Dýptarleit og breiddarleit

Bergur Snorrason

23. febrúar 2021

- Hvernig ítrum við í gegnum alla hnúta nets.

- Hvernig ítrum við í gegnum alla hnúta nets.
- Þetta má að sjálfsögðu gera á marga vegu, en algengt er að notast við annað að tvennu:

- Hvernig ítrum við í gegnum alla hnúta 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 alla hnúta 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 alla hnúta 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 aðferðir byggja á því að byrja í einhverjum hnút og heimsækja svo nágranna hans.

Dýptarleit

- Dýptarleit byrjar í einhverjum hnút.

Dýptarleit

- Dýptarleit byrjar í einhverjum hnút.
- Sá hnútur er kallaður *upphafshnúturinn*.

Dýptarleit

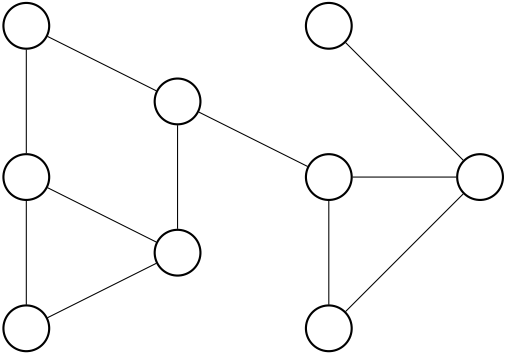
- Dýptarleit byrjar í einhverjum hnút.
- Sá hnútur er kallaður *upphafshnúturinn*.
- Í hverju skrefi heimsækir leitin einhvern nágranna hnútsins sem hefur ekki verið heimsóttur áður í leitinni.

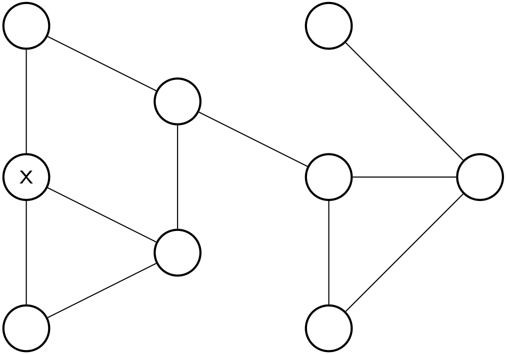
Dýptarleit

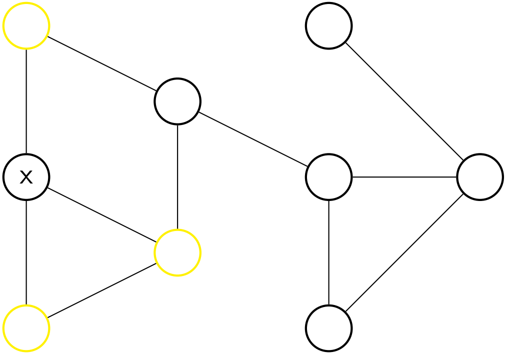
- Dýptarleit byrjar í einhverjum hnút.
- Sá hnútur er kallaður *upphafshnúturinn*.
- Í hverju skrefi heimsækir leitin einhvern nágranna hnútsins sem hefur ekki verið heimsóttur áður í leitinni.
- Ef allir nágrannar hafa verið heimsóttir þá er farið til baka og nágrannar síðasta hnúts eru skoðaðir.

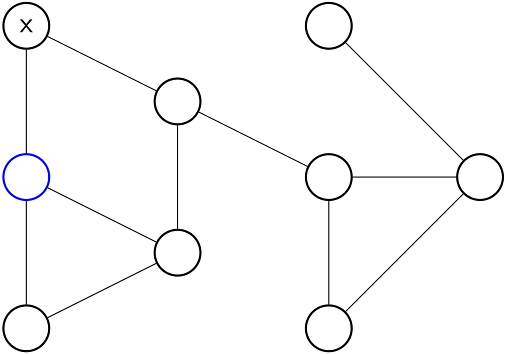
Dýptarleit

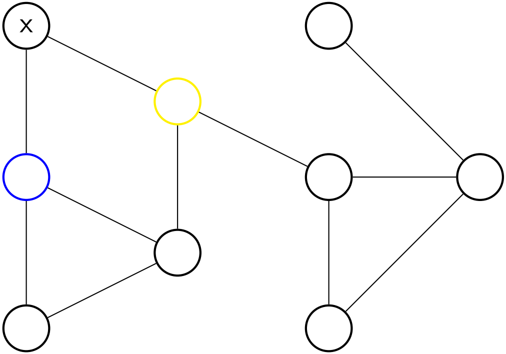
- Dýptarleit byrjar í einhverjum hnút.
- Sá hnútur er kallaður *upphafshnúturinn*.
- Í hverju skrefi heimsækir leitin einhvern nágranna hnútsins sem hefur ekki verið heimsóttur áður í leitinni.
- Ef allir nágrannar hafa verið heimsóttir þá er farið til baka og nágrannar síðasta hnúts eru skoðaðir.
- Tökum dæmi.

