

Dýptarleit og breiddarleit

Bergur Snorrason

20. febrúar 2021

- ▶ Hvernig ítrum við í gegnum allar nóður stefnds nets.

- ▶ Hvernig ítrum við í gegnum allar nóður stefnds nets.
- ▶ Þetta má að sjálfsgöðu gera á marga vegu, en algengt er að notast við annað að tvennu:

- ▶ Hvernig ítrum við í gegnum allar nóður stefnds nets.
- ▶ Þetta má að sjálfsögðu gera á marga vegu, en algengt er að notast við annað að tvennu:
 - ▶ *Dýptarleit* (e. *deapth-first search*).

- ▶ Hvernig ítrum við í gegnum allar nóður stefnds nets.
- ▶ 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*).

- ▶ Hvernig ítrum við í gegnum allar nóður stefnds nets.
- ▶ 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.

Dýptarleit

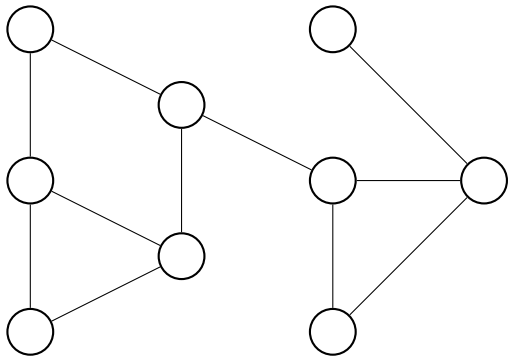
- ▶ Dýptarleit byrjar í einhverri nóðu.
- ▶ Í hverju skrefi heimsækir leitin nóðu sem hefur ekki verið heimsótt áður í leitinni.

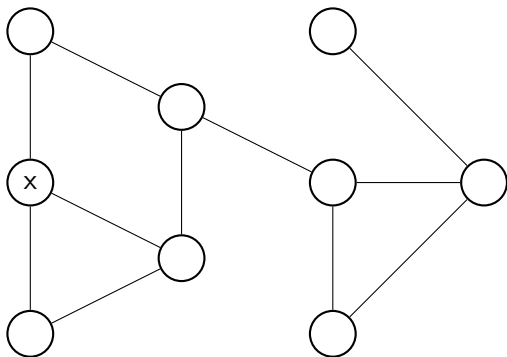
Dýptarleit

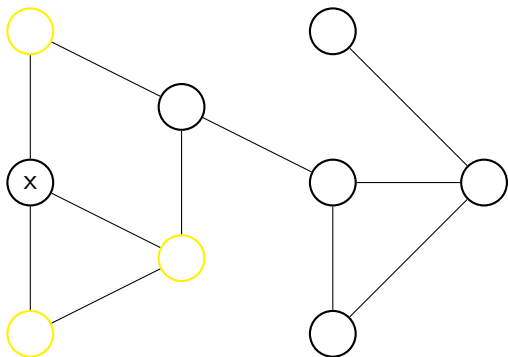
- ▶ Dýptarleit byrjar í einhverri nóðu.
- ▶ Í hverju skrefi heimsækir leitin nóðu sem hefur ekki verið heimsótt áður í leitinni.
- ▶ Ef allir nágrannar hafa verið heimsóttir þá er farið til baka og nágrannar síðustu nóðu eru skoðaðir.

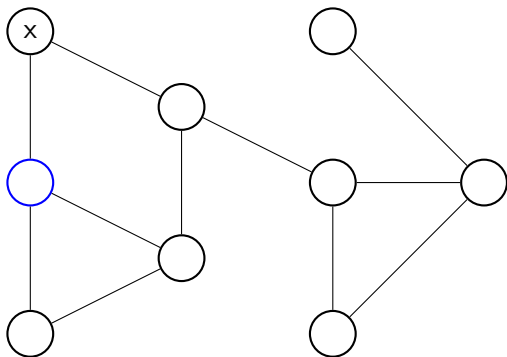
Dýptarleit

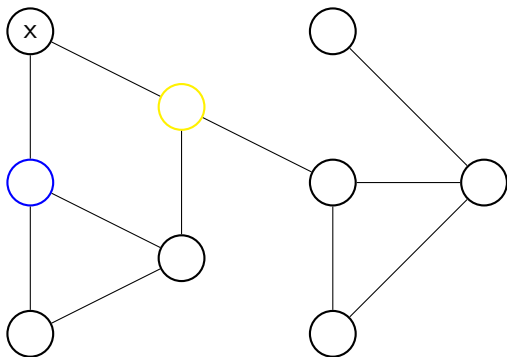
- ▶ Dýptarleit byrjar í einhverri nóðu.
- ▶ Í hverju skrefi heimsækir leitin nóðu sem hefur ekki verið heimsótt áður í leitinni.
- ▶ Ef allir nágrannar hafa verið heimsóttir þá er farið til baka og nágrannar síðustu nóðu eru skoðaðir.
- ▶ Tökum dæmi.

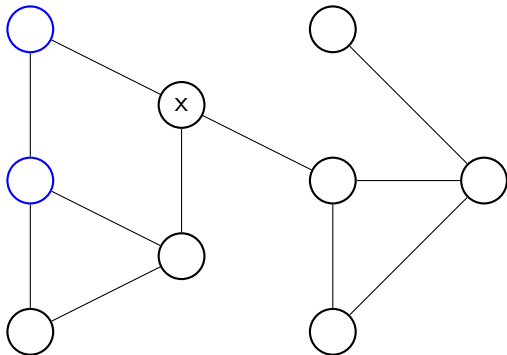


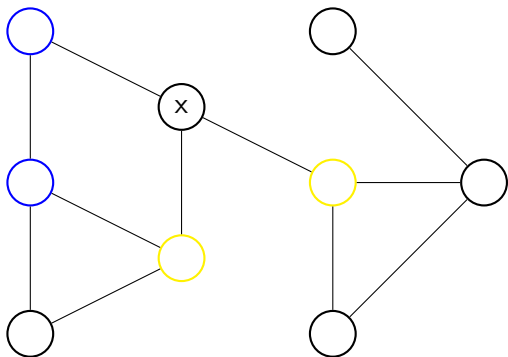


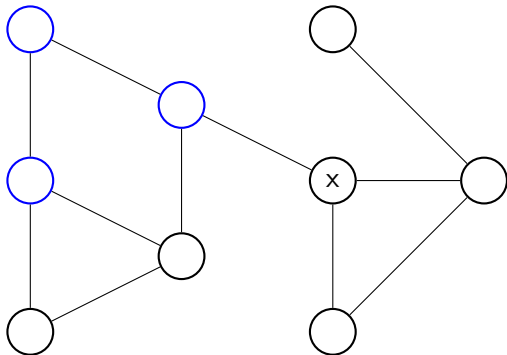


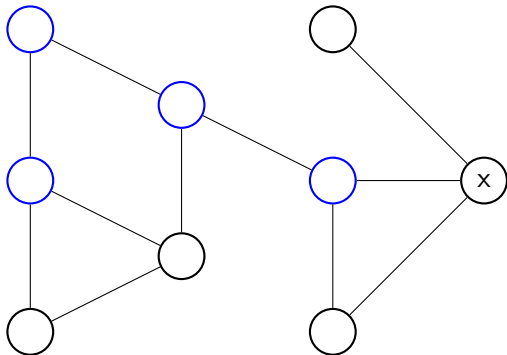


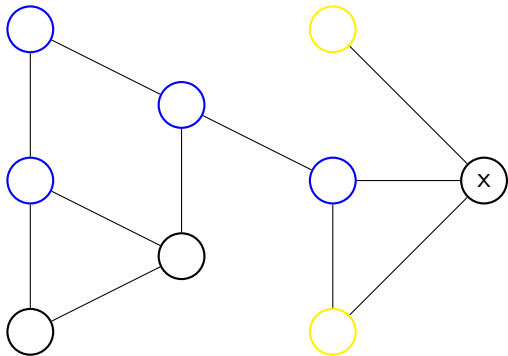


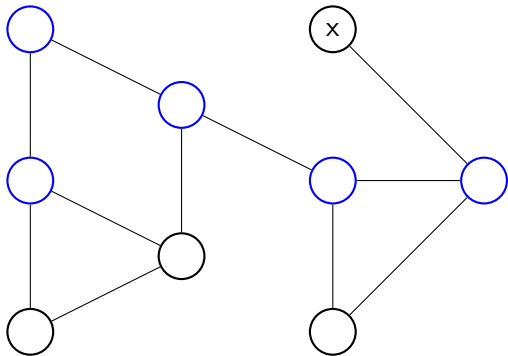


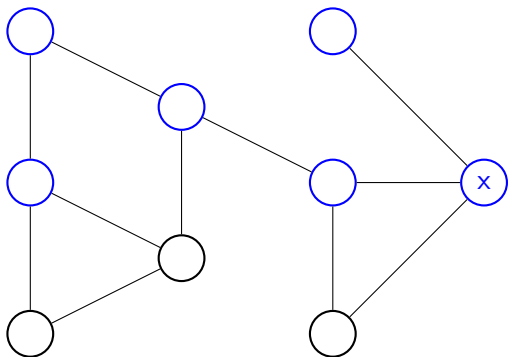


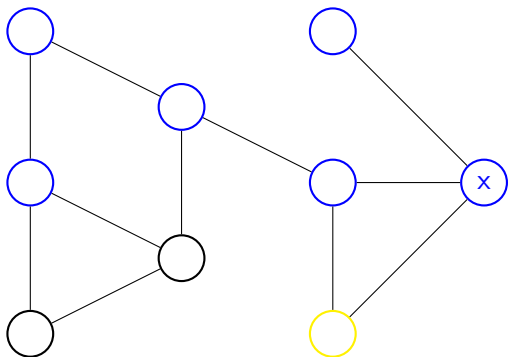


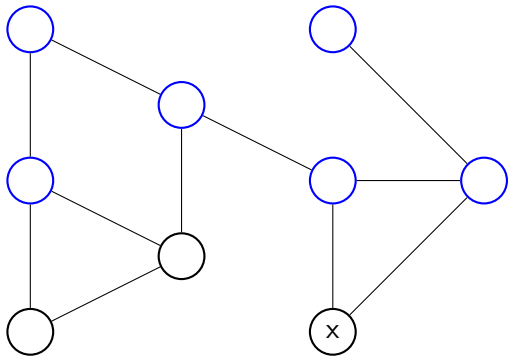


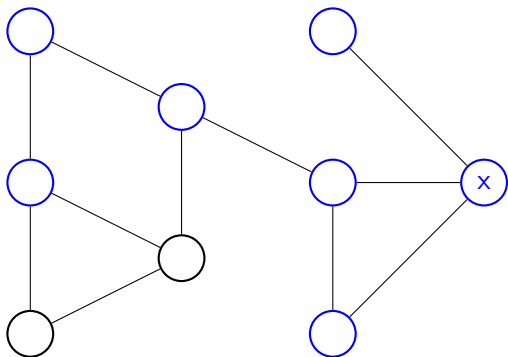


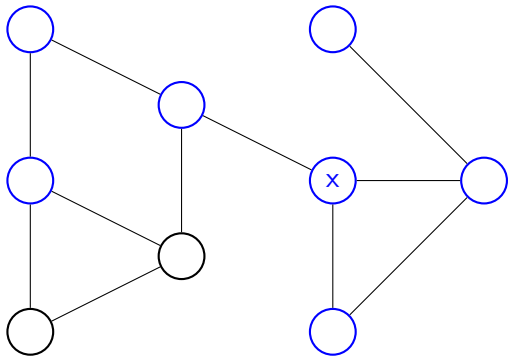


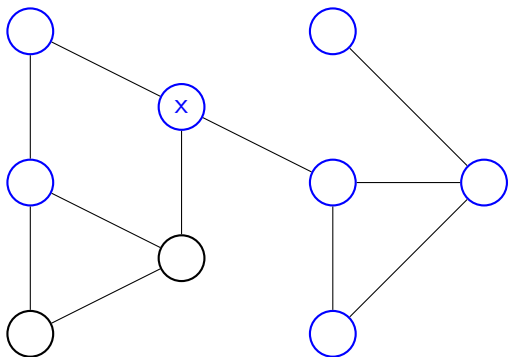


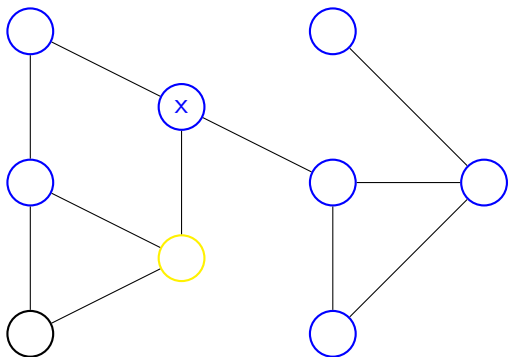


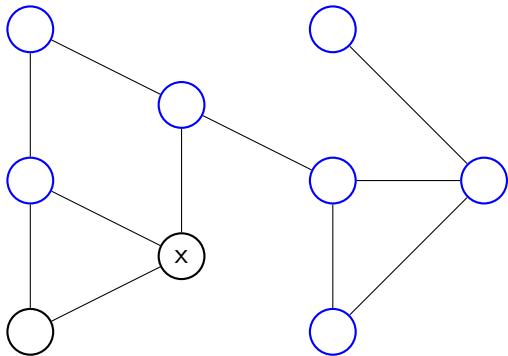


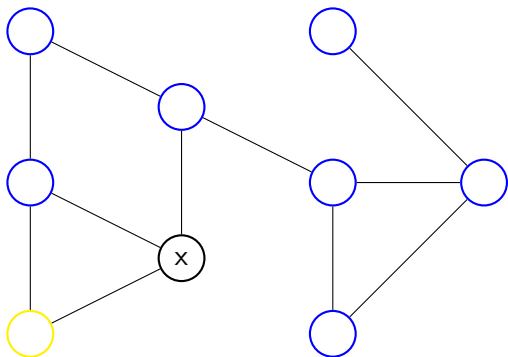


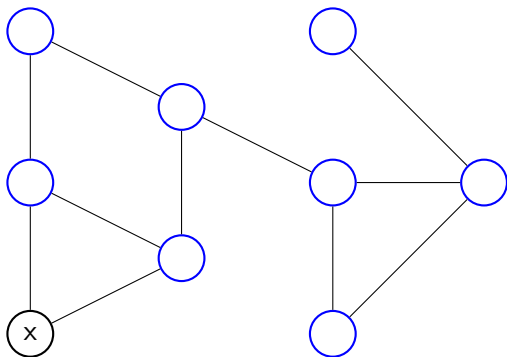


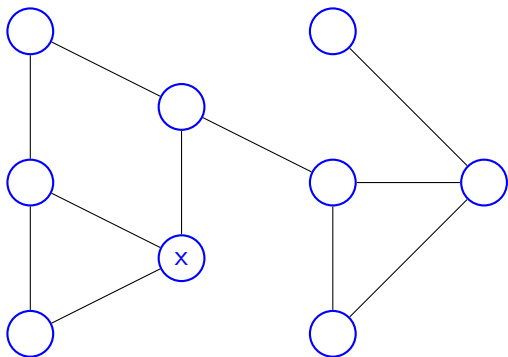


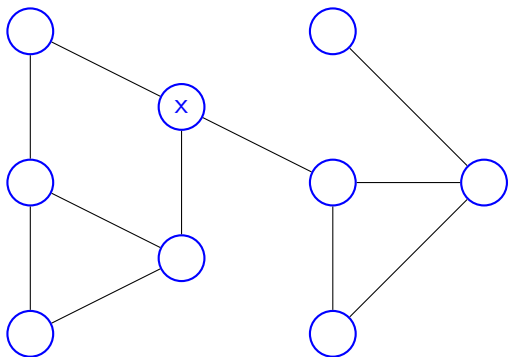


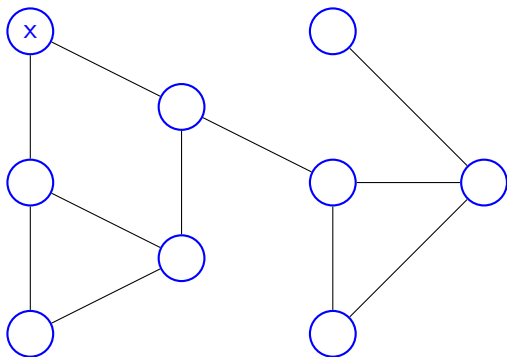


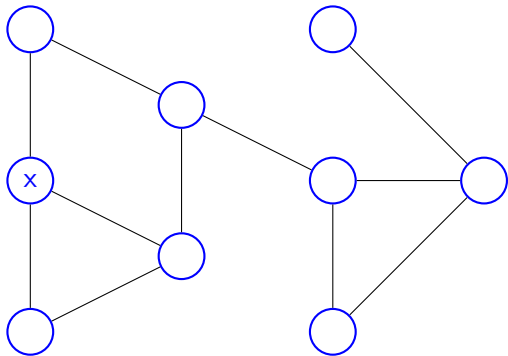


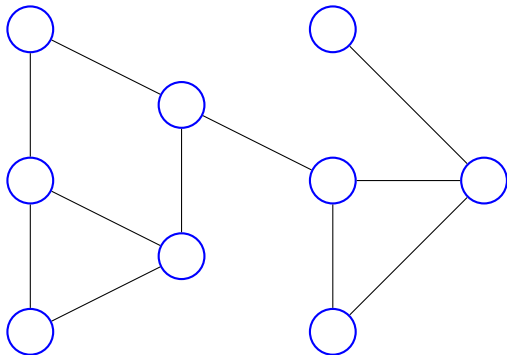












- ▶ Þegar kemur að því að útfæra dýptarleit er oftast notast við endurkvæmni.

- ▶ Þegar kemur að því að útfæra dýptarleit er oftast notast við endurkvæmni.
- ▶ Endurkvæmnin sér sjálfkrafa um að „fara til baka”.

- ▶ Þegar kemur að því að útfæra dýptarleit er oftast notast við endurkvæmni.
- ▶ Endurkvæmnin sér sjálfkrafa um að „fara til baka”.

```
7 vi v;  
8 void dfs(vvi& g, int x)  
9 {  
10     int i;  
11     printf("Vid erum i nodu %d\n", x + 1);  
12     v[x] = 1;  
13     rep(i, g[x].size()) if (v[g[x][i]] == 0) dfs(g, g[x][i]);  
14 }
```


- ▶ Þegar kemur að því að útfæra dýptarleit er oftast notast við endurkvæmni.
- ▶ Endurkvæmnin sér sjálfkrafa um að „fara til baka”.

```
7 vi v;  
8 void dfs(vvi& g, int x)  
9 {  
10     int i;  
11     printf("Vid erum i nodu %d\n", x + 1);  
12     v[x] = 1;  
13     rep(i, g[x].size()) if (v[g[x][i]] == 0) dfs(g, g[x][i]);  
14 }
```

- ▶ Eftir kall á `dfs(0)` segir `v[j]` okkur hvort til sé vegur frá nóðu 0 til nóðu `j`.

- ▶ Tökum eftir að leitin heimsækir hverju nóðu í mesta lagi einu sinni og ferðast eftir hverjum legg í mesta lagi tvisvar (einu sinni í stefndu neti).

- ▶ Tökum eftir að leitin heimsækir hverju nóðu í mesta lagi einu sinni og ferðast eftir hverjum legg í mesta lagi tvisvar (einu sinni í stefndu neti).
- ▶ Svo tímaflækjan er $\mathcal{O}(\quad)$.

