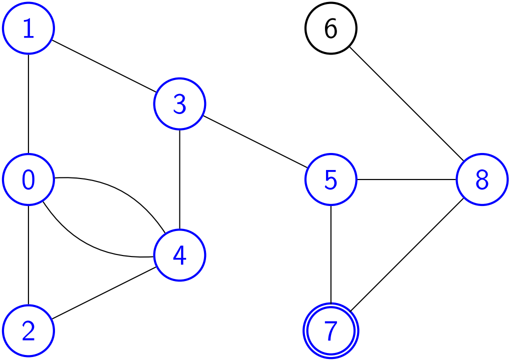

```
s = [
 2,
 4
]
```

# Dýptarleit



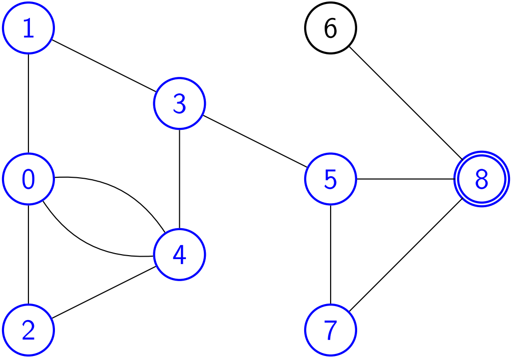

```
s = [
 7,
 8,
 2,
 4
]
```

# Dýptarleit



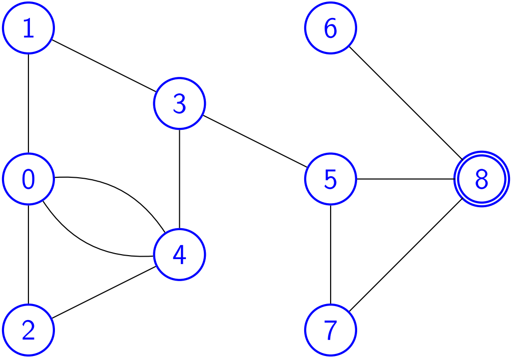

```
s = [
 8,
 2,
 4
]
```

# Dýptarleit



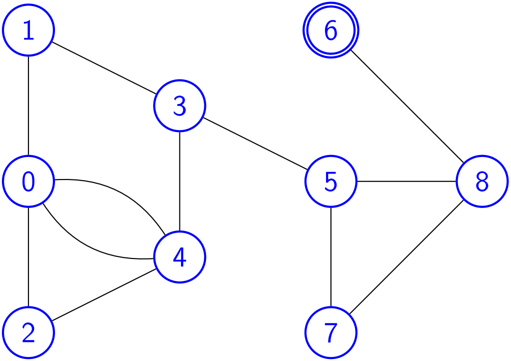

```
s = [
 2,
 4
]
```

# Dýptarleit



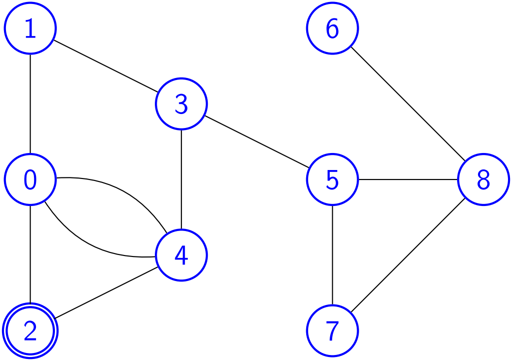

```
s = [
 6,
 2,
 4
]
```

# Dýptarleit



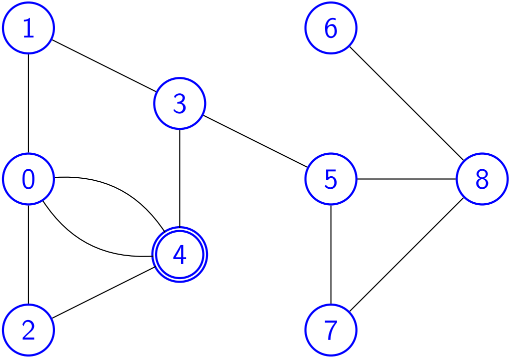

```
s = [
 2,
 4
]
```

# Dýptarleit



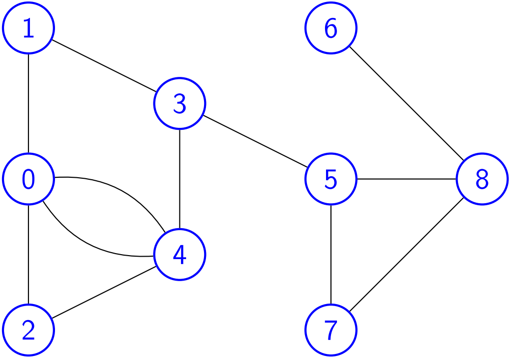

```
s = [
 4
]
```

# Dýptarleit



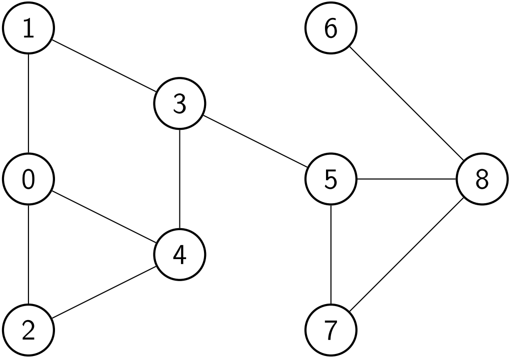

```
s = [
]
```

# Dýptarleit



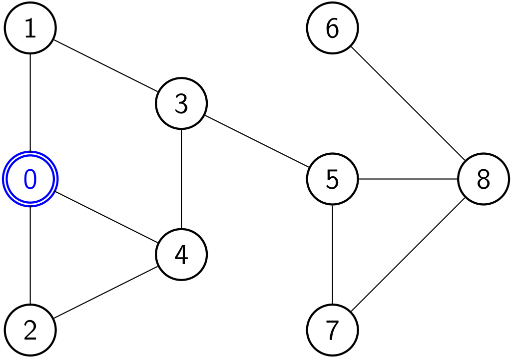

```
s = [
]
```

# Breiddarleit



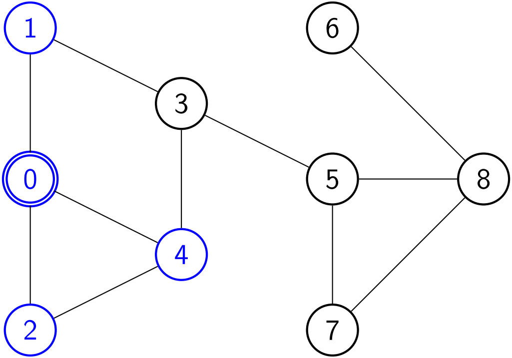

```
q = [
 0
]
```

# Breiddarleit



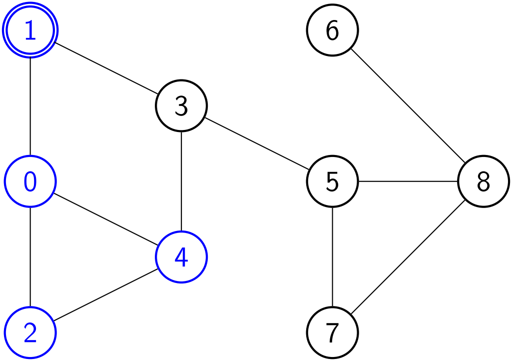

```
q = [
]
```

# Breiddarleit



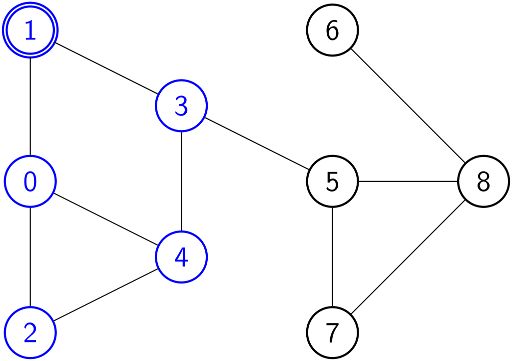

```
q = [
  1,
  2,
  4
]
```

# Breiddarleit



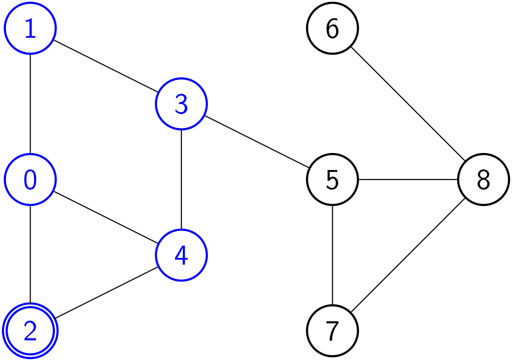

```
q = [
 2,
 4
]
```

# Breiddarleit



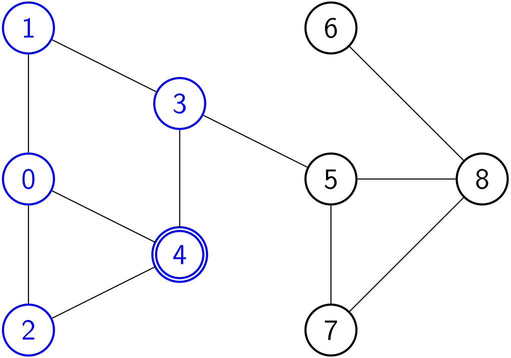

```
q = [
 2,
 4,
 3
]
```

# Breiddarleit



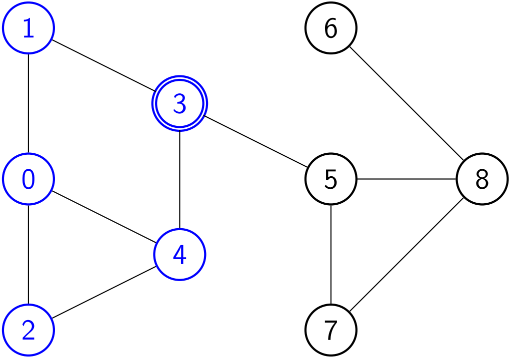

```
q = [
 4,
 3
]
```

# Breiddarleit



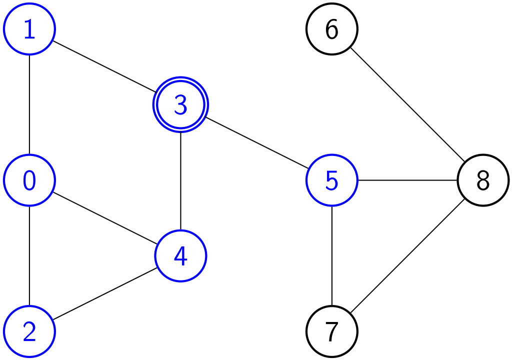

```
q = [
 3
]
```

# Breiddarleit



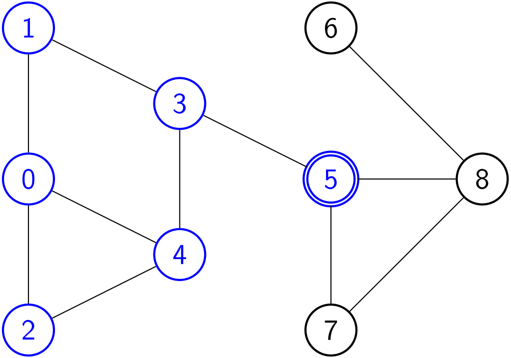

```
q = [
]
```

# Breiddarleit



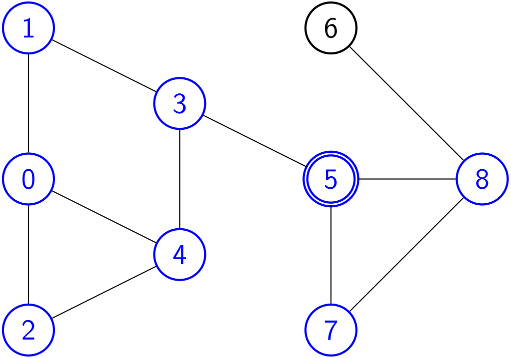

```
q = [
 5
]
```

# Breiddarleit



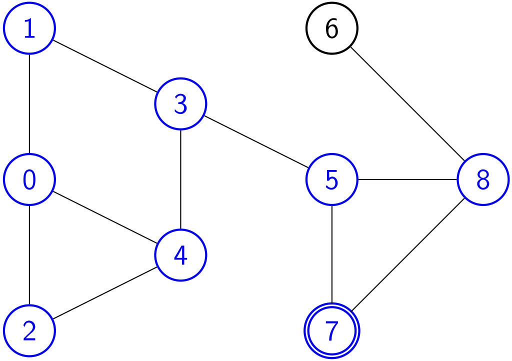

```
q = [
]
```

# Breiddarleit



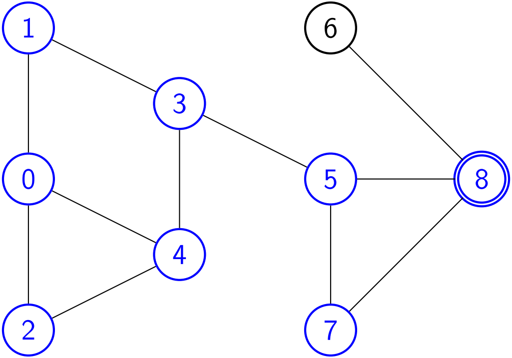

```
q = [
 7,
 8
]
```

# Breiddarleit



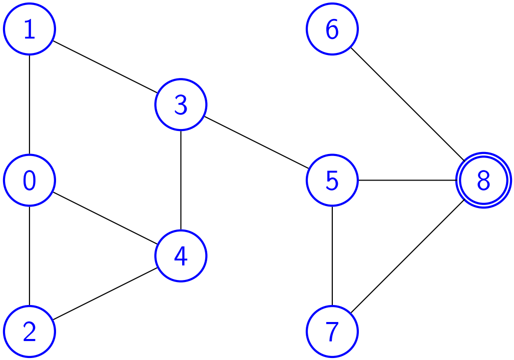

```
q = [
 8
]
```

# Breiddarleit



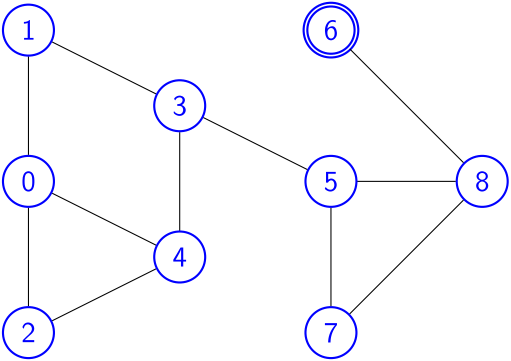

```
q = [
]
```

# Breiddarleit



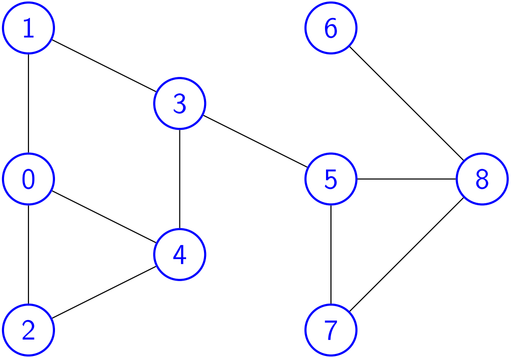

```
q = [
 6
]
```

# Breiddarleit




```
q = [
]
```

# Breiddarleit




```
q = [
]
```