1. Numbrinäit

1 sekund 20 punkti

Lahendused `naitlah1.c`, `naitlah1.pas` ja `naitlah1.py` töötavad kõik samal moel:

- Programmi teksti on tabelina kirjutatud korras tablool põlevate segmentide kirjeldus iga numbri jaoks.
- Vaadatakse läbi numbrid 0...9 ja iga numbri juures kontrollitakse, kas sisendfailist loetud nähtavate segmentide kirjeldus sobib selle numbri ettenähtud segmentidega.

Numbri sobivuse kontrolliks vaadeldakse kõiki segmente ühekaupa:

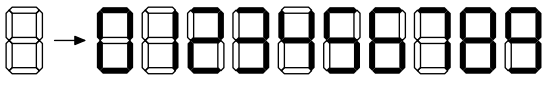
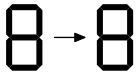
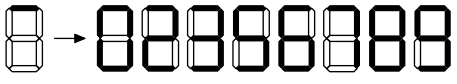
- Kui tablool segment põleb ja numbris peakski põlema, siis on tabloo seal kohas korras ja number on võimalik.
- Kui tablool segment ei põle ja numbris ei peakski põlema, siis ei ole teada, kas tabloo on korras, aga number on võimalik.
- Kui tablool segment ei põle, aga numbris peaks põlema, siis võib tabloo see segment läbi põlenud olla, aga number on katkisel tablool võimalik.
- Kui tablool segment põleb, aga numbris ei peaks põlema, siis ei ole see number isegi läbipõlenud segmentidega tablool võimalik.

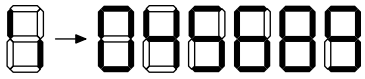
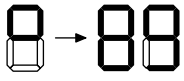
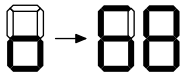
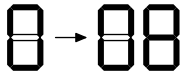
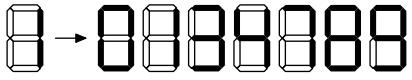
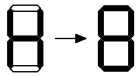
Huvitav on ainult viimane variant, sest see tähendab, et vaadeldav number ei saa olla sisendis toodud segmentidega tablool. Kõigil muul juhtudel on number endiselt võimalik.

Kõik numbrid, mille segmentides ei leitud vastuolusid, väljastatakse.

Lahenduse algoritmiline keerukus on $O(\text{võimalikke numbreid} \cdot \text{segmente tablool})$. Kuna see on ainult $10 \cdot 7 = 70$, siis töötavad kõik lahendused välkkiirelt.

Testid

1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 

10. 
11. 
12. 
13. 
14. 
15. 
16. 
17. 
18. 
19. 
20. 

20 testi, à 1 punkt, kokku 20 punkti.

2. Tööplaan

1 sekund 40 punkti

Selle ülesande lahendamiseks on kasulik kõigepealt tähele panna, et pool sisendandmetest on liigne: mistahes tööd saab kohe tegema hakata siis, kui sellel tööl ei ole ühtki eeldust, see tähendab, kui see töö ei esine sisendis üheski (A_i, B_i) paaris B_i kohal. Selle tingimuse kontrollimiseks on mitmeid erineva efektiivsusega võimalusi.

Kõige lihtsam on iga töö kohta B_i väärtuste jada läbi käia ja selle töö olemasolu vahetult kontrollida. Selline lahendus vaatab läbi N tööd ja kontrollib igaühe jaoks M väärtust, ehk teeb kokku suurusjärku $N \cdot M$ operatsiooni, mis jääb suuremates testides ajahätta.

Teine võimalus on B_i väärtuste jada ära sorteerida. Siis on võimalik selle ühe läbimisega tuvastada ja jooksvalt väljastada kõik sellest puuduvad tööde numbrid. Mingit efektiivset üldotstarbelist sorteerimisalgoritmi kasutades kulub M kirje sorteerimiseks $M \cdot \log M$ operatsiooni, mis oleks juba piisavalt efektiivne.

Kõige lihtsam on luua iga töö jaoks olekumuutuja (näiteks esinemiste loendur). N operatsiooniga on võimalik need loendurid algväärtustada, seejärel M operatsiooniga B_i väärtuste jada läbi käia ja iga B_i jaoks vastavat loendurit suurendada. Lõpuks saame veel N operatsiooniga loendurid uuesti üle vaadata ja väljastada need tööd, mille loendur on null. See ongi kõige efektiivsem võimalik lahendus.