- ▶ Tökum eftir að leitin heimsækir hverju nóðu í mesta lagi einu sinni og ferðast eftir hverjum legg í mesta lagi tvisvar (einu sinni í stefndu neti).
- ▶ Svo tímaflækjan er $\mathcal{O}(E + V)$.

- ▶ Tökum eftir að leitin heimsækir hverju nóðu í mesta lagi einu sinni og ferðast eftir hverjum legg í mesta lagi tvisvar (einu sinni í stefndu neti).
- ▶ Svo tímaflækjan er $\mathcal{O}(E + V)$.
- ▶ Við getum í rauninni ekki beðið um betri tímaflækju.

- ▶ Tökum eftir að leitin heimsækir hverju nóðu í mesta lagi einu sinni og ferðast eftir hverjum legg í mesta lagi tvisvar (einu sinni í stefndu neti).
- ▶ Svo tímaflækjan er $\mathcal{O}(E + V)$.
- ▶ Við getum í rauninni ekki beðið um betri tímaflækju.
- ▶ Við munum alltaf þurfa að skoða allar nóður og ef við skoðum ekki alla leggi þá erum við að hunsa uppbyggingu netsins.

Breiddarleit

- ▶ Við byrjum á að merkja allar nóður sem „ósnertar“, nema við merkjum eina nóðu sem „séð“.

Breiddarleit

- ▶ Við byrjum á að merkja allar nóður sem „ósnertar“, nema við merkjum eina nóðu sem „séð“.
- ▶ Sú nóða er kölluð *upphafsnóðan*.

Breiddarleit

- ▶ Við byrjum á að merkja allar nóður sem „ósnerar“, nema við merkjum eina nóðu sem „séð“.
- ▶ Sú nóða er kölluð *upphafsnóðan*.
- ▶ Við endurtökum svo sömu skrefin þar til engar „séðar“ nóður eru eftir:

Breiddarleit

- ▶ Við byrjum á að merkja allar nóður sem „ósnerar“, nema við merkjum eina nóðu sem „séð“.
- ▶ Sú nóða er kölluð *upphafsnóðan*.
- ▶ Við endurtökum svo sömu skrefin þar til engar „séðar“ nóður eru eftir:
 - ▶ Veljum þá „séðu“ nóðu sem við sáum fyrst.

Breiddarleit

- ▶ Við byrjum á að merkja allar nóður sem „ósnerar“, nema við merkjum eina nóðu sem „séð“.
- ▶ Sú nóða er kölluð *upphafsnóðan*.
- ▶ Við endurtökum svo sömu skrefin þar til engar „séðar“ nóður eru eftir:
 - ▶ Veljum þá „séðu“ nóðu sem við sáum fyrst.
 - ▶ Merkjum alla „ósnera“ nágranna hennar sem „séða“.

Breiddarleit

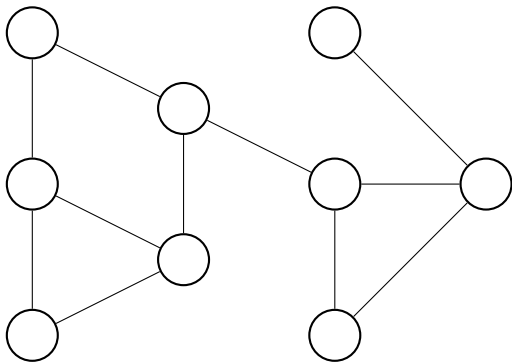
- ▶ Við byrjum á að merkja allar nóður sem „ósnertar“, nema við merkjum eina nóðu sem „séð“.
- ▶ Sú nóða er kölluð *upphafsnóðan*.
- ▶ Við endurtökum svo sömu skrefin þar til engar „séðar“ nóður eru eftir:
 - ▶ Veljum þá „séðu“ nóðu sem við sáum fyrst.
 - ▶ Merkjum alla „ósnertra“ nágranna hennar sem „séða“.
 - ▶ Merkjum upprunalegu nóðuna „kláraða“.

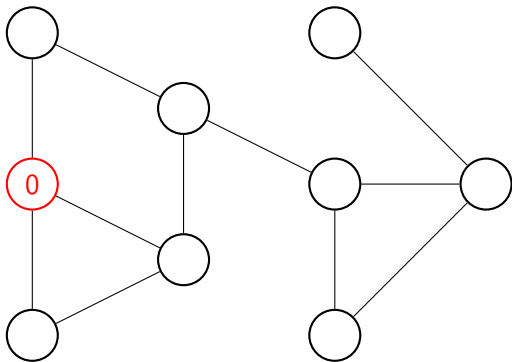
Breiddarleit

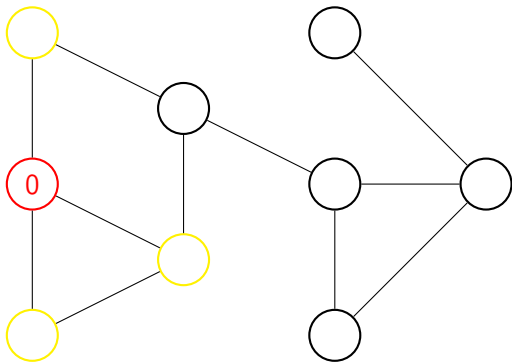
- ▶ Við byrjum á að merkja allar nóður sem „ósnerar“, nema við merkjum eina nóðu sem „séð“.
- ▶ Sú nóða er kölluð *upphafsnóðan*.
- ▶ Við endurtökum svo sömu skrefin þar til engar „séðar“ nóður eru eftir:
 - ▶ Veljum þá „séðu“ nóðu sem við sáum fyrst.
 - ▶ Merkjum alla „ósnera“ nágranna hennar sem „séða“.
 - ▶ Merkjum upprunalegu nóðuna „kláraða“.
- ▶ Tökum dæmi.

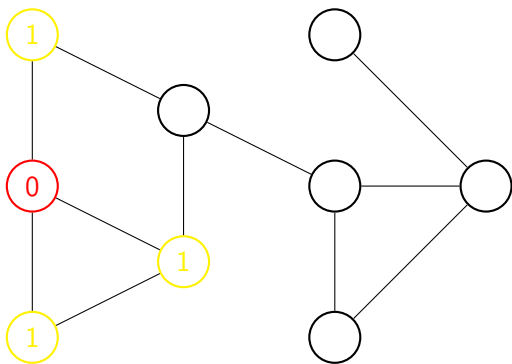
Breiddarleit

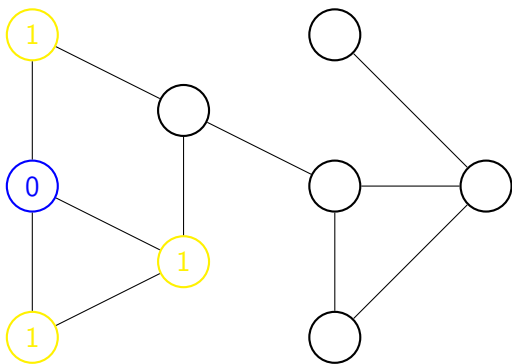
- ▶ Við byrjum á að merkja allar nóður sem „ósnerar“, nema við merkjum eina nóðu sem „séð“.
- ▶ Sú nóða er kölluð *upphafsnóðan*.
- ▶ Við endurtökum svo sömu skrefin þar til engar „séðar“ nóður eru eftir:
 - ▶ Veljum þá „séðu“ nóðu sem við sáum fyrst.
 - ▶ Merkjum alla „ósnera“ nágranna hennar sem „séða“.
 - ▶ Merkjum upprunalegu nóðuna „kláraða“.
- ▶ Tökum dæmi.
- ▶ Við munum merkja „séðu“ nóður með hvenær við sáum þær.

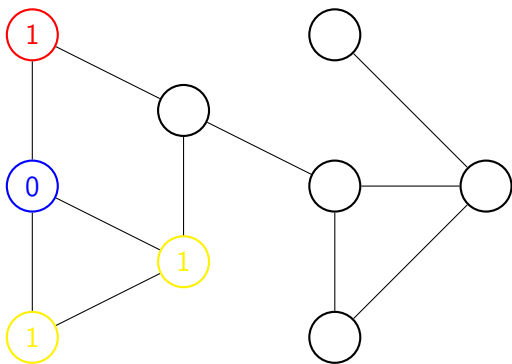


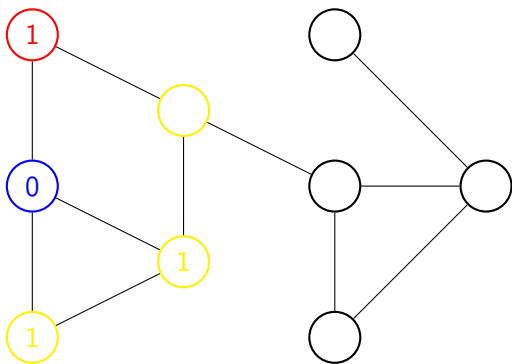


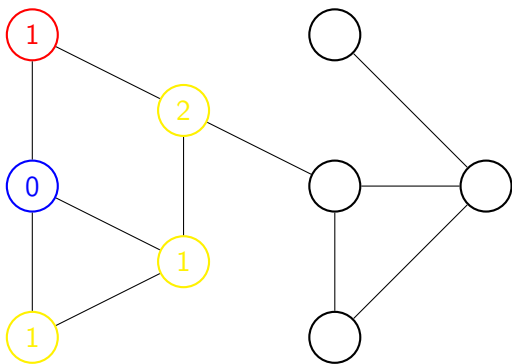


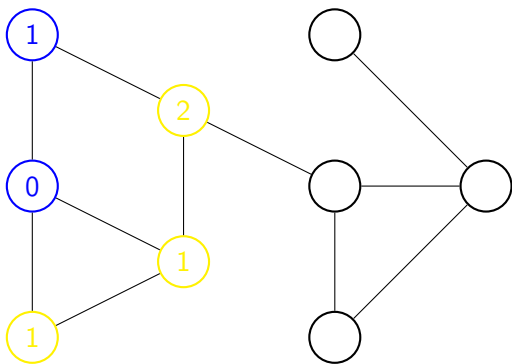


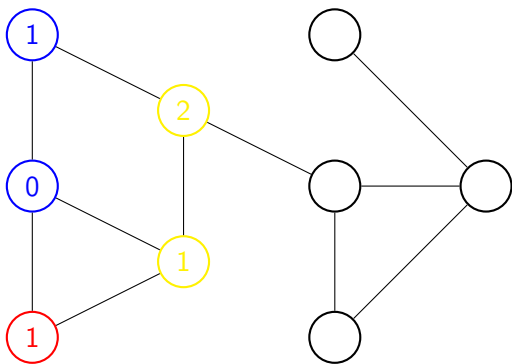


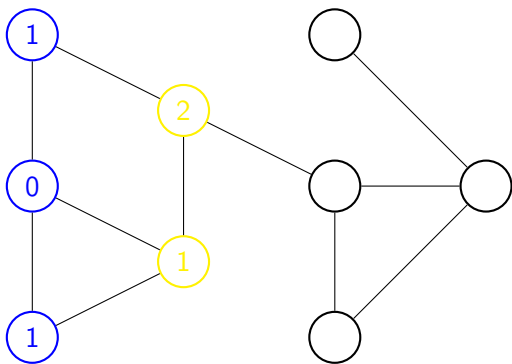


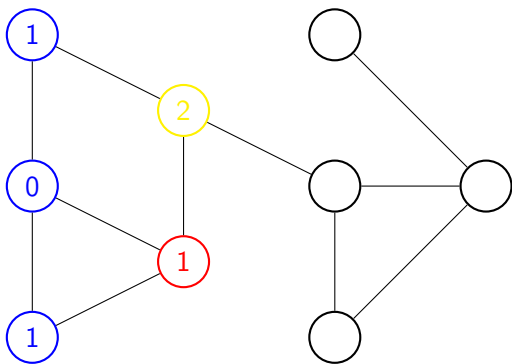


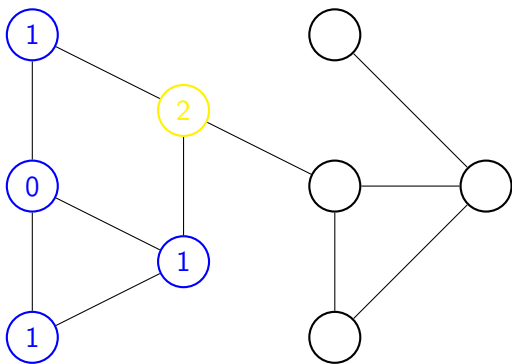


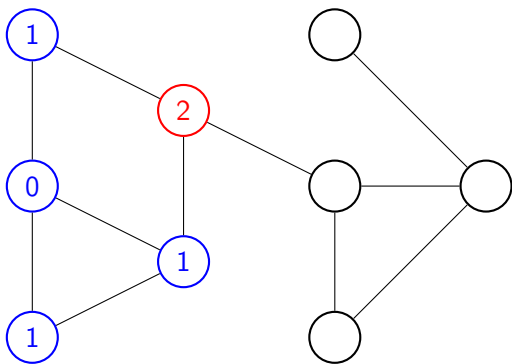


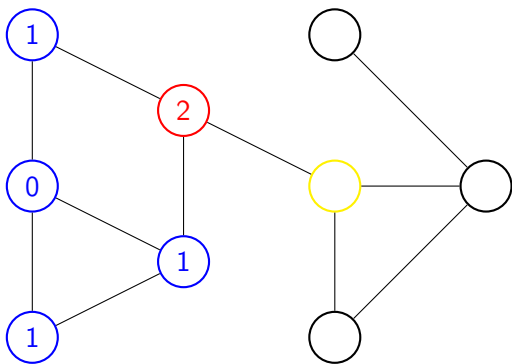


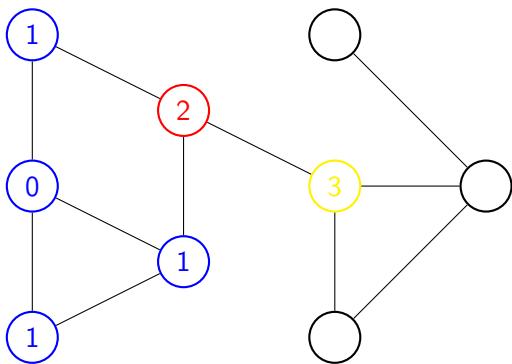


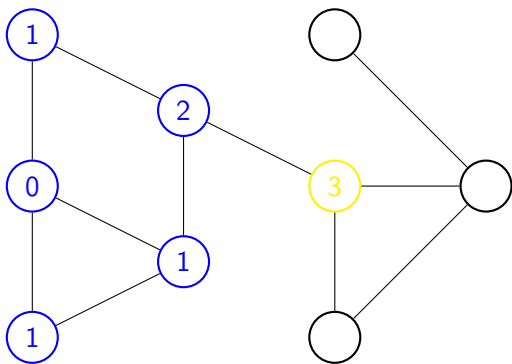


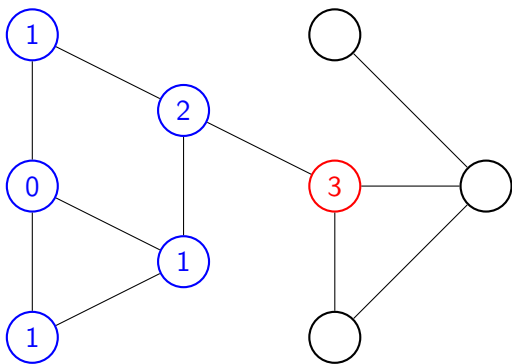


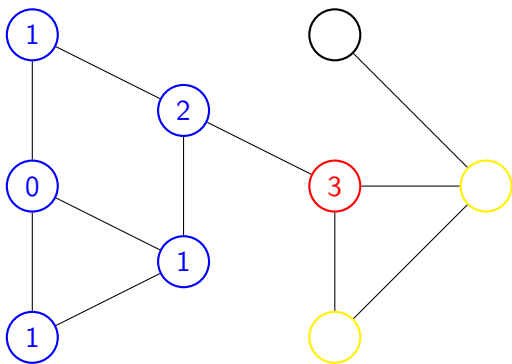


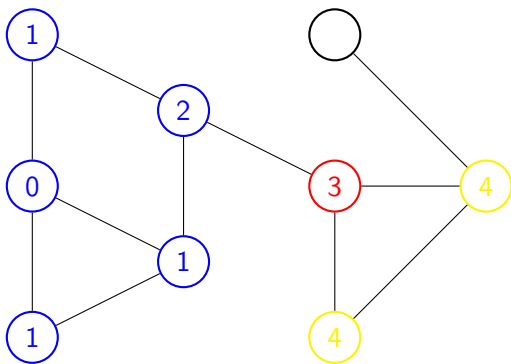


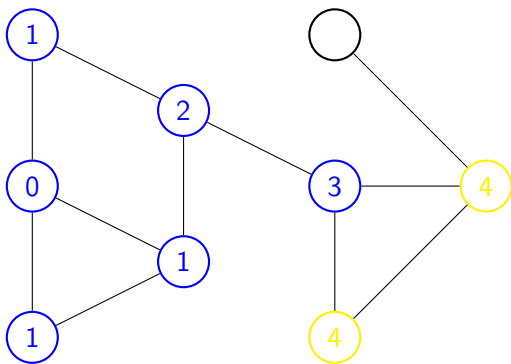


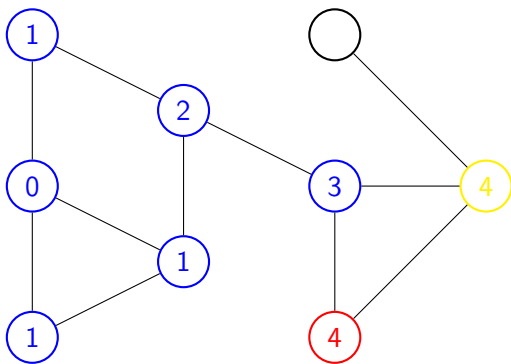


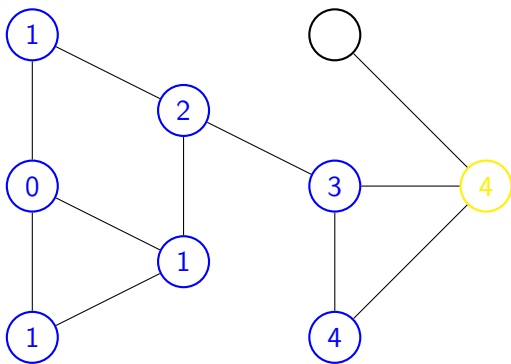


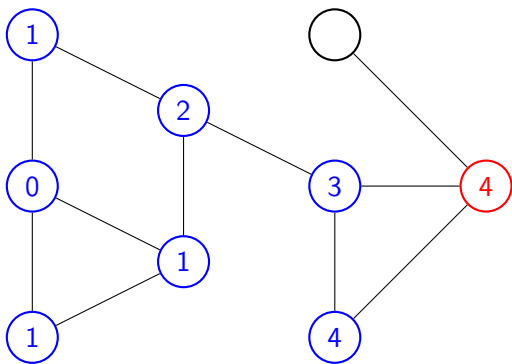


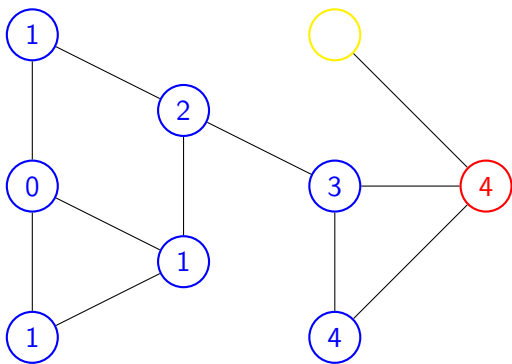


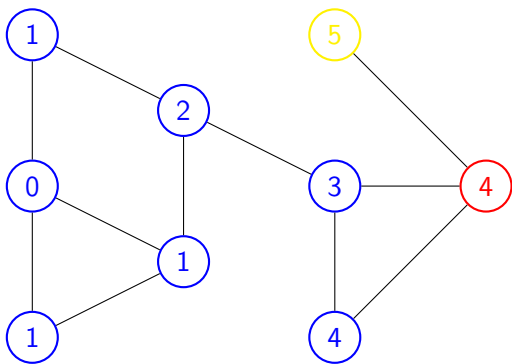


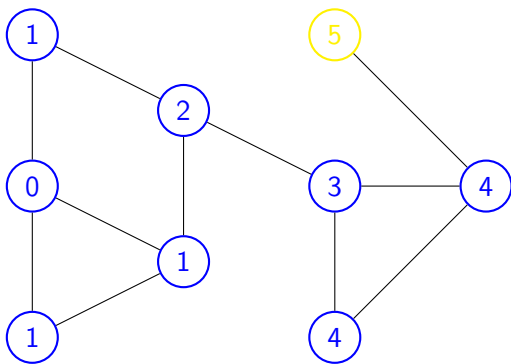


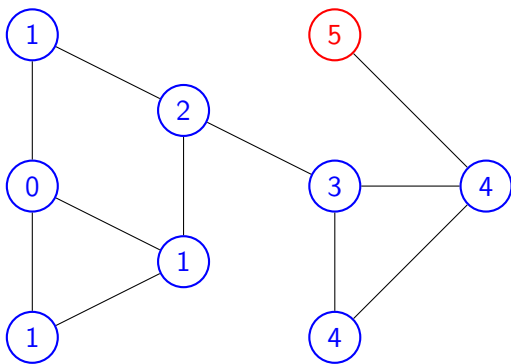


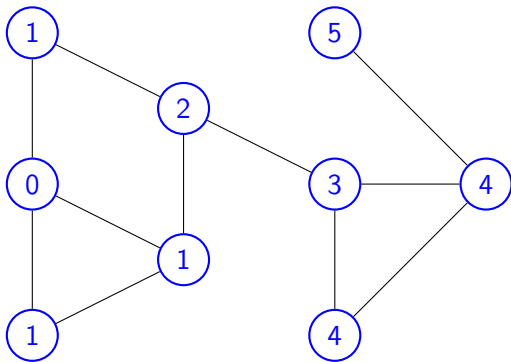












- ▶ Við munum halda utan um „séðu” nóðurnar með biðröð.

- ▶ Við munum halda utan um „séðu” nóðurnar með biðröð.
- ▶ Við byrjum því á að setja upphafsnóðuna okkar í biðröðina.

- ▶ Við munum halda utan um „séðu” nóðurnar með biðröð.
- ▶ Við byrjum því á að setja upphafsnóðuna okkar í biðröðina.
- ▶ Við tökum svo nóðu úr biðröðinni, setjum alla „óséða” nágranna hennar í biðröðina og höldum áfram þangað til biðröðin er tóm.

- ▶ Við munum halda utan um „séðu” nóðurnar með biðröð.
- ▶ Við byrjum því á að setja upphafsnóðuna okkar í biðröðina.
- ▶ Við tökum svo nóðu úr biðröðinni, setjum alla „óséða” nágranna hennar í biðröðina og höldum áfram þangað til biðröðin er tóm.

```
21     vi d(n, -1);
22     queue<int> q;
23     q.push(0);
24     d[0] = 0;
25     while (q.size() > 0)
26     {
27         int x = q.front();
28         q.pop();
29         for (i = 0; i < g[x].size(); i++) if (d[g[x][i]] == -1)
30         {
31             q.push(g[x][i]);
32             d[g[x][i]] = d[x] + 1;
33         }
34     }
```

- ▶ Við segjum að nódur u og v séu fjarlægð k frá hvori annari ef stysti vegurinn frá u til v er af lengd k .

- ▶ Við segjum að nódur u og v séu fjarlægð k frá hvori annari ef stysti vegurinn frá u til v er af lengd k .
- ▶ Við segjum líka að það séu k skref á milli nódanna.