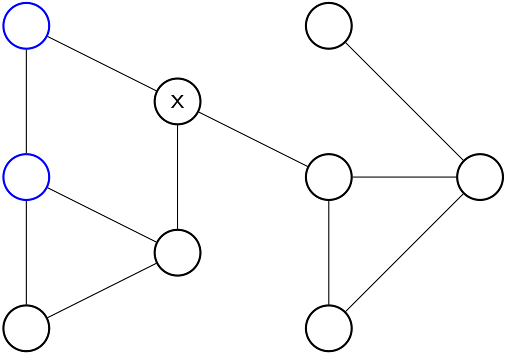


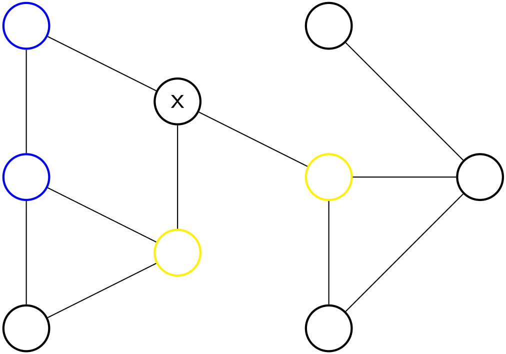

x

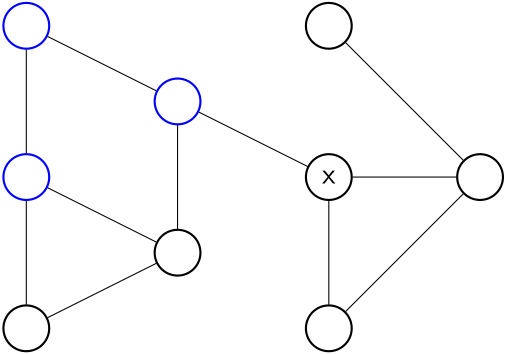

x

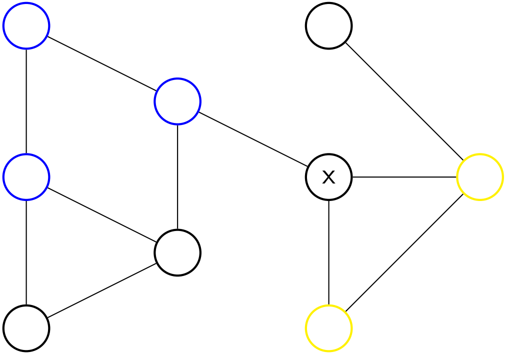

x

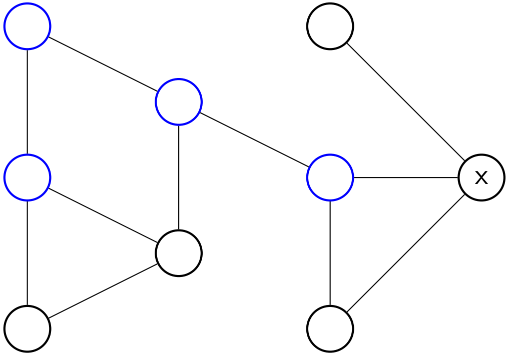

x

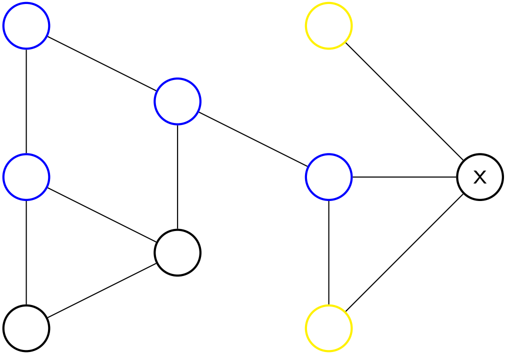

x

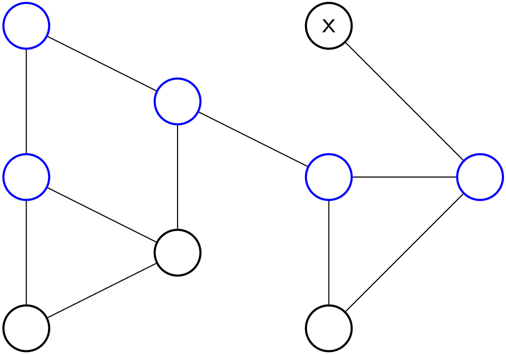

x

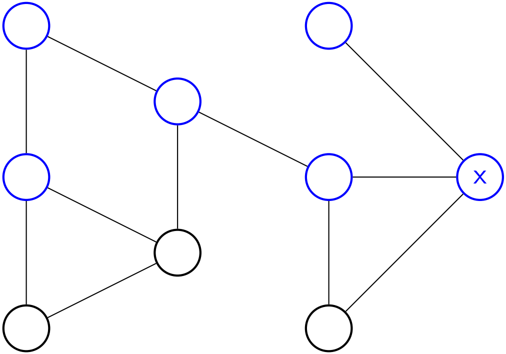

x

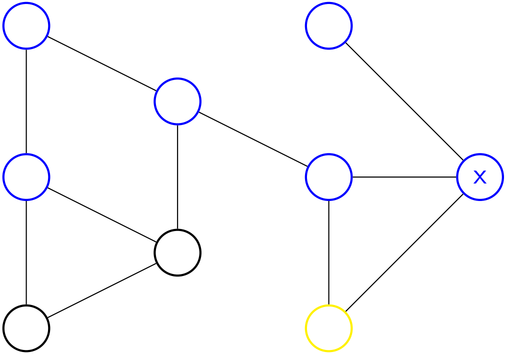

x

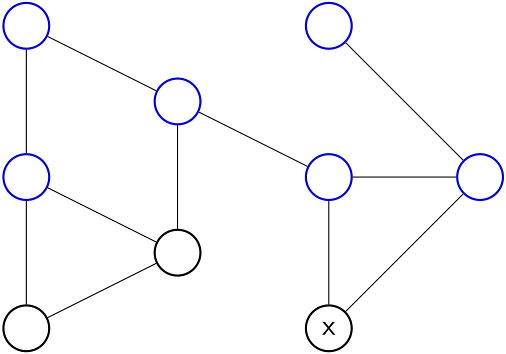

x

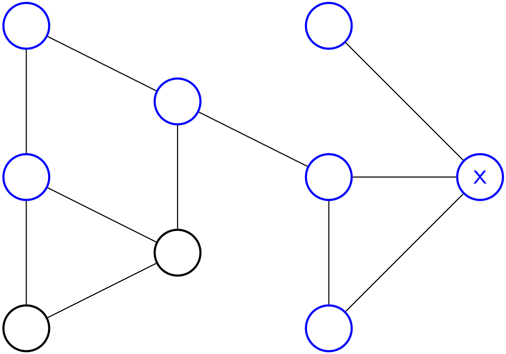

x

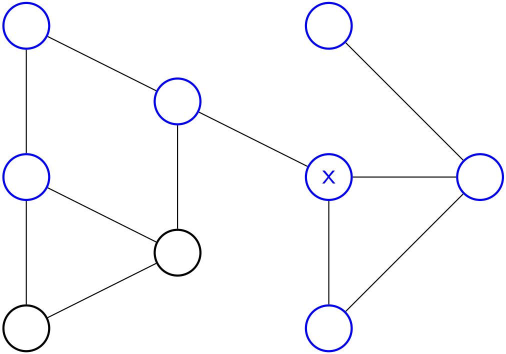

x

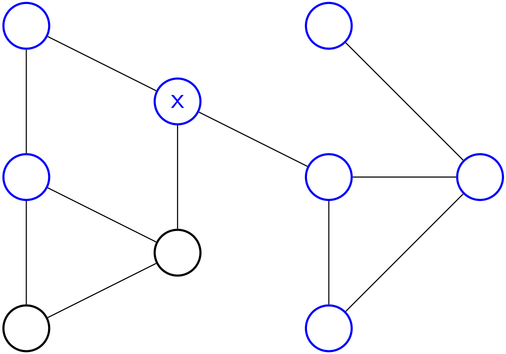

x

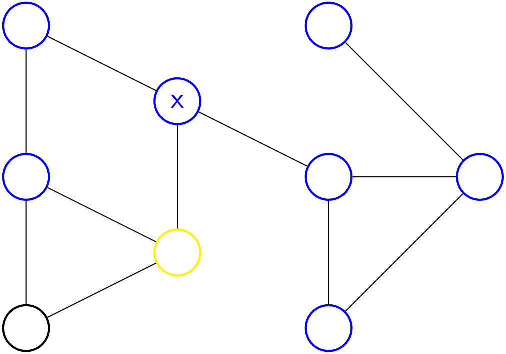

x

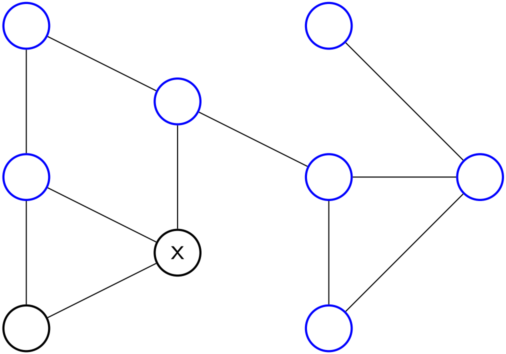

x

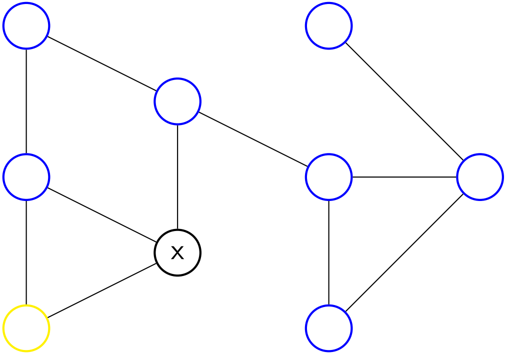

x

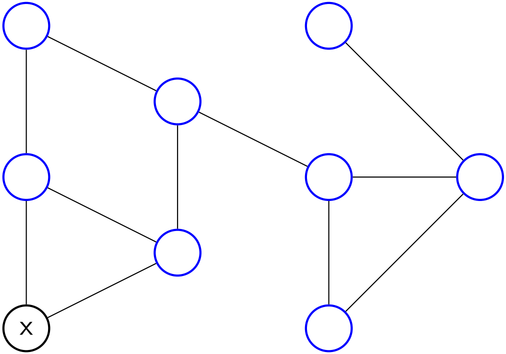

x

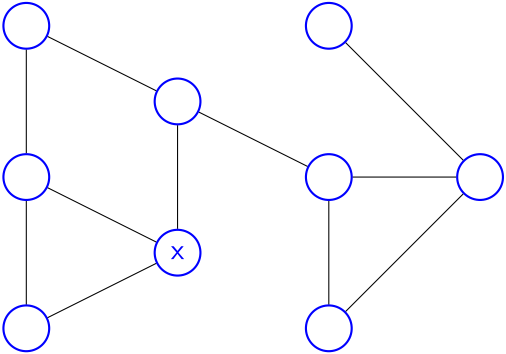

x

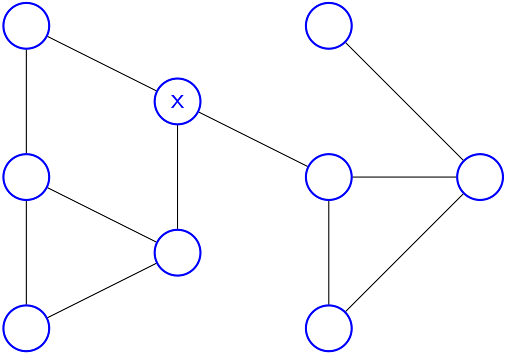

x

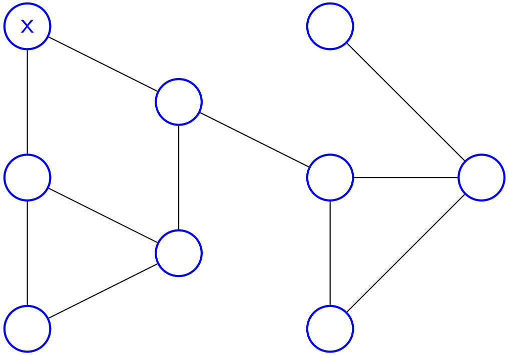

x

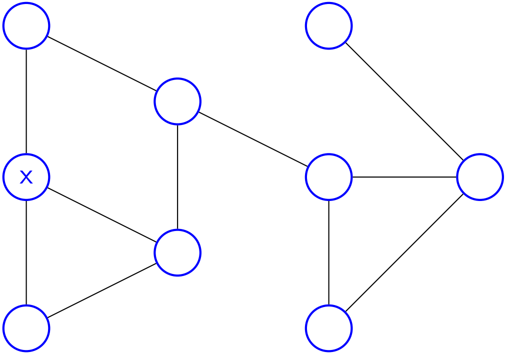

x

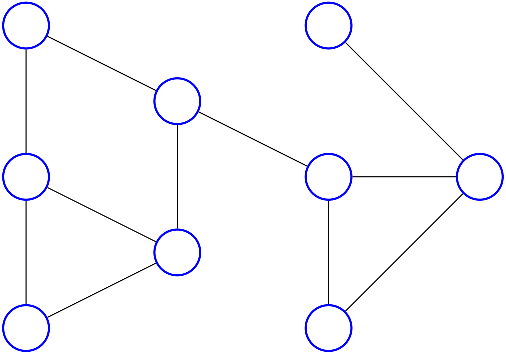

x


x


x


x


x



- 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 sjálkrafa um að „fara til baka”.

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

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

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

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

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

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

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

- Tökum eftir að leitin heimsækir hvern hnút 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 hvern hnút 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 hvern hnút 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 alla hnúta og ef við skoðum ekki alla leggi þá erum við að hunsa uppbyggingu netsins.

Breiddarleit

- Við byrjum á að merkja alla hnúta sem „ósnerta“, nema við merkjum einn hnút sem „séðan“.

Breiddarleit

- Við byrjum á að merkja alla hnúta sem „ósnerta", nema við merkjum einn hnút sem „séðan".
- Sá hnútur er kallaður *upphafshnúturinn*.

Breiddarleit

- Við byrjum á að merkja alla hnúta sem „ósnerta", nema við merkjum einn hnút sem „séðan".
- Sá hnútur er kallaður *upphafshnúturinn*.
- Við endurtökum svo sömu skrefin þar til engir „séðir" hnútar eru eftir:

Breiddarleit

- Við byrjum á að merkja alla hnúta sem „ósnerta", nema við merkjum einn hnút sem „séðan".
- Sá hnútur er kallaður *upphafshnúturinn*.
- Við endurtökum svo sömu skrefin þar til engir „séðir" hnútar eru eftir:
  - Veljum þann „séða" hnút sem við sáum fyrst.

Breiddarleit

- Við byrjum á að merkja alla hnúta sem „ósnerta", nema við merkjum einn hnút sem „séðan".
- Sá hnútur er kallaður *upphafshnúturinn*.
- Við endurtökum svo sömu skrefin þar til engir „séðir" hnútar eru eftir:
    - Veljum þann „séða" hnút sem við sáum fyrst.
    - Merkjum alla „ósnerta" nágranna hans sem „séða".

Breiddarleit

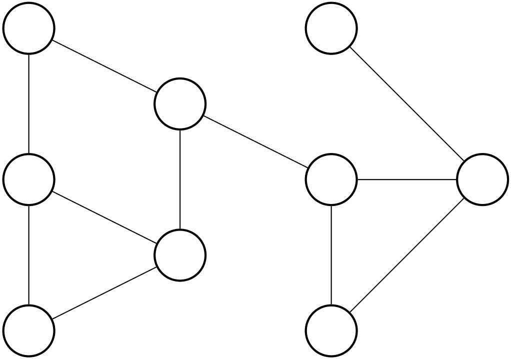
- Við byrjum á að merkja alla hnúta sem „ósnerta", nema við merkjum einn hnút sem „séðan".
- Sá hnútur er kallaður *upphafshnúturinn*.
- Við endurtökum svo sömu skrefin þar til engir „séðir" hnútar eru eftir:
  - Veljum þann „séða" hnút sem við sáum fyrst.
  - Merkjum alla „ósnerta" nágranna hans sem „séða".
  - Merkjum upprunalegu hnútinn „kláraðann".

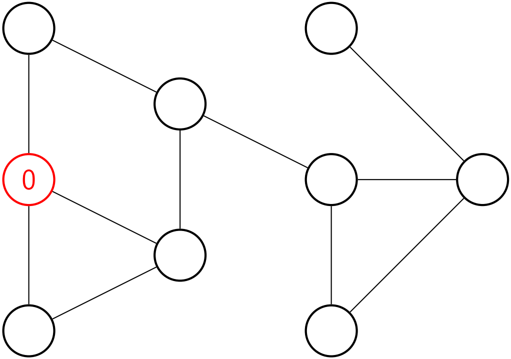
Breiddarleit

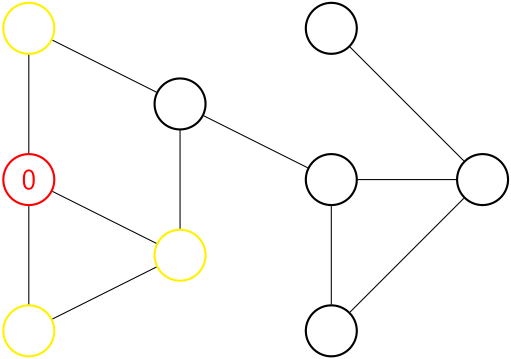
- Við byrjum á að merkja alla hnúta sem „ósnerta", nema við merkjum einn hnút sem „séðan".
- Sá hnútur er kallaður *upphafshnúturinn*.
- Við endurtökum svo sömu skrefin þar til engir „séðir" hnútar eru eftir:
  - Veljum þann „séða" hnút sem við sáum fyrst.
  - Merkjum alla „ósnerta" nágranna hans sem „séða".
  - Merkjum upprunalegu hnútinn „kláraðann".
- Tökum dæmi.

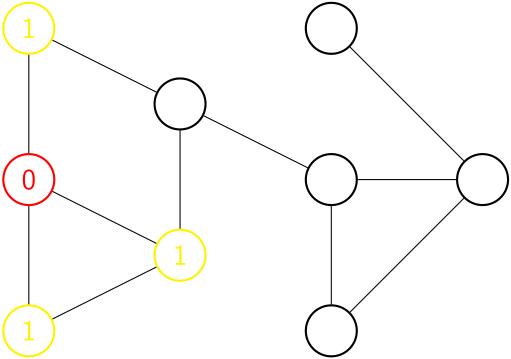
Breiddarleit

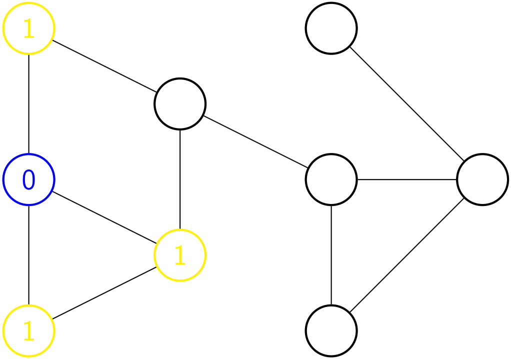
- Við byrjum á að merkja alla hnúta sem „ósnerta", nema við merkjum einn hnút sem „séðan".
- Sá hnútur er kallaður *upphafshnúturinn*.
- Við endurtökum svo sömu skrefin þar til engir „séðir" hnútar eru eftir:
  - Veljum þann „séða" hnút sem við sáum fyrst.
  - Merkjum alla „ósnerta" nágranna hans sem „séða".
  - Merkjum upprunalegu hnútinn „kláraðann".
- Tökum dæmi.
- Við munum merkja „séða" hnúta með hvenær við sáum þá.