Testid

1. $N = 5$, $M = 3$. Väike lihtne test.
2. $N = 10$, $M = 1$. Väike lihtne test.
3. $N = 10$, $M = 9$. Tööd moodustavad ahela.
4. $N = 10$, $M = 45$. Iga töö sõltub kõigist, mille numbrid on väiksemad.
5. $N = 10$, $M = 5$. Kolm tööd on omavahel tupikus, lisaks kaks paari niisama sõltuvusi.
6. $N = 50$, $M = 400$. Tööd 5 kihis, iga kihi kõik tööd sõltuvad eelmise kihi kõigist töödest, kihid on eestpoolt tahapoole lugedes $21 \ldots 30$, $1 \ldots 10$, $41 \ldots 50$, $31 \ldots 40$, $11 \ldots 20$.
7. $N = 100$, $M = 4950$. Iga töö sõltub kõigist, mille numbrid on suuremad.
8. $N = 1000$, $M = 990$. Üks pikk ahel, lisaks mõned üksikud tööd.
9. $N = 10\,000$, $M = 10\,000$. Suur juhuslik test.
10. $N = 50\,000$, $M = 50\,000$. Maksimaalne juhuslik test.

10 testi, à 4 punkti, kokku 40 punkti.

3. Telefoniraamat

1 sekund 40/20 punkti

Selle ülesande lahendamiseks polegi vaja muud teha kui tekstis kirjeldatud algoritm korralikult ära programmeerida.

Kõige lihtsam on tuua sisse kaks muutujat, veel uurimise all oleva raamatuosa esitamiseks. Algseisus on nende väärtuseks muidugi $vasak = 1$ ja $parem = N$. Siis avaldub järgmisena vaadeldava lehekülje number alati kujul $uus = (vasak + parem)/2$, kus paaritu summa korral ümardame jagamise tulemuse allapoole.

Kui uus ongi otsitav lehekülg, on algoritmi töö sellega lõppenud. Kui vajalik lehekülg on raamatus tagapool kui uus, ahendame ülesande teksti põhjal otsimispiirkonda omistamisega $vasak := uus + 1$. Kui vajalik lehekülg on eespool, siis omistame $parem := uus - 1$.

Testid

1. Üheleheküljeline "raamat".
2. Kõik poolitused alguse suunas, alati leidub keskmine lehekülg.
3. Kõik poolitused alguse suunas, kunagi ei leidu keskmist lehekülge.
4. Kõik poolitused lõpu suunas, alati leidub keskmine lehekülg.
5. Kõik poolitused lõpu suunas, kunagi ei leidu keskmist lehekülge.
6. Otsitav lehekülg on keskmine.
7. Paarisarv lehekülgi, otsitav keskkohale eelnev.
8. Maksimaalne arv lehekülgi.
9. Otsitav lehekülg teises pooles.
10. Otsitav lehekülg esimeses pooles.

Põhikooli rühm: 10 testi, à 4 punkti, kokku 40 punkti.

Algajate rühm: 10 testi, à 2 punkti, kokku 20 punkti.

4. Taim

1 sekund 40 punkti

Muidugi on üks võimalus selle ülesande lahendamiseks taime kasvamist harude kaupa simuleerida ja vahetult loendada need, mis täpselt küsitaval nädalal õide puhkevad. Selline lahendus jääb suuremates testides ajahätta, sest puu harude arv kasvab aja edenedes väga kiiresti.

Kui tähistame i-ndal nädalal 3-, 2- ja 1- nädalaseks saavate peaharude arvud $P3_i$, $P2_i$, $P1_i$ ja sel nädalal kasvavate kõrvalharude arvu K_i, siis kehtivad järgmised seosed:

- Otsitav uute õite arv i-ndal nädalal on võrdne 3-nädalaseks saavate peaharude arvuga $P3_i$.
- 3-nädalasi peaharusid saab ainult 2-nädalaste pikemaks kasvatamisega, ehk $P3_i = P2_{i-1}$.
- 2-nädalasi peaharusid saab ainult 1-nädalaste pikemaks kasvatamisega, ehk $P2_i = P1_{i-1}$.
- 1-nädalasi peaharusid saab ainult kõrvalharu hargnemisega, ehk $P1_i = K_{i-1}$.
- Kõrvalharusid saab nii olemasolevate kõrvalharude kui ka 1- ja 2-nädalaste peaharude hargnemisega, ehk $K_i = K_{i-1} + P1_{i-1} + P2_{i-1}$.

