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 korkeintaan 2 riviä.

Erään algoritmin ajan kulutus on $1 + V \log_2(E+1)$, missä V on risteysten ja E niiden välisten tienpätkien määrä. Kunnes toisin sanotaan, kysytään tarkkaa arvoa eikä esim. $O(\ldots)$-vastausta.

1. Eräässä kylässä $V = 5$ ja $E = 15$. Paljonko algoritmi kuluttaa aikaa?

2. Eräässä kaupungissa $E = 4V - 1$. Sievennä ajan kulutus muotoon, jossa E ei esiinny.

3. Toisessa kaupungissa $E \leq (V+1)^2 - 1$. Ilmoita algoritmin ajan kulutukselle mahdollisimman hyvä yläraja, jossa E ei esiinny.

4. Ilmoita algoritmin ajan kulutus $O(\ldots)$-merkinnällä, jossa esiintyy sekä V että E. Ilmoita se myös niin, että $E \leq (V+1)^2 - 1$ ja vain V esiintyy. Isoilla x pätee $\log_2(x+1) \approx \log_2 x + \frac{1{,}44}{x}$. Tarpeettoman monimutkaisesta vastauksesta vähennetään pisteitä.

Polynomin arvon laskeminen suoraviivaisesti

5. Ilmoita algoritmin POLYNOMI1 ajan kulutus Θ-merkinnällä.

```
POLYNOMI1(A, x)
y := 0
for i := 0 to n do
    y := y + A[i] · POTENSSI(x, i)
```

```
POTENSSI(x, n)
y := 1
for i := 1 to n do y := y · x
return y
```

Polynomin arvo annetulla x:n arvolla voidaan laskea myös seuraavasti:
$((\ldots((A[n]x) + A[n-1])x + \ldots + A[2])x + A[1])x + A[0]$.

6. Esitä polynomi $7x^5 - 5x^4 + x^2 - 8x + 11$ edellä kuvatussa muodossa.

7. Olkoon $s_i = A[0] + A[1]x + \ldots + A[i]x^i$. Kuinka paljon on s_1? Entä s_{-1}?

8. Oheisen algoritmin silmukkainvariantti on $s_n = yx^i + s_{i-1}$. Perustele, että se on voimassa kun riville 2 tullaan ensimmäisen kerran.

9. Kirjoita silmukkainvarianttia vastaava kaava, joka on voimassa rivin 3 lopussa. Tee sama rivien 4 ja 5 lopuille.

```
1  y := A[n] ; i := n
2  while i > 0 do
3      i := i − 1
4      y := y · x
5      y := y + A[i]
```

10. Perustele, että algoritmin lopputulos on polynomin arvo annetulla x:n arvolla.

11. Ilmoita algoritmin ajan kulutus Θ-merkinnällä.

Keko ja koodinlukutaito

12. Etsi ohjelmointivirhe oheisesta aliohjelmasta.

13. Kun keon kaikki alkiot ovat yhtäsuuret, on aliohjelma erityisen nopea. Minkä yksityiskohdan ansiota tämä on, ja kuinka nopea aliohjelma on silloin?

```
1  void poista_keosta_suurin(){
2    if( keko.size() < 1 ){ return; }
3    unsigned h = keko.size() - 1, i = 0, j = 1;
4    while( true ){
5      if( j+1 < h && keko[j+1] <= keko[j] ){ ++j; }
6      if( j >= h || keko[j] <= keko[h] ){ break; }
7      keko[i] = keko[j]; i = j; j = 2*i + 1;
8    }
9    keko[i] = keko[h]; keko.resize(h);
10 }
```

Käännä

Graafiaalgoritmi- ja hajautustaulukysymyksiä

14. Mikä etu Dijkstran algoritmilla on A*-algoritmiin verrattuna?

15. Mikä etu A*-algoritmilla on Dijkstran algoritmiin verrattuna?

16. Mikä etu Bellman–Ford-algoritmilla on kurssin muihin reitinetsintäalgoritmeihin verrattuna?

17. Hajautusfunktiona on n mod 5. Piirrä hajautustaulu, jossa on 4211, 29, 8, 2025, 128 ja 256.

18. Hajautustaulun kantataulukon koko on k ja alkioita on n. Kuinka monta osoitinta siinä on?

Lanttilassa käytettävien kolikoiden arvot ovat kokonaislukuja. Ne on kerrottu aidosti kasvavassa järjestyksessä taulukossa `arvot`, ja `arvot[0] == 1`.

19. Kirjoita ahne ohjelmanpätkä, jolle annetaan muodostettava rahasumma muuttujassa `tavoite` ja joka kertoo mukaan otettavat kolikot! Saat luottaa, että `tavoite >= 0`. Vastaus talletetaan taulukkoon `maarat`, jota indeksoidaan kolikon arvolla ja joka on alustettu tarpeeksi suureksi ja täyteen nollaa.

20. Osoita esimerkillä, että ahne algoritmi ei välttämättä johda pienimpään määrään kolikoita!

21. Kirjoita rekursiivinen aliohjelma, joka selvittää, mikä on pienin määrä kolikoita, jolla `tavoite` voidaan muodostaa. Se saa olla hyvin hidas. Sen ei tarvitse täyttää taulukkoa `maarat` (se lisätään seuraavassa kohdassa).

22. Lisää edellisen kohdan vastaukseen taulukon `maarat` täyttäminen!

23. Edellinen ohjelma on hidas jo pienehköilläkin tavoitteilla, kuten 100. Nopeuden parantamiseksi lisää siihen jo laskettujen vastausten tallettaminen!

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.

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

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

26&27. Kirjoita algoritmi, jonka jälkeen `ss` osoittaa solmua, jonka edellä puussa on `ii` solmua.

Syötteessä on n alkiota, joista pitää löytää k pienintä. Luku k on suuri ja n on sitäkin paljon suurempi.

28&29. Kuvaile nopea algoritmi, ja kerro sen ajan kulutus. Muistia on käytettävissä paljon. (Ei tarvitse kirjoittaa pseudokoodia tms.) Mitä nopeampi algoritmi, sen enemmän pisteitä.

30. Kuten edellinen, mutta nyt muistia on käytettävissä vain $\Theta(k)$. Data tulee syötteessä alkio kerrallaan, eikä muisti riitä kaiken sen tallettamiseen yhtäaikaa.

loppu