- ▶ Við segjum að nódur u og v séu fjarlægð k frá hvori annari ef stysti vegurinn frá u til v er af lengd k .
- ▶ Við segjum líka að það séu k skref á milli nóchanna.
- ▶ Ef enginn vegur er á milli nóchanna segjum við að lengdin á milli þeirra sé ∞ .

- ▶ Við segjum að nóður u og v séu fjarlægð k frá hvori annari ef stysti vegurinn frá u til v er af lengd k .
- ▶ Við segjum líka að það séu k skref á milli nóðanna.
- ▶ Ef enginn vegur er á milli nóðanna segjum við að lengdin á milli þeirra sé ∞ .
- ▶ Milivægur eiginleiki breiddarleitar er að hún heimsækir fyrst þær nóður sem eru næst upphafsnóðunni.

- ▶ Við segjum að nóður u og v séu fjarlægð k frá hvori annari ef stysti vegurinn frá u til v er af lengd k .
- ▶ Við segjum líka að það séu k skref á milli nóðanna.
- ▶ Ef enginn vegur er á milli nóðanna segjum við að lengdin á milli þeirra sé ∞ .
- ▶ Milivægur eiginleiki breiddarleitar er að hún heimsækir fyrst þær nóður sem eru næst upphafsnóðunni.
- ▶ Með öðrum orðum, ef u er k_1 skref frá upphafsnóðunni og v er k_2 skref frá upphafsnóðunni, $k_1 \neq k_2$, þá heimsækir breiddarleit u á undan v þá og því aðeins að $k_1 < k_2$.

- ▶ Við segjum að nóður u og v séu fjarlægð k frá hvori annari ef stysti vegurinn frá u til v er af lengd k .
- ▶ Við segjum líka að það séu k skref á milli nóðanna.
- ▶ Ef enginn vegur er á milli nóðanna segjum við að lengdin á milli þeirra sé ∞ .
- ▶ Milivægur eiginleiki breiddarleitar er að hún heimsækir fyrst þær nóður sem eru næst upphafsnóðunni.
- ▶ Með öðrum orðum, ef u er k_1 skref frá upphafsnóðunni og v er k_2 skref frá upphafsnóðunni, $k_1 \neq k_2$, þá heimsækir breiddarleit u á undan v þá og því aðeins að $k_1 < k_2$.
- ▶ Við getum því notað breiddarleit til að finna fjarlægðina frá upphafsnóðunni að öllum öðrum nóðum.

- ▶ Líkt og í dýptarleit þá heimsækjum við hverja nóðu í mesta lagi einu sinni og ferðumst eftir hverjum legg í mesta lagi tvisvar (einu sinni í stefndu neti).

- ▶ Líkt og í dýptarleit þá heimsækjum við hverja nóðu í mesta lagi einu sinni og ferðumst eftir hverjum legg í mesta lagi tvisvar (einu sinni í stefndu neti).
- ▶ Svo tímaflækjan er aftur $\mathcal{O}(\quad)$.

- ▶ Líkt og í dýptarleit þá heimsækjum við hverja nóðu í mesta lagi einu sinni og ferðumst eftir hverjum legg í mesta lagi tvisvar (einu sinni í stefndu neti).
- ▶ Svo tímaflækjan er aftur $\mathcal{O}(E + V)$.