Võttes lisaks arvesse ka lähtepunkti $P3_0 = 0$, $P2_0 = 0$, $P1_0 = 0$, $K_0 = 0$, on lihtne arvutada $P3_1$, $P2_1$, $P1_1$, K_1, neist omakorda $P3_2$, $P2_2$, $P1_2$, K_2 ja samamoodi jätkates saada $P3_N$, $P2_N$, $P1_N$, K_N väärtused $4 \cdot N$ omistamisega.

Korrektse lahenduse saamiseks on lisaks vaja panna tähele, et töödeldavad väärtused kasvavad üsna kiiresti ja kasutada tuleks vähemalt 64-bitiseid täisarvmuutujaid.

Testid

1. $N = 1$. Arvutuse alguse korrektsuse kontrolliks.
2. $N = 2$. Arvutuse alguse korrektsuse kontrolliks.
3. $N = 3$. Arvutuse alguse korrektsuse kontrolliks.
4. $N = 4$. Esimene rekurrentselt arvutatav väärtus.
5. $N = 7$. Esimene väärtus, mida ülesande tekstis antud pole.
6. $N = 9$. Seda on joonise pealt juba raske õigesti kokku lugeda.
7. $N = 12$. Seda on joonise pealt juba üsna võimatu õigesti kokku lugeda.
8. $N = 17$. Juhuslik vahepealne väärtus.
9. $N = 22$. Ületäitumine 16-bitises märgiga täisarvus.
10. $N = 23$. Ületäitumine 16-bitises märgita täisarvus.
11. $N = 26$. Juhuslik vahepealne väärtus.
12. $N = 30$. Naiivne variantide läbivaatus jääb ajahätta.
13. $N = 35$. Efektiivne variantide läbivaatus jääb ajahätta.
14. $N = 40$. Ületäitumine 32-bitises märgiga täisarvus.
15. $N = 41$. Ületäitumine 32-bitises märgita täisarvus.
16. $N = 42$. Täpsuskadu 32-bitise IEEE ujukomaarvuga.
17. $N = 65$. Täpsuskadu Exceli või OOo Calc'iga.
18. $N = 66$. Täpsuskadu 64-bitise IEEE ujukomaarvuga.
19. $N = 69$. Täpsuskadu 80-bitise IEEE ujukomaarvuga.
20. $N = 75$. Maksimaalne test.

20 testi, à 2 punkti, kokku 40 punkti.

5. Koondamine

1 sekund 40/20 punkti

Olgu sõne pikkus S tähte ning tähistame tähe rasteresituse kõrgust ja laiust vastavalt K ja L (antud ülesandes seega $K = 7$ ja $L = 7$).

Üks võimalik lahendus on mälus hoida maatriksina sõne esimese otsa kujutist ja hakata sellele paremalt tähti juurde lisama. Paigutades tähe maatriksi paremasse serva ja hakates teda sealt pikselhaaval vasakule nihutama, teeb lihtsameelne lahendus igal nihutamisel $O(K \cdot L)$ sammu. Et igat tähte on vaja nihutada $O(S \cdot L)$ piksli võrra, kulub aega $O(K \cdot L^2 \cdot S^2)$, mis peaks veel 1 sekundi ajapiiri sisse mahtuma. Kui lisatav täht paigutada mitte maatriksi paremasse serva, vaid viimase tähe kõrvale, saab lahenduse ajakuluga $O(K \cdot L^2 \cdot S)$, sest uut tähte tuleb siis nihutada vaid $O(L)$ piksli võrra. Nii on tehtud näidislahenduses `kernlah1.java`.

