Vastaa tentin järjestäjän antamalle paperille. Kirjoja, laskinta tms. ei saa olla tentissä. Kukin tehtävä on 1 tai 2 pisteen arvoinen. Muiden kuin koodaa tai piirrä -tehtävien mallivastaukset ovat melko lyhyet, tyypillisesti 1, ..., 3 riviä.

1. Millä nimellä alla oleva algoritmi (ja sen kaltaiset) tunnetaan?

   puolitushaku

```
1  int ala = 0, yla = A.size();
2  while( ala < yla ){
3    int vali = (ala + yla)/2;
4    if( A[vali] >= x ){ yla = vali; }else{ ala = vali+1; }
5  }
```

2. Mikä ehto `A`:n täytyy toteuttaa, jotta algoritmi tekisi mitä sen pitää tehdä?

   `A`:n pitää olla kasvavassa suuruusjärjestyksessä.

3. Miksi rivillä 1 lukee `yla = A.size();` eikä `yla = A.size() - 1;`?

   Jotta algoritmi tuottaisi tulokseksi `A.size()` silloin, kun `x` on suurempi kuin mikään `A`:n alkio.

4. Oletetaan, että riviltä 4 poistetaan "`+1`". Anna esimerkki syötteestä, jolla algoritmi toimii väärin, ja kerro miten se toimii väärin.

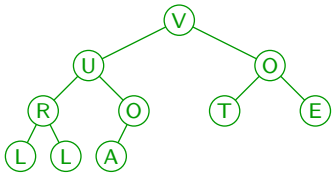
   Jos $\texttt{A} = [0,2]$ ja $\texttt{x} = 1$, niin se jää ikuiseen silmukkaan, jossa se sijoittaa toistuvasti nollan `vali`:in ja `ala`:an.

Oletetaan, että keossa on suurin ylinnä.

5. Mitkä palvelut keko tarjoaa nopeasti?

   Suurimman alkion palauttamisen ja poistamisen sekä alkion lisäämisen.

6. Piirrä $[\mathsf{V},\mathsf{U},\mathsf{O},\mathsf{R},\mathsf{O},\mathsf{T},\mathsf{E},\mathsf{L},\mathsf{L},\mathsf{A}]$ puuna ikään kuin se olisi keko.



7. Perustele, että $[\mathsf{V},\mathsf{U},\mathsf{O},\mathsf{R},\mathsf{O},\mathsf{T},\mathsf{E},\mathsf{L},\mathsf{L},\mathsf{A}]$ ei ole keko.

   T:n vanhempi on $\mathsf{O} < \mathsf{T}$, vastoin keon vaatimuksia.

8. Jos keon korkeus on h, niin kuinka monta solmua siinä on vähintään ja enintään?

   vähintään 2^h ja enintään $2^{h+1} - 1$

Sekalaisia kysymyksiä

9. Kirjoita mikä tahansa ohjelmanpätkä, jonka suoritusaika on hitaimmillaan $\Theta(n^2)$ ja nopeimmillaan $\Theta(n)$.

```
1  for( i=0; i<n; ++i ){
2    if( A[i] > 0 ){
3      for( j=0; j<n; ++j ){ B[j] += A[i]; }
4    }
5  }
```

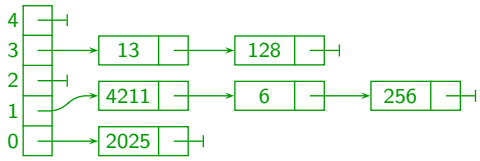
10. Kirjoita tälle invariantti: `i = 0; while( i < n && A[i] < 0 ){ ++i; }`

    $0 \leq i \leq n$ ja osataulukon $A[0 \ldots i-1]$ alkiot ovat negatiivisia.

11. Miksi ihmisen nimen ensimmäisen kirjaimen käyttö hajautusfunktiona on huono ajatus?

    Joihinkin listoihin osuu paljon nimiä, joihinkin tuskin lainkaan.

12. Hajautusfunktiona on n mod 5. Piirrä hajautustaulu, jossa on 4211, 13, 6, 2025, 128 ja 256.



13&14. Miksi hajautuslistojen määrän ei pidä olla liian suuri ja miksi ei liian pieni?

    Liian suuri tuhlaa muistia kantataulukossa ilman vastaavaa nopeutushyötyä. Liian pieni hidastaa hajautustaulun toimintoja, koska monet alkiot osuvat pitkiin listoihin, joita joudutaan selaamaan.

15. Mitä tarkoittaa ahne algoritmi?

    Ratkaisu rakennetaan vähän kerrassaan siten, että aina tehdään sillä hetkellä lupaavimmalta näyttävä lisäys ilman kokonaisuuden tarkastelua.

16. Mitä tarkoittaa dynaaminen ohjelmointi?

    Ratkaisun rakentamisessa tarvittavia välituloksia talletetaan esim. taulukkoon, jotta kun samaa tarvitaan uudelleen, sitä ei tarvitse laskea uudelleen.

17. Kerro lyhyesti, miten kasvava taulukko (esimerkiksi C++:n `vector`) toimii.

    Kun taulukolle varattu muisti loppuu, varataan kaksinkertainen muistialue ja siirretään taulukon sisältö sinne.

