1. Raadiolevi

1 sekund 20 punkti

Selle ülesande lahendamiseks pole eriti muud võimalust kui arvutada järjest iga saatja signaali tugevus vastuvõtja asukohas ja valida välja see saatja, mille signaal on tugevaim; kasutades ülesande tekstis toodud tähistusi, võiks algoritm olla umbes selline:

$parimS := 0$
for $i := 1$ to N:
    $s := P_i/((X - X_i)^2 + (Y - Y_i)^2)$
    if $s > parimS$ then:
        $parimS := s$
        $parimI := i$
return $parimI$

Failis `levilah1.java` toodud lahendus just niimoodi töötabki.

Veel väärivad mainimist paar realisatsiooni detaili, mille unustamine võib lahenduse rikkuda:

- Kuigi nii koordinaadid kui nende vahed jäävad 16-bitise märgiga täisarvu määramispiirkonda, on kauguste ruudud juba piisavalt suured, et nendega opereerimiseks peab kasutama vähemalt 32-bitiseid muutujaid. 16-bitised on näiteks Pascali `integer` (selle asemel tuleks kasutada tüüpi `longint`) ning C, C++ ja Java `short` (selle asemel tuleks kasutada tüüpi `int`).
- Nii saatjate võimsused kui kauguste ruudud on täisarvud. Paljudes programmikeeltes (C, C++, Java, Python 2.x) on kahe täisarvu jagatis alati täisarv (kui võimsus kauguse ruuduga ei jagu, visatakse signaali tugevuse murdosa lihtsalt minema). Nendes keeltes on õige vastuse saamiseks vaja vähemalt üks kahest operandist enne jagamist reaalarvuks teisendada. Täisarvulise jagamise tulemuse reaalarvu tüüpi muutujale omistamine ei aita, sest jagamine on selleks ajaks juba täisarvulisena ära tehtud ja saadud tulemuse reaalarvutüüpi teisendamine kaotsiläinud murdosa enam tagasi ei too.

Testid

1. $N = 2$. Väike lihtne test, maksimaalse võimsusega saatja ongi otsitav.
2. $N = 4$. Koordinaadid on väikesed arvud, täisarvudega arvutades ümardamisvea ohtu pole (ka 16-bitiste täisarvudega arvutades saab õige vastuse).
3. $N = 5$. Koordinaadid on väikesed arvud, täisarvudena jagada ei tohi (16-bitiseid täisarve ruutu tõstes ja siis reaalarvudena jagades saab õige vastuse).
4. $N = 4$. Koordinaadid on suured arvud, täisarvudena jagada ei tohi (16-bitiste täisarvude ruutu tõstmisel tekib ületäitumine; 32-bitiseid täisarve ruutu tõstes ja siis reaalarvudena jagades saab õige vastuse).
5. $N = 20$. Juhuslikult genereeritud test.
6. $N = 50$. Juhuslikult genereeritud test.
7. $N = 100$. Juhuslikult genereeritud test.
8. $N = 250$. Juhuslikult genereeritud test.
9. $N = 500$. Juhuslikