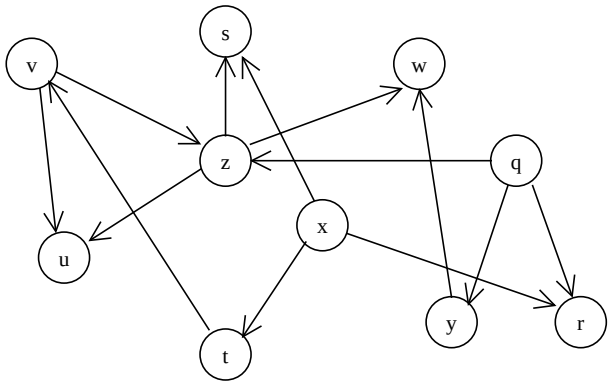
Tietorakenteet, laskuharjoitus 11, 19-23.4.

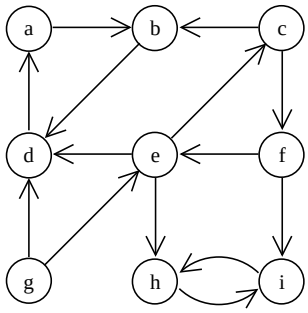
1. Monisteessa leveyssuuntainen läpikäynti on toteutettu iteratiivisesti (ilman rekursiota), kun taas syvyyssuuntainen läpikäynti on toteutettu rekursiivisesti. Esitä ideatasolla, miten leveyssuuntaisen läpikäynnin voi toteuttaa rekursiivisesti ja syvyyssuuntaisen läpikäynnin iteratiivisesti.

2. Järjestä kuvan verkko topologisesti käyttäen luennolla esitettyä menetelmää. Oleta, että solmut ovat vieruslistoissa aakkosjärjestyksessä.



3. Muodosta allaolevan verkon vahvasti yhtenäiset komponentit luennolla esitetyllä algoritmilla. Oleta, että kaaret on talletettu vieruslistoihin aakkosjärjestyksessä.

   Mikä on pienin määrä lisättäviä kaaria, jolla alkuperäisestä verkosta saataisiin vahvasti yhtenäinen?



4. Jos verkko tallennetaan vieruslistoina, se vie vähän tilaa muistissa, mutta on hidasta tarkistaa, kuuluuko tietty kaari verkkoon. Jos verkko tallennetaan vierusmatriisina, se vie paljon tilaa muistissa, mutta on nopeaa tarkistaa, kuuluuko tietty kaari verkkoon. Suunnittele kolmas tapa tallentaa verkko, joka sekä vie vähän muistia että mahdollistaa nopean kaaren tarkistuksen. Tavoite: tilankäyttö $O(|E| + |V|)$, kaaren tarkistus ajassa $O(\log |V|)$. Onko ratkaisussasi mitään huonoa?

5. Esitä algoritmi, joka tarkistaa, onko suuntaamaton verkko *kaksijakoinen* eli voiko sen solmut värittää kahdella värillä niin, että jokainen kaari yhdistääkaksi eriväristä solmua.

6. Tee ohjelma, joka laskee, kuinka monta huonetta talossa on. Syötteenä algoritmille annetaan kaksiulotteisena merkkitaulukkona esitetty talon kuvaus, jossa # tarkoittaa seinää ja . tarkoittaa lattiaa.

   Syöte voi olla esim:

```
############
#..#..#....#
#..#..#....#
#####.#.#..#
#...#...#..#
###.########
#.....#....#
############
```

Yllä kuvatussa talossa on yhteensä 4 huonetta

Rastiin riittää myös huolellisesti mietitty tarkka pseudokoodi.

7. **Bonustehtävä jolla voi korvata 2 muuta tehtävää joko näistä tai aiemmista laskareista**

   Ohjelma saa syötteeksi joukon riippuvuuksia. Syötteenä voisi olla esim. kurssien suoritusjärjestysten keskinäiset riippuvuudet:

   ```
   ohpe ohja
   ohja tira
   tira lama
   tira tiralabra
   ohja javalabra
   javalabra tiralabra
   ohpe ohma
   tiralabra ohtuprojekti
   javalabra ohtuprojekti
   ohma ohtu
   ohtu ohtuprojekti
   lama ohtuprojekti
   ohtuprojekti ohja
   ```

   Eli jokaisella rivillä on ilmaistu yksi riippuvuus, esim. `ohja tira` määrittelee, että `ohja` täytyy suorittaa ennen `tira`:a.

   Ohjelma tarkistaa ensin, etteivät riippuvuudet muodosta sykliä. Esim. jos edelliseen lisättäisiin

   ```
   ohtuprojekti tira
   ```

   Niin muodostuisi riippuvuussykli:

   ```
   ohtuprojekti tira tiralabra ohtuprojekti
   ```

   Riippuvuussykliä ei ole pakko tulostaa. Riittää, että ohjelma ilmoittaa että sykli löytyi. Tyylikkäintä on tietysti tulostaa myös sykli.

   Jos riippuvuussykliä ei ole, tulostaa ohjelma *yhden* mahdollisen peräkkäisen järjestyksen, joka ei riko yhtään riippuvuutta.

   Tee toteutus haluamallasi ohjelmointikielellä.