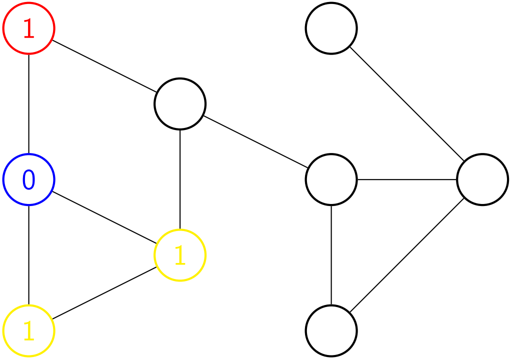


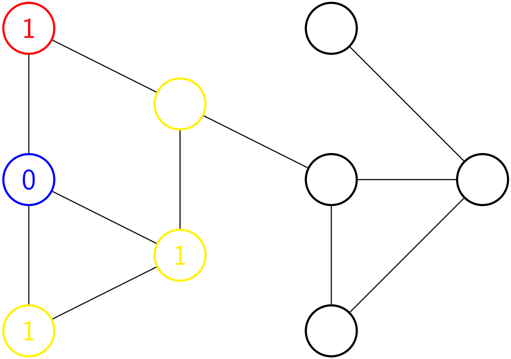

0

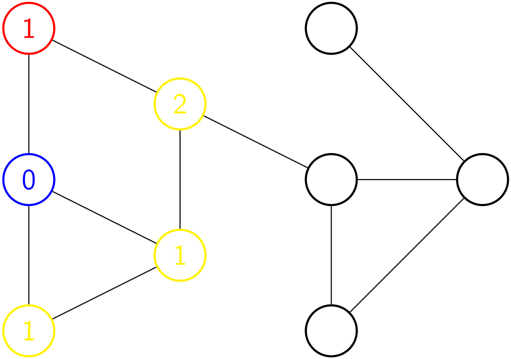

0

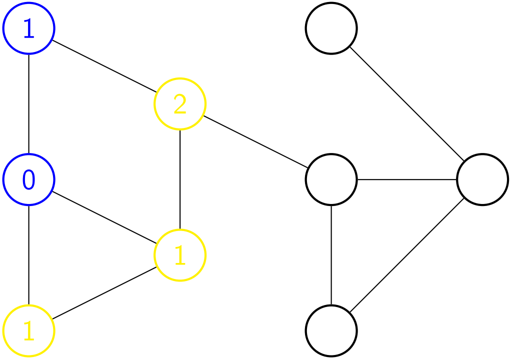

1
0
1
1

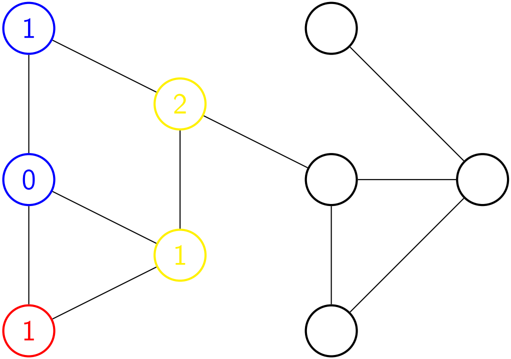

1
0
1
1

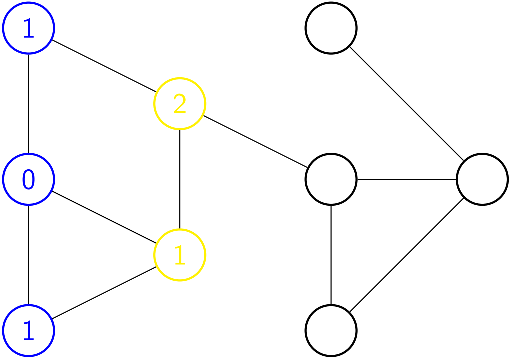

1
0
1
1

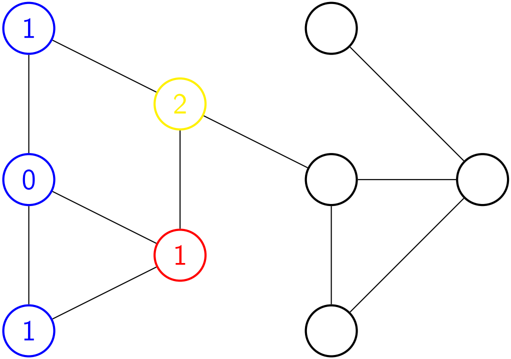

1
0
1
1

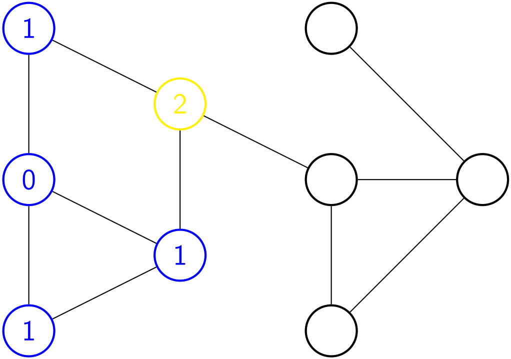

1
2
0
1
1

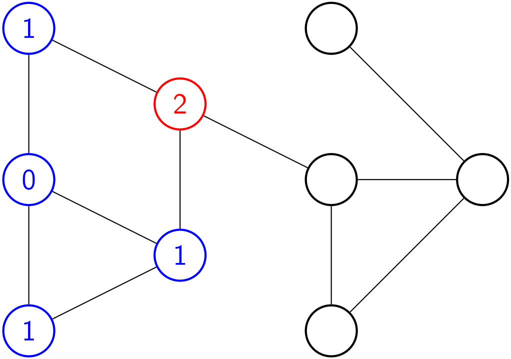

1
2
0
1
1

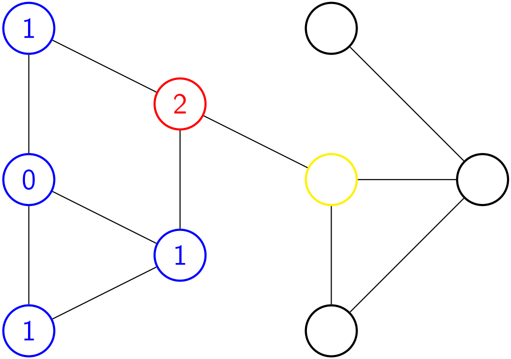

1
2
0
1
1

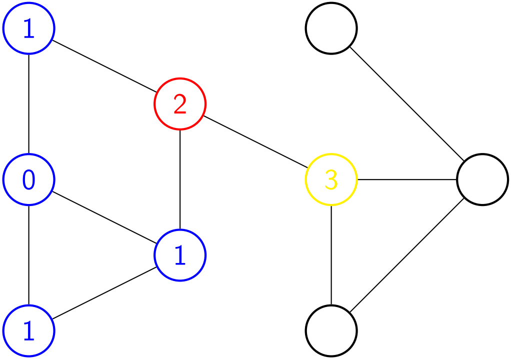

1
2
0
1
1

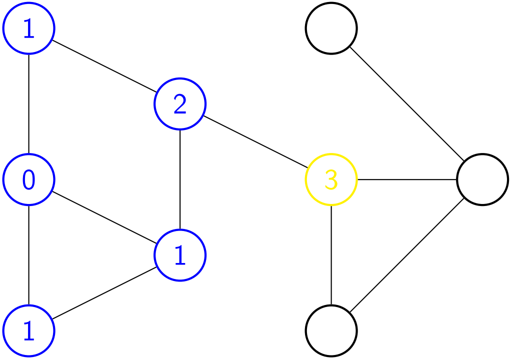

1
2
0
1
1

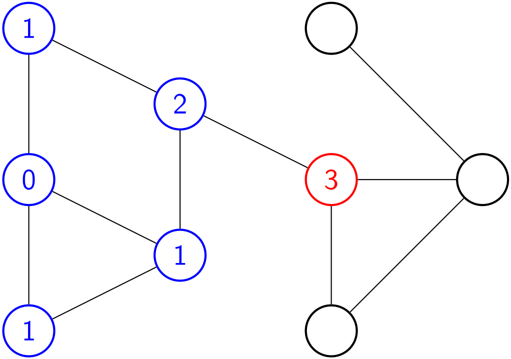

1
2
0
1
1

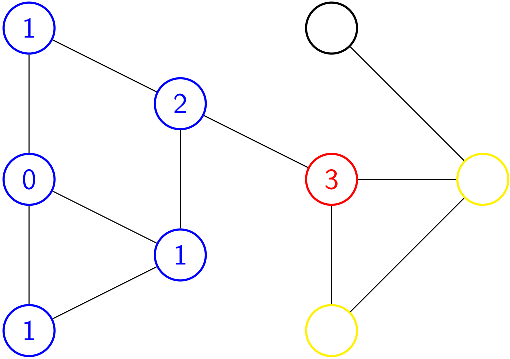

1
2
0
1
1

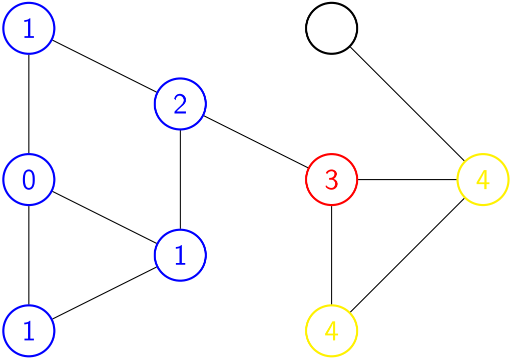

1
2
0
1
1

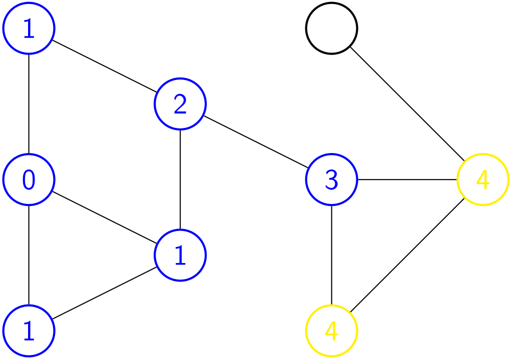

1
2
0
3
1
1

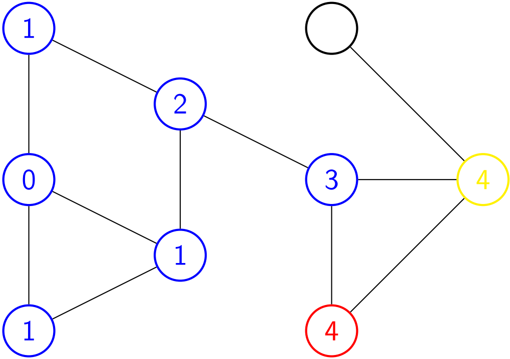

1
2
0
3
1
1

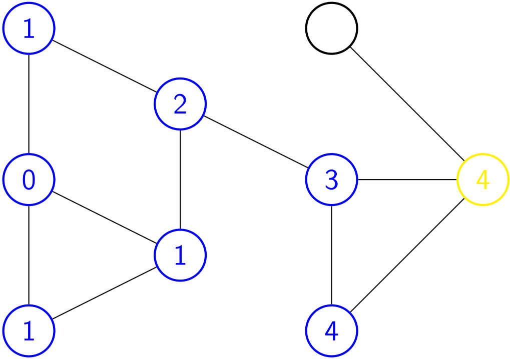

1
2
0
1
1
3

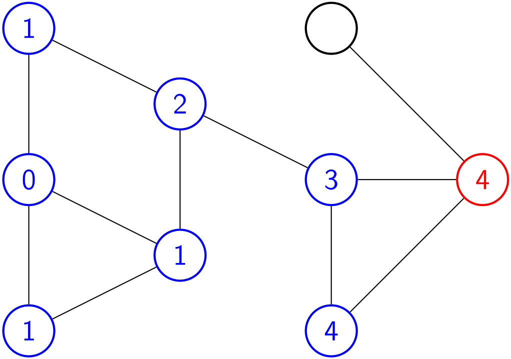

1
2
0
3
1
1

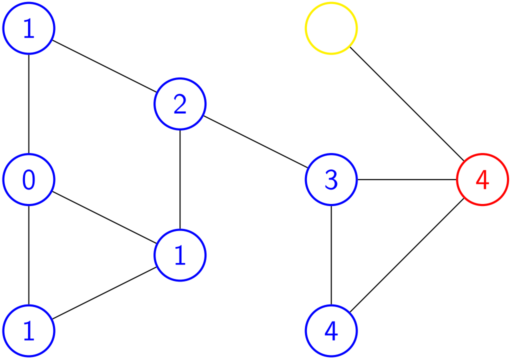

1
2
0
1
1
3
4
4

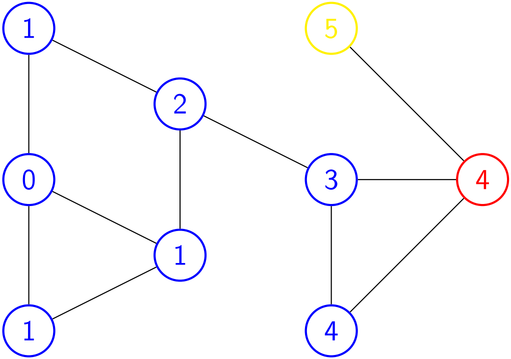

1
2
0
1
1
3
4
4

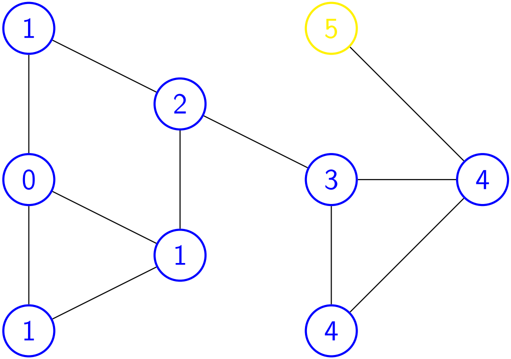

1
2
0
1
1
3
4
4

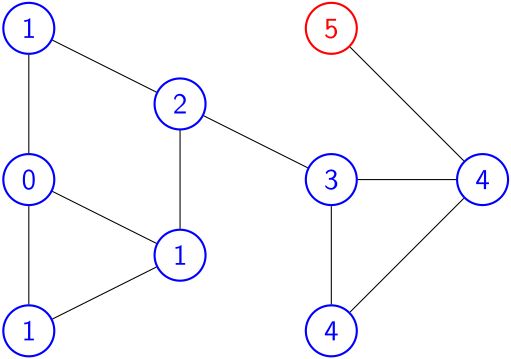