Samanburður

- ▶ Báðar leitirnar segja okkur til hvaða nóða má komast frá upphafsnóðunni og gera það með sömu tímaflækju.

Samanburður

- ▶ Báðar leitirnar segja okkur til hvaða nóða má komast frá upphafsnóðunni og gera það með sömu tímaflækju.
- ▶ Breiddarleit gefur okkur einnig fjarlægð allra nóða frá upphafsnóðunni.

Samanburður

- ▶ Báðar leitirnar segja okkur til hvaða nóða má komast frá upphafsnóðunni og gera það með sömu tímaflækju.
- ▶ Breiddarleit gefur okkur einnig fjarlægð allra nóða frá upphafsnóðunni.
- ▶ Í dýptarleit getum við unnið áfram með gögnin eftir endurkvæma kallið okkar, sem býður upp á mikla fjölbreyttni.

Samanburður

- ▶ Báðar leitirnar segja okkur til hvaða nóða má komast frá upphafsnóðunni og gera það með sömu tímaflækju.
- ▶ Breiddarleit gefur okkur einnig fjarlægð allra nóða frá upphafsnóðunni.
- ▶ Í dýptarleit getum við unnið áfram með gögnin eftir endurkvæma kallið okkar, sem býður upp á mikla fjölbreyttni.
- ▶ Dýptarleit má því finna í reikniritum sem finna grannröð neta, tengipunkta og brýr (þetta verður allt skilgreint seinna).

- Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c .

- ▶ Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c .
- ▶ Inntakið inniheldur síðan r strengir, allir af lengd c .

- ▶ Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c .
- ▶ Inntakið inniheldur síðan r strengir, allir af lengd c .
- ▶ Strengirnar byrjar og endar allir á 'X' ásamt því að fyrsti og síðasti strengurinn inniheldur bara stafinn 'X'.

- ▶ Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c .
- ▶ Inntakið inniheldur síðan r strengir, allir af lengd c .
- ▶ Strengirnar byrjar og endar allir á 'X' ásamt því að fyrsti og síðasti strengurinn inniheldur bara stafinn 'X'.
- ▶ Annars innihalda strengirnir bara stafina 'X', '.' og '0'.

- ▶ Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c .
- ▶ Inntakið inniheldur síðan r strengir, allir af lengd c .
- ▶ Strengirnar byrjar og endar allir á 'X' ásamt því að fyrsti og síðasti strengurinn inniheldur bara stafinn 'X'.
- ▶ Annars innihalda strengirnir bara stafina 'X', '.' og '0'.
- ▶ Dæmi um slíkan streng er:

```

1  11 14
2  XXXXXXXXXXXXXXXX
3  X.X.....X
4  X.X.XXXX.X.X.X
5  X.X.....XXXX.X
6  X0XXXXXX...X.X
7  X..X.X...X.X.X
8  X...XX.XXX.X.X
9  X....X...X.X.X
10 XXXX.XXX.X.X.X
11 X.....X...X
12 XXXXXXXXXXXXXXXX

```

- ▶ Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c .
- ▶ Inntakið inniheldur síðan r strengir, allir af lengd c .
- ▶ Strengirnar byrjar og endar allir á 'X' ásamt því að fyrsti og síðasti strengurinn inniheldur bara stafinn 'X'.
- ▶ Annars innihalda strengirnir bara stafina 'X', '.' og '0'.
- ▶ Dæmi um slíkan streng er:

```

1  11 14
2  XXXXXXXXXXXXXXXX
3  X.X.....X
4  X.X.XXXX.X.X.X
5  X.X.....XXXX.X
6  XOXXXXXX...X.X
7  X..X.X...X.X.X
8  X...XX.XXX.X.X
9  X....X...X.X.X
10 XXXX.XXX.X.X.X
11 X.....X...X
12 XXXXXXXXXXXXXXXX

```

- ▶ Við viljum svo prenta sama borð, nema í stað bókstafana á að koma hversu fá skref við þurfum að taka til frá '0' til að komast þangað ef við megum ferðast upp, niður, til hægri og til vinstri.

- ▶ Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c .
- ▶ Inntakið inniheldur síðan r strengir, allir af lengd c .
- ▶ Strengirnar byrjar og endar allir á 'X' ásamt því að fyrsti og síðasti strengurinn inniheldur bara stafinn 'X'.
- ▶ Annars innihalda strengirnir bara stafina 'X', '.' og '0'.
- ▶ Dæmi um slíkan streng er:

```

1  11 14
2  XXXXXXXXXXXXXXXX
3  X.X.....X
4  X.X.XXXX.X.X.X
5  X.X.....XXXX.X
6  XOXXXXXX...X.X
7  X..X.X...X.X.X
8  X...XX.XXX.X.X
9  X....X...X.X.X
10 XXXX.XXX.X.X.X
11 X.....X...X
12 XXXXXXXXXXXXXXXX

```

- ▶ Við viljum svo prenta sama borð, nema í stað bókstafana á að koma hversu fá skref við þurfum að taka til frá '0' til að komast þangað ef við megum ferðast upp, niður, til hægri og til vinstri.
- ▶ Fyrir þá reiti sem við komumst ekki á prentum við -1.

► Sem dæmi hefur inntakið

```
1 11 14
2 XXXXXXXXXXXXXXXX
3 X.X.....X
4 X.X.XXXX.X.X.X
5 X.X.....XXXX.X
6 XOXXXXXX...X.X
7 X..X.X...X.X.X
8 X...XX.XXX.X.X
9 X....X...X.X.X
10 XXXX.XXX.X.X.X
11 X.....X...X
12 XXXXXXXXXXXXXXXX
```

úttakið

```
1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
2 -1 3 -1 47 46 45 44 43 42 41 40 39 38 -1
3 -1 2 -1 48 -1 -1 -1 -1 43 -1 41 -1 37 -1
4 -1 1 -1 49 50 51 52 53 -1 -1 -1 -1 36 -1
5 -1 0 -1 -1 -1 -1 -1 -1 21 22 23 -1 35 -1
6 -1 1 2 -1 -1 -1 18 19 20 -1 24 -1 34 -1
7 -1 2 3 4 -1 -1 17 -1 -1 -1 25 -1 33 -1
8 -1 3 4 5 6 -1 16 15 14 -1 26 -1 32 -1
9 -1 -1 -1 -1 7 -1 -1 -1 13 -1 27 -1 31 -1
10 -1 11 10 9 8 9 10 11 12 -1 28 29 30 -1
11 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

- ▶ Við getum túlkað þessa mynd sem net.

- ▶ Við getum túlkað þessa mynd sem net.
- ▶ Ímyndum okkur að hver auður reitur sé nóða.

- ▶ Við getum túlkað þessa mynd sem net.
- ▶ Ímyndum okkur að hver auður reitur sé nóða.
- ▶ Við tengjum svo aðliggjandi auða nóður með leggjum.

- ▶ Við getum túlkað þessa mynd sem net.
- ▶ Ímyndum okkur að hver auður reitur sé nóða.
- ▶ Við tengjum svo aðliggjandi auða nóður með leggjum.
- ▶ Þar við viljum finna fjalægðir frá tiltekinni nóðu til allra annara nóða notum við leit.

- ▶ Við getum túlkað þessa mynd sem net.
- ▶ Ímyndum okkur að hver auður reitur sé nóða.
- ▶ Við tengjum svo aðliggjandi auða nóður með leggjum.
- ▶ Þar við viljum finna fjalægðir frá tiltekinni nóðu til allra annara nóða notum við breiddarleit.

- ▶ Við getum túlkað þessa mynd sem net.
- ▶ Ímyndum okkur að hver auður reitur sé nóða.
- ▶ Við tengjum svo aðliggjandi auða nóður með leggjum.
- ▶ Þar við viljum finna fjalægðir frá tiltekinni nóðu til allra annara nóða notum við breiddarleit.
- ▶ Fjöldi nóða í netinu er alltaf minni en $r \cdot c$ og fjöldi leggja er alltaf minni en $2 \cdot r \cdot c$.

- ▶ Við getum túlkað þessa mynd sem net.
- ▶ Ímyndum okkur að hver auður reitur sé nóða.
- ▶ Við tengjum svo aðliggjandi auða nóður með leggjum.
- ▶ Þar við viljum finna fjalægðir frá tiltekinni nóðu til allra annara nóða notum við breiddarleit.
- ▶ Fjöldi nóða í netinu er alltaf minni en $r \cdot c$ og fjöldi leggja er alltaf minni en $2 \cdot r \cdot c$.
- ▶ Svo þetta reiknirit er $\mathcal{O}(\quad)$.

- ▶ Við getum túlkað þessa mynd sem net.
- ▶ Ímyndum okkur að hver auður reitur sé nóða.
- ▶ Við tengjum svo aðliggjandi auða nóður með leggjum.
- ▶ Þar við viljum finna fjalægðir frá tiltekinni nóðu til allra annara nóða notum við breiddarleit.
- ▶ Fjöldi nóða í netinu er alltaf minni en $r \cdot c$ og fjöldi leggja er alltaf minni en $2 \cdot r \cdot c$.
- ▶ Svo þetta reiknirit er $\mathcal{O}(r \cdot c)$.