Pannes tähele, et koondamisel on olulised vaid tähe "vasakpoolne" ja "parempoolne" kuju, s.o. ridade vasakpoolseimad ja parempoolseimad pikslid, saab tõhusama lahenduse: kui tähestiku iga tähe kohta koostada eelnevalt üks järjend iga rea vasakpoolseima piksli veerunumbriga ja teine iga rea parempoolseima piksli veerunumbriga, võib sõne koondatud esituse laiuse leida ajaga $O(K \cdot S)$. Näidislahenduses `kernlah2.java` on seda mõttekäiku veidi edasi arendatud ja tähe vasakpoolse kuju asemel vaadeldud tähe "paksendi" vasakpoolset kuju, kus tähe paksendiks nimetame kõigi pikslite hulka, mis külge- või nurkapidi puutuvad vastu tähe musta pikslit. Nii muutub sõne koondamine veel veidi lihtsamaks (ajakulu küll endiselt $O(K \cdot S)$).

Veel üks võimalus on arvutada kõigepealt välja iga tähestikus esineva tähepaari koondatud laius. Sõne koondatud laiuse saab seejärel leida ajaga $O(S)$. Selleks on vaja eeldust, et koondamisel piisab arvestada tähe puutumist vaid talle vahetult eelneva ja järgneva tähega, mis on aga antud fondi puhul täidetud, sest igal tähel peale 'I' on igas reas vähemalt üks must piksel ning 'I'-tähele saab tühja ritta musta piksli juurde panna ilma, et tema koondamisomadused muutuksid.

Eelnevalt väljaarvutatud koondamistabelit kasutatakse ka päris fontides: osalt arvutamise ajamahukuse tõttu, osalt seetõttu, et tihti kohendab fondi koostaja koondamist käsitsi, kuna antud ülesandes kasutatu taoline lihtne algoritm alati parimat tulemust ei anna. Käesolevas ülesandes oli põhimõtteliselt võimalik kõigi $26^2 = 676$ tähepaari koondamised käsitsi arvutada (võib-olla tekstitoimetiga fondis nullid ja ühed paremini eristatavate märkidega asendades), aga arvatavasti oli kiirem ja veakindlam siiski programm kirjutada.

Testid

1. `I` — ühe piksli laiune, sisaldab tühja rida.
2. `C` — kõige laiem ja kõige kitsam koht pole ühes reas.

3.–7. Igas sisendis kaks tähte, katsetavad erinevaid võimalikke puutumisi.

8. `FJL` — näitab, et järgneva tähe (`J`) 7×7 esitus võib alata varem kui eelmise tähe (`F`) oma lõpeb.
9. `OVOO` — katsetab veel võimalikke puutumisi.

10.–17. Kõik võimalikud 676 tähepaari järjest; kaks sõnet suurima lubatud tähtede arvuga.

18. Sisaldab äärtes väga kitsaid tähti (`I`) ja nende vahel suvalisi tähti. Kuna `I`-l on vasakul ja paremal valgeid veerge, siis sõne, mis algab ja lõpeb `I`-ga, võimaldab veenduda, et lahendus mõõdab sõne laiust vasakpoolseimast mustast pikslist parempoolseima musta pikslini.
19. Mõõduka pikkusega suvaliste tähtedega sõne.
20. Suurima võimaliku laiusega sõne.

Algajate rühm: 20 testi, à 2 punkti, kokku 40 punkti.

Edasijõudnute rühm: 20 testi, à 1 punkt, kokku 20 punkti.

Kasutatavas fondis esinevad kõik võimalikud puutumised:

- ülemises reas otse: `TT`, `FF`;
- vasakult alt ülemises reas: `PT`;
- paremalt alt ülemises reas: `TS`;
- alumises reas otse: `LL`;
- vasakult ülalt alumises reas: `BZ`;
- paremalt ülalt alumises reas: `LS`;
- mõnes keskmises reas otse: `OO`;
- vasakult ülalt mõnes keskmises reas: `OJ`, `VO`;
- vasakult alt mõnes keskmises reas: `OV`.

Ühetäheline sõne on erijuht, millega peab arvestama; samas võimaldab see mõnel poolikul lahendusel lisapunkte saada.

6. Kell

1 sekund 40 punkti

Lahendused `naitlah1.c` ja `naitlah1.py` töötavad mõlemad samal moel:

- Programmi teksti on tabelina kirjutatud korras tablool põlevate segmentide kirjeldus iga numbri jaoks.
- Uuritakse tunde ja minuteid eraldi. Tundide puhul kontrollitakse iga väärtuse 00...23 jaoks, kas selle kümneliste ja üheliste numbrid on sisendis antud tablooga kooskõlas. Minutite puhul vaadatakse läbi 00...59.