1
2
0
3
4
1
1
4

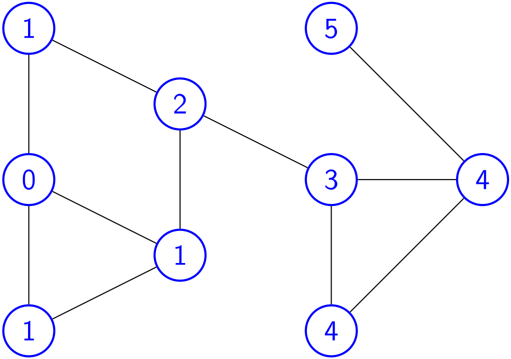

1
2
0
1
1
3
4
4


1
2
0
1
1
3
4
4


1
5
2
0
3
4
1
1
4


1
5
2
0
3
4
1
1
4


1
5
2
0
3
4
1
1
4


1
5
2
0
3
4
1
1
4

- Við munum halda utan um „séða" hnúta með biðröð.

- Við munum halda utan um „séða“ hnúta með biðröð.
- Við byrjum því á að setja upphafshnútinn okkar í biðröðina.

- Við munum halda utan um „séða" hnúta með biðröð.
- Við byrjum því á að setja upphafshnútinn okkar í biðröðina.
- Við tökum svo hnút úr biðröðinni, setjum alla „óséða" nágranna hans í biðröðina og höldum áfram þangað til biðröðin er tóm.

- Við munum halda utan um „séða" hnúta með biðröð.
- Við byrjum því á að setja upphafshnútinn okkar í biðröðina.
- Við tökum svo hnút úr biðröðinni, setjum alla „óséða" nágranna hans í biðröðina og höldum áfram þangað til biðröðin er tóm.

```
vi d(n, -1);
queue<int> q;
q.push(0);
d[0] = 0;
while (q.size() > 0)
{
    int x = q.front();
    q.pop();
    rep(i, g[x].size()) if (d[g[x][i]] == -1)
    {
        q.push(g[x][i]);
        d[g[x][i]] = d[x] + 1;
    }
}
```

- Við segjum að hnútar u og v séu fjarlægð k frá hvorum öðrum ef stysti vegurinn frá u til v er af lengd k.