```

1 #include <bits/stdc++.h>
2 #define rep(E, F) for (E = 0; E < (F); E++)
3 using namespace std;
4 typedef pair<int, int> ii;
5
6 int isin(int x, int y, int r, int c)
7 { // Segir okkur hvort (x, y) sé fyrir utan myndina okkar.
8   if (x >= r || x < 0 || y >= c || y < 0) return 0;
9   return 1;
10 }
11
12 int main()
13 {
14   int i, j, r, c, x, y, g[4][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
15   cin >> r >> c;
16   char a[r][c + 1];
17   rep(i, r) cin >> a[i];
18   rep(i, r) rep(j, c) if (a[i][j] == 'O') x = i, y = j;
19   int d[r][c];
20   rep(i, r) rep(j, c) d[i][j] = -1;
21   queue<ii> q;
22   q.push(ii(x, y)), d[x][y] = 0;
23   while (q.size() > 0)
24   {
25     ii p = q.front(); q.pop();
26     x = p.first, y = p.second;
27     rep(i, 4)
28     {
29       int xx = x + g[i][0], yy = y + g[i][1];
30       if (a[xx][yy] == 'X' || !isin(xx, yy, r, c) || d[xx][yy] != -1)
31         continue;
32       q.push(ii(xx, yy)), d[xx][yy] = d[x][y] + 1;
33     }
34   }
35   rep(i, r) { rep(j, c) printf("%2d ", d[i][j]); printf("\n"); }
36   return 0;
37 }

```

- ▶ Við getum breitt dýptarleitar útfærslunni okkar lítillega til að finna alla einfalda vegi í neti sem byrja í tiltekinni nóðu.

- ▶ Við getum breitt dýptarleitar útfærslunni okkar lítillega til að finna alla einfalda vegi í neti sem byrja í tiltekinni nóðu.
- ▶ Til að koma í veg fyrir að heimsækja nóðu oftar en einu sinni í dýpterleit merkjum við hana og heimsækjum ekki merktar nóður.

- ▶ Við getum breitt dýptarleitar útfærslunni okkar lítillega til að finna alla einfalda vegi í neti sem byrja í tiltekinni nóðu.
- ▶ Til að koma í veg fyrir að heimsækja nóðu oftar en einu sinni í dýpterleit merkjum við hana og heimsækjum ekki merktar nóður.
- ▶ Við munum ennþá þurfa að merkja nóður því við erum að leita að einföldum vegum (almennt er ekki takmarkaður fjöldi vega í neti).

- ▶ Við getum breitt dýptarleitar útfærslunni okkar lítillega til að finna alla einfalda vegi í neti sem byrja í tiltekinni nóðu.
- ▶ Til að koma í veg fyrir að heimsækja nóðu oftar en einu sinni í dýpterleit merkjum við hana og heimsækjum ekki merktar nóður.
- ▶ Við munum ennþá þurfa að merkja nóður því við erum að leita að einföldum vegum (almennt er ekki takmarkaður fjöldi vega í neti).
- ▶ Munur er að við munum merkja nóðuna þegar við sjáum hana, halda áfram endurkvæmt til ómerktra nágranna hennar og afmerkja hana svo.

```
7 vi v, p;
8 void dfs(vvi& g, int x)
9 {
10     int i, f = 1;
11     v[x] = 1;
12     p.push_back(x);
13     rep(i, g[x].size()) if (v[g[x][i]] == 0)
14         dfs(g, g[x][i]), f = 0;
15     if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
16     p.pop_back();
17     v[x] = 0;
18 }
```

```

7 vi v, p;
8 void dfs(vvi& g, int x)
9 {
10     int i, f = 1;
11     v[x] = 1;
12     p.push_back(x);
13     rep(i, g[x].size()) if (v[g[x][i]] == 0)
14         dfs(g, g[x][i]), f = 0;
15     if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
16     p.pop_back();
17     v[x] = 0;
18 }

```

- Þetta forrit prentar alla einfalda vegi sem ekki mætti lengja.

```

7  vi v, p;
8  void dfs(vvi& g, int x)
9  {
10     int i, f = 1;
11     v[x] = 1;
12     p.push_back(x);
13     rep(i, g[x].size()) if (v[g[x][i]] == 0)
14         dfs(g, g[x][i]), f = 0;
15     if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
16     p.pop_back();
17     v[x] = 0;
18 }

```

- ▶ Þetta forrit prentar alla einfalda vegi sem ekki mætti lengja.
- ▶ Til að hámarka fjölda slíkra vega getum við búið til net þar sem öll pör nóða eru nágrannar.

```

7  vi v, p;
8  void dfs(vvi& g, int x)
9  {
10     int i, f = 1;
11     v[x] = 1;
12     p.push_back(x);
13     rep(i, g[x].size()) if (v[g[x][i]] == 0)
14         dfs(g, g[x][i]), f = 0;
15     if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
16     p.pop_back();
17     v[x] = 0;
18 }

```

- ▶ Þetta forrit prentar alla einfalda vegi sem ekki mætti lengja.
- ▶ Til að hámarka fjölda slíkra vega getum við búið til net þar sem öll pör nóða eru nágrannar.
- ▶ Þá myndi þetta forrit prenta allar umraðanir.

```

7  vi v, p;
8  void dfs(vvi& g, int x)
9  {
10     int i, f = 1;
11     v[x] = 1;
12     p.push_back(x);
13     rep(i, g[x].size()) if (v[g[x][i]] == 0)
14         dfs(g, g[x][i]), f = 0;
15     if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
16     p.pop_back();
17     v[x] = 0;
18 }

```

- ▶ Þetta forrit prentar alla einfalda vegi sem ekki mætti lengja.
- ▶ Til að hámarka fjölda slíkra vega getum við búið til net þar sem öll pör nóða eru nágrannar.
- ▶ Þá myndi þetta forrit prenta allar umraðanir.
- ▶ Tímaflækjan er því $\mathcal{O}(\quad)$.

```

7  vi v, p;
8  void dfs(vvi& g, int x)
9  {
10     int i, f = 1;
11     v[x] = 1;
12     p.push_back(x);
13     rep(i, g[x].size()) if (v[g[x][i]] == 0)
14         dfs(g, g[x][i]), f = 0;
15     if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
16     p.pop_back();
17     v[x] = 0;
18 }

```

- ▶ Þetta forrit prentar alla einfalda vegi sem ekki mætti lengja.
- ▶ Til að hámarka fjölda slíkra vega getum við búið til net þar sem öll pör nóða eru nágrannar.
- ▶ Þá myndi þetta forrit prenta allar umraðanir.
- ▶ Tímaflækjan er því $\mathcal{O}((V + 1)!)$.