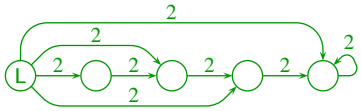
Olkoon $|V|$ suunnatun graafin solmujen määrä ja $|E|$ kaarten määrä. Seuraavaksi on alla olevaa algoritmia koskevia kysymyksiä. Muuttujat on alustettu järkevästi.

```
1  solmut[ lahto ].etaisyys = 0;
2  unsigned kierros = 1; bool muuttui = true;
3  while( kierros <= solmut.size() && muuttui ){
4    ++kierros; muuttui = false;
5    for( unsigned kk = 0; kk < kaaret.size(); ++kk ){
6      kaarityyppi & KK = kaaret[ kk ]; solmutyyppi & SS = solmut[ KK.karki ];
7      if( solmut[ KK.hanta ].etaisyys == aareton ){ continue; }
8      pituustyyppi uusi = solmut[ KK.hanta ].etaisyys + KK.pituus;
9      if( uusi < SS.etaisyys ){
10       SS.etaisyys = uusi; SS.reitti = kk; muuttui = true;
11     }
12   }
13 }
```

18. Piirrä graafi, jossa on 5 solmua ja 8 kaarta, jossa `lahto`:sta on reitti jokaiseen solmuun, ja jolla algoritmi on hidas. Kerro, miksi se on hidas.

Uloin silmukka kiertää $|V|$ kertaa, koska `muuttui` muuttuu sen jokaisella kierroksella paitsi viimeisellä.

19. Piirrä graafi, jossa on 5 solmua ja 8 kaarta, jossa `lahto`:sta on reitti jokaiseen solmuun, ja jolla algoritmi on nopea. Kerro, miksi se on nopea.



Uloin silmukka kiertää 2 kertaa, koska `muuttui` lakkaa muuttumasta.

20. Ilmoita algoritmin suoritusaika O-merkinnällä ja Ω-merkinnällä.

$O(|V||E|)$ ja $\Omega(|E|)$

21. Perustele tehtävän 20 O-vastauksesi.

Kaikki muu on vakioaikaista paitsi silmukat. Jokaisella ulomman silmukan kierroksella sisempi silmukka kiertää $|E|$ kierrosta. Ulomman silmukan kierrosten määrä on enintään $|V|$. Se on $|V|$ esim. tehtävän 18 tapauksessa.

22. Perustele tehtävän 20 Ω-vastauksesi.

Kaikki muu on vakioaikaista paitsi silmukat. Jokaisella ulomman silmukan kierroksella sisempi silmukka kiertää $|E|$ kierrosta. Ulomman silmukan kierrosten määrä on vähintään 1. Se on vakio esim. tehtävän 19 tapauksessa.

23. Miten algoritmin lopetettua voidaan päätellä, onko solmusta `lahto` reittiä solmuun `maali`?

```
solmut[ maali ].etaisyys != aareton
```

24. Kirjoita rekursiivinen aliohjelma, joka tulostaa mahdollisimman lyhyen reitin solmujen numerot `lahto` ensin ja `maali` viimeisenä. Saat luottaa, että mahdollisimman lyhyt reitti on olemassa.

```
1  void tulosta( unsigned ss ){
2    if( ss != lahto ){ tulosta( kaaret[ solmut[ ss ].reitti ].hanta ); }
3    std::cout << ' ' << ss;
4  }
```

25. Piirrä graafi, jossa ei ole mahdollisimman lyhyttä reittiä lähdöstä itseensä.

26. Miten algoritmin lopetettua voidaan päätellä, onko solmusta `lahto` mahdollisimman lyhyttä reittiä solmuun `maali`?

    Vastaus 23 eli jokin reitti on olemassa, ja vastauksen 24 tyyliin takaperin kuljettaessa jokaiselle kaarelle `KK` pätee `solmut[ KK.hanta ].etaisyys + KK.pituus` = `solmut[ KK.karki ].etaisyys` kunnes kohdataan `lahto`.

Binäärihakupuun solmussa on kentät `vasen`, `oikea`, `avain` ja (solmusta alkavan alipuun) `koko`. Aluksi `ss` osoittaa juurta. Jos seuraavissa tehtävissä mainittua solmua ei ole olemassa, niin algoritmin jälkeen `ss`:n pitää osoittaa ei minnekään.

27. Kirjoita algoritmi, jonka jälkeen `ss` osoittaa puun viimeistä solmua.

```
1  if( ss ){
2    while( ss->oikea ){ ss = ss->oikea; }
3  }
```

28. Kirjoita algoritmi, jonka jälkeen `ss` osoittaa solmua, jonka avain on `x`.

```
1  while( ss && ss->avain != x ){
2    if( ss->avain > x ){ ss = ss->vasen; }else{ ss = ss->oikea; }
3  }
```

29&30. Kirjoita algoritmi, jonka jälkeen `ss` osoittaa solmua, jonka edellä puussa on `ii` solmua.

```
1  while( ss ){
2    unsigned mm = 0;
3    if( ss->vasen ){ mm = ss->vasen->koko; }
4    if( ii == mm ){ break; }
5    if( ii < mm ){ ss = ss->vasen; }
6    else{ ii -= mm+1; s1 = s1->oikea; }
7  }
```

loppu