- Við segjum að hnútar u og v séu fjarlægð k frá hvorum öðrum ef stysti vegurinn frá u til v er af lengd k.
- Við segjum líka að það séu k skref á milli hnútanna.

- Við segjum að hnútar u og v séu fjarlægð k frá hvorum öðrum ef stysti vegurinn frá u til v er af lengd k.
- Við segjum líka að það séu k skref á milli hnútanna.
- Ef enginn vegur er á milli hnútanna segjum við að lengdin á milli þeirra sé ∞.

- Við segjum að hnútar u og v séu fjarlægð k frá hvorum öðrum ef stysti vegurinn frá u til v er af lengd k.
- Við segjum líka að það séu k skref á milli hnútanna.
- Ef enginn vegur er á milli hnútanna segjum við að lengdin á milli þeirra sé ∞.
- Mikilvægur eiginleiki breiddarleitar er að hún heimsækir fyrst þá hnúta sem eru næst upphafshnútnum.

- Við segjum að hnútar u og v séu fjarlægð k frá hvorum öðrum ef stysti vegurinn frá u til v er af lengd k.
- Við segjum líka að það séu k skref á milli hnútanna.
- Ef enginn vegur er á milli hnútanna segjum við að lengdin á milli þeirra sé ∞.
- Mikilvægur eiginleiki breiddarleitar er að hún heimsækir fyrst þá hnúta sem eru næst upphafshnútnum.
- Með öðrum orðum, ef u er k_1 skref frá upphafshnútnum og v er k_2 skref frá upphafshnútunum, $k_1 \neq k_2$, þá heimsækir breiddarleit u á undan v þá og því aðeins að $k_1 < k_2$.

- Við segjum að hnútar u og v séu fjarlægð k frá hvorum öðrum ef stysti vegurinn frá u til v er af lengd k.
- Við segjum líka að það séu k skref á milli hnútanna.
- Ef enginn vegur er á milli hnútanna segjum við að lengdin á milli þeirra sé ∞.
- Mikilvægur eiginleiki breiddarleitar er að hún heimsækir fyrst þá hnúta sem eru næst upphafshnútnum.
- Með öðrum orðum, ef u er k_1 skref frá upphafshnútnum og v er k_2 skref frá upphafshnútunum, $k_1 \neq k_2$, 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 upphafshnútnum að öllum öðrum hnútum.

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

- Líkt og í dýptarleit þá heimsækjum við hvern hnút 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 hnúta má komast frá upphafshnútnum og gera það með sömu tímaflækju.

Samanburður

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

Samanburður

- Báðar leitirnar segja okkur til hvaða hnúta má komast frá upphafshnútnum og gera það með sömu tímaflækju.
- Breiddarleit gefur okkur einnig fjarlægð allra hnúta frá upphafshnútnum.
- 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 hnúta má komast frá upphafshnútnum og gera það með sömu tímaflækju.
- Breiddarleit gefur okkur einnig fjarlægð allra hnúta frá upphafshnútnum.
- 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 enda 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 enda allir á '`X`' ásamt því að fyrsti og síðasti strengurinn inniheldur bara stafinn '`X`'.
- Annars innihalda strengirnir bara stafina '`X`', '`.`' og eitt stykki '`O`'.

- Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c.
- Inntakið inniheldur síðan r strengir, allir af lengd c.
- Strengirnar byrjar og enda allir á ‘`X`’ ásamt því að fyrsti og síðasti strengurinn inniheldur bara stafinn ‘`X`’.
- Annars innihalda strengirnir bara stafina ‘`X`’, ‘`.`’ og eitt stykki ‘`O`’.
- Dæmi um slíkan streng er:

```
11 14
XXXXXXXXXXXXXX
X.X..........X
X.X.XXXX.X.X.X
X.X.....XXXX.X
XOXXXXXX...X.X
X..X.X...X.X.X
X...XX.XXX.X.X
X....X...X.X.X
XXXX.XXX.X.X.X
X........X...X
XXXXXXXXXXXXXX
```

- Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c.
- Inntakið inniheldur síðan r strengir, allir af lengd c.
- Strengirnar byrjar og enda allir á ‘`X`’ ásamt því að fyrsti og síðasti strengurinn inniheldur bara stafinn ‘`X`’.
- Annars innihalda strengirnir bara stafina ‘`X`’, ‘`.`’ og eitt stykki ‘`O`’.
- Dæmi um slíkan streng er:

```
11 14
XXXXXXXXXXXXXX
X.X..........X
X.X.XXXX.X.X.X
X.X.....XXXX.X
XOXXXXXX...X.X
X..X.X...X.X.X
X...XX.XXX.X.X
X....X...X.X.X
XXXX.XXX.X.X.X
X........X...X
XXXXXXXXXXXXXX
```

- Við viljum svo prenta sama borð, nema í stað bókstafana á að koma hversu fá skref við þurfum að taka til frá ‘`O`’ til að komast þangað ef við megum ferðast upp, niður, til hægri og til vinstri, en ekki á reitunum með ‘`X`’.

- Fyrsta lína inntaksins inniheldur tvær heiltölur, r og c.
- Inntakið inniheldur síðan r strengir, allir af lengd c.
- Strengirnar byrjar og enda allir á ‘`X`’ ásamt því að fyrsti og síðasti strengurinn inniheldur bara stafinn ‘`X`’.
- Annars innihalda strengirnir bara stafina ‘`X`’, ‘`.`’ og eitt stykki ‘`O`’.
- Dæmi um slíkan streng er:

```
11 14
XXXXXXXXXXXXXX
X.X..........X
X.X.XXXX.X.X.X
X.X.....XXXX.X
XOXXXXXX...X.X
X..X.X...X.X.X
X...XX.XXX.X.X
X....X...X.X.X
XXXX.XXX.X.X.X
X........X...X
XXXXXXXXXXXXXX
```

- Við viljum svo prenta sama borð, nema í stað bókstafana á að koma hversu fá skref við þurfum að taka til frá ‘`O`’ til að komast þangað ef við megum ferðast upp, niður, til hægri og til vinstri, en ekki á reitunum með ‘`X`’.
- Fyrir þá reiti sem við komumst ekki á prentum við -1.

- Sem dæmi hefur inntakið

```
11 14
XXXXXXXXXXXXXX
X.X..........X
X.X.XXXX.X.X.X
X.X.....XXXX.X
XOXXXXXX...X.X
X..X.X...X.X.X
X...XX.XXX.X.X
X....X...X.X.X
XXXX.XXX.X.X.X
X........X...X
XXXXXXXXXXXXXX
```

úttakið

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

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

- Við getum túlkað þessa mynd sem net.
- Ímyndum okkur að hver auður reitur sé hnútur.
- Við tengjum svo aðliggjandi auða hnúta með leggjum.
- Þar sem við viljum finna fjalægðir frá tilteknum hnút til allra annara hnúta notum við leit.