Iga numbri iga võimaliku väärtuse juures uuritakse, kas sisendfailist loetud nähtavate segmentide kirjeldus sobib selle väärtuse ettenähtud segmentidega. Selleks vaadeldakse kõiki segmente ühekaupa:

- Kui tablool segment põleb ja numbris peakski põlema, siis on tabloo seal kohas korras ja number on võimalik.
- Kui tablool segment ei põle ja numbris ei peakski põlema, siis ei ole teada, kas tabloo on korras, aga number on võimalik.
- Kui tablool segment ei põle, aga numbris peaks põlema, siis võib tabloo see segment läbi põlenud olla, aga number on katkisel tablool võimalik.
- Kui tablool segment põleb, aga numbris ei peaks põlema, siis ei ole see number isegi läbipõlenud segmentidega tablool võimalik.

Huvitav on ainult viimane variant, sest see tähendab, et vaadeldav number ei saa olla sisendis toodud segmentidega tablool. Kõigil muul juhtudel on number endiselt võimalik.

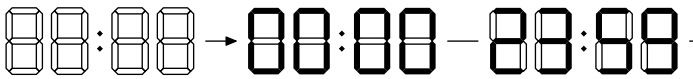
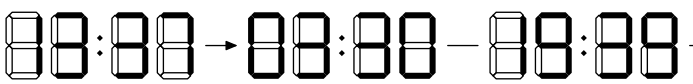
Kui leidub tablooga sobivaid tundide ja minutite väärtusi, siis väljastatakse nende seast minimaalsed ja maksimaalsed. Kõige varasem võimalik kellaaeg on minimaalse tundide ja minutite arvuga, hiliseim maksimaalsetega.

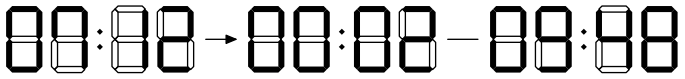
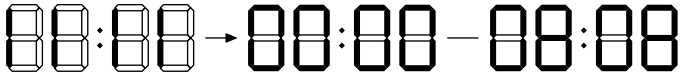
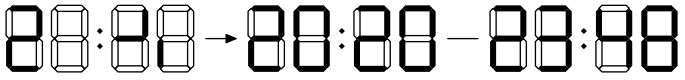
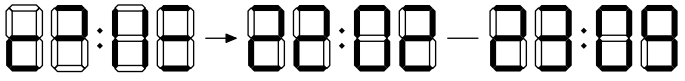
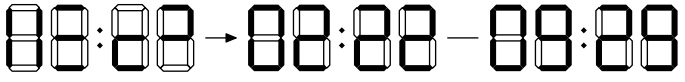
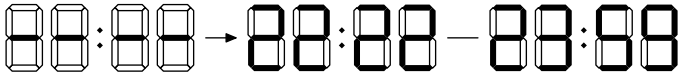
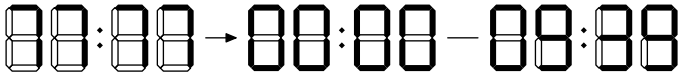
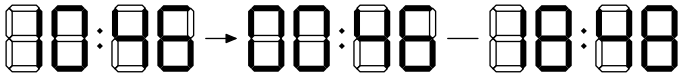
Kui tablool nähtavate segmentidega kokku sobivate tundide või minutite hulk on tühi, siis on kell väga katki, ehk viga on tõsisem kui lihtsalt läbipõlenud segmendid ja väljastatakse `VIGA`.

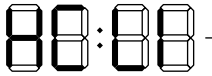
Pythonis kirjutatud näidislahendus hoiab võimalikke tundide ja minutite väärtusi mälus `set()` andmetüübina. C näidislahendus ei hoia kogu hulka mälus, vaid proovib kõik variandid läbi ja tagastab ainult minimaalse ja maksimaalse elemendi, mis on natuke efektiivsem.