- Við getum túlkað þessa mynd sem net.
- Ímyndum okkur að hver auður reitur sé hnútur.
- Við tengjum svo aðliggjandi auða hnúta með leggjum.
- Þar sem við viljum finna fjalægðir frá tilteknum hnút til allra annara hnúta notum við breiddarleit.

- Við getum túlkað þessa mynd sem net.
- Ímyndum okkur að hver auður reitur sé hnútur.
- Við tengjum svo aðliggjandi auða hnúta með leggjum.
- Þar sem við viljum finna fjalægðir frá tilteknum hnút til allra annara hnúta notum við breiddarleit.
- Fjöldi hnúta 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 hnútur.
- Við tengjum svo aðliggjandi auða hnúta með leggjum.
- Þar sem við viljum finna fjalægðir frá tilteknum hnút til allra annara hnúta notum við breiddarleit.
- Fjöldi hnúta 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 $\mathcal{O}(\qquad\qquad) = \mathcal{O}(\qquad)$.

- Við getum túlkað þessa mynd sem net.
- Ímyndum okkur að hver auður reitur sé hnútur.
- Við tengjum svo aðliggjandi auða hnúta með leggjum.
- Þar sem við viljum finna fjalægðir frá tilteknum hnút til allra annara hnúta notum við breiddarleit.
- Fjöldi hnúta 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 $\mathcal{O}(E + V) = \mathcal{O}(\quad)$.

- Við getum túlkað þessa mynd sem net.
- Ímyndum okkur að hver auður reitur sé hnútur.
- Við tengjum svo aðliggjandi auða hnúta með leggjum.
- Þar sem við viljum finna fjalægðir frá tilteknum hnút til allra annara hnúta notum við breiddarleit.
- Fjöldi hnúta 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 $\mathcal{O}(E + V) = \mathcal{O}(r \cdot c)$.

```cpp
#include <bits/stdc++.h>
#define rep(E, F) for (E = 0; E < (F); E++)
using namespace std;
typedef pair<int, int> ii;

int isin(int x, int y, int r, int c)
{ // Segir okkur hvort (x, y) sé fyrir utan myndina okkar.
    if (x >= r || x < 0 || y >= c || y < 0) return 0;
    return 1;
}

int main()
{
    int i, j, r, c, x, y, g[4][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
    cin >> r >> c;
    char a[r][c + 1];
    rep(i, r) cin >> a[i];
    rep(i, r) rep(j, c) if (a[i][j] == 'O') x = i, y = j;
    int d[r][c];
    rep(i, r) rep(j, c) d[i][j] = -1;
    queue<ii> q;
    q.push(ii(x, y)), d[x][y] = 0;
    while (q.size() > 0)
    {
        ii p = q.front(); q.pop();
        x = p.first, y = p.second;
        rep(i, 4)
        {
            int xx = x + g[i][0], yy = y + g[i][1];
            if (a[xx][yy] == 'X' || !isin(xx, yy, r, c) || d[xx][yy] != -1)
                continue;
            q.push(ii(xx, yy)), d[xx][yy] = d[x][y] + 1;
        }
    }
    rep(i, r) { rep(j, c) printf("%2d ", d[i][j]); printf("\n"); }
    return 0;
}
```

Tæmandi leit á vegi

- Við getum breitt dýptarleitar útfærslunni okkar lítillega til að finna alla einfalda vegi í neti sem byrja í tilteknum hnút.

Tæmandi leit á vegi

- Við getum breitt dýptarleitar útfærslunni okkar lítillega til að finna alla einfalda vegi í neti sem byrja í tilteknum hnút.
- Til að koma í veg fyrir að heimsækja hnút oftar en einu sinni í dýpterleit merkjum við hann og heimsækjum ekki merkta hnúta.

Tæmandi leit á vegi

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

Tæmandi leit á vegi

- Við getum breitt dýptarleitar útfærslunni okkar lítillega til að finna alla einfalda vegi í neti sem byrja í tilteknum hnút.
- Til að koma í veg fyrir að heimsækja hnút oftar en einu sinni í dýpterleit merkjum við hann og heimsækjum ekki merkta hnúta.
- Við munum ennþá þurfa að merkja hnúta því við erum að leita að einföldum vegum (almennt er ekki takmarkaður fjöldi vega í neti).
- Munurinn er að við munum merkja hnút þegar við sjáum hann, halda áfram endurkvæmt til ómerktra nágranna hans og afmerkja hann svo.

```
vi v, p;
void dfs(vvi& g, int x)
{
    int i, f = 1;
    v[x] = 1;
    p.push_back(x);
    rep(i, g[x].size()) if (v[g[x][i]] == 0)
        dfs(g, g[x][i]), f = 0;
    if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
    p.pop_back();
    v[x] = 0;
}
```

```
vi v, p;
void dfs(vvi& g, int x)
{
    int i, f = 1;
    v[x] = 1;
    p.push_back(x);
    rep(i, g[x].size()) if (v[g[x][i]] == 0)
        dfs(g, g[x][i]), f = 0;
    if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
    p.pop_back();
    v[x] = 0;
}
```

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

```
vi v, p;
void dfs(vvi& g, int x)
{
    int i, f = 1;
    v[x] = 1;
    p.push_back(x);
    rep(i, g[x].size()) if (v[g[x][i]] == 0)
        dfs(g, g[x][i]), f = 0;
    if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
    p.pop_back();
    v[x] = 0;
}
```

- 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 hnúta eru nágrannar.

```
vi v, p;
void dfs(vvi& g, int x)
{
    int i, f = 1;
    v[x] = 1;
    p.push_back(x);
    rep(i, g[x].size()) if (v[g[x][i]] == 0)
        dfs(g, g[x][i]), f = 0;
    if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
    p.pop_back();
    v[x] = 0;
}
```

- 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 hnúta eru nágrannar.
- Þá myndi þetta forrit prenta allar umraðanir.

```
vi v, p;
void dfs(vvi& g, int x)
{
    int i, f = 1;
    v[x] = 1;
    p.push_back(x);
    rep(i, g[x].size()) if (v[g[x][i]] == 0)
        dfs(g, g[x][i]), f = 0;
    if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
    p.pop_back();
    v[x] = 0;
}
```

- 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 hnúta eru nágrannar.
- Þá myndi þetta forrit prenta allar umraðanir.
- Tímaflækjan er því $\mathcal{O}($ $)$.

```
vi v, p;
void dfs(vvi& g, int x)
{
    int i, f = 1;
    v[x] = 1;
    p.push_back(x);
    rep(i, g[x].size()) if (v[g[x][i]] == 0)
        dfs(g, g[x][i]), f = 0;
    if (f) { rep(i, p.size()) printf("%d ", p[i] + 1); printf("\n"); }
    p.pop_back();
    v[x] = 0;
}
```

- 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 hnúta eru nágrannar.
- Þá myndi þetta forrit prenta allar umraðanir.
- Tímaflækjan er því $\mathcal{O}((V+1)!)$.