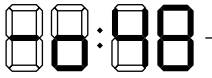
Lahenduse algoritmiline keerukus on $O((\text{tundide arv} + \text{minutite arv}) \cdot 2 \cdot \text{võimalikke numbreid} \cdot \text{segmente numbris})$. Kuna see on ainult $(24+60) \cdot 2 \cdot 10 \cdot 7 = 11760$, siis töötavad kõik lahendused välkkiirelt.

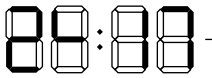
Testid

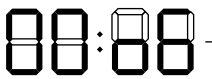
1.  00:00-23:59
   Ükski segment ei põle, seega kõik võivad olla läbi põlenud ja kõik numbrid on võimalikud.
2.  08:08-19:49
3.  02:00-23:59
4.  03:30-19:39

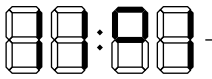
5.  00:02-09:48
6.  00:00-08:08
7.  20:20-23:48
8.  22:02-23:09
9.  02:22-09:29
10.  08:40-09:48
11.  03:04-23:49
12.  22:22-23:59
13.  00:00-09:39
14.  00:46-18:48
15.  VIGA

    Kõik üksikud numbrid on võimalikud, kuid 88:88 ei ole 24-tunni süsteemi kellaaeg.
16.  VIGA

    Tundide kümneliste kohal saab olla vaid 8.
17.  VIGA

    Tundide kümneliste kohal saab olla vaid 2 ja üheliste kohal vaid 6 või 8.
18.  VIGA

    Tundide üheliste kohal saab olla vaid 4, 5, 6, 8 või 9.
19.  VIGA

    Minutite kümneliste kohal saab olla vaid 6 või 8.
20.  VIGA

    Minutite kümneliste kohal saab olla vaid 8 või 9.

20 testi, à 2 punkti, kokku 40 punkti.

7. Vead

1 sekund 40 punkti

See ülesanne võib esmapilgul tunduda väga keeruline, sest tegu on graafiga, kus on kahe erineva tähendusega servi (sõltuvused ja duplikaadiviited), millest ühed on lisaks suunatud ja teised suunamata.

Siiski on lahenduseks vaja ainult üsna põhilisi algoritme, kui läheneda ülesande lahendamisele etapiviisilisekt:

1. Vaadelda graafi, mille tipud on veakirjed ja suunamata servad duplikaadiviited. Siis on unikaalsed vead täpselt selle graafi sidususkomponendid (ingl *connected component*), mis on lihtsalt tuvastatavad sügavuti läbimisega, kulutades $O(N + K)$ operatsiooni.
2. Valida igast sidususkomponendist üks (näiteks minimaalse numbriga) kirje selle "esindajaks" ja asendada sõltuvuste infos kõigi teiste sellesse komponenti kuuluvate kirjete numbrid esindaja numbriga. Kui märkida iga veakirje juurde tema esindaja number (mida on lihtne teha kohe sidususkomponentide tuvastamise käigus), siis saab selle asenduse teha $O(M)$ operatsiooniga.
3. Vaadelda graafi, mille tipud on sidususkomponentide esindajad ja suunatud servad sõltuvuseviited. Siis on vaja selle graafi topoloogilist järjestust (ingl *topological ordering*), mille saab leida sügavuti läbimisega, kulutades $O(V + M)$ operatsiooni, kus $V \leqslant N$ on esimeses punktis leitud sidususkomponentide ehk unikaalsete vigade arv.

Testid

1. $N = 5$, $K = 1$, $M = 1$.
2. $N = 10$, $K = 8$, $M = 1$.
3. $N = 10$, $K = 1$, $M = 9$.
4. $N = 16$, $K = 6$, $M = 12$. Erijuht: duplikaadiviited tsüklis.
5. $N = 50$, $K = 45$, $M = 9$.
6. $N = 100$, $K = 50$, $M = 49$.
7. $N = 1000$, $K = 998$, $M = 1$.
8. $N = 1005$, $K = 900$, $M = 500$. Erijuht: duplikaadiviited on antud "tagurpidi"; $1 \to 10$, $2 \to 10$, $\ldots$, $9 \to 10$ asemel $10 \to 1$, $10 \to 2$, $\ldots$, $10 \to 9$.
9. $N = 30\,000$, $K = 1000$, $M = 1$.
10. $N = 50\,000$, $K = 1000$, $M = 25\,000$.

10 testi, à 4 punkti, kokku 40 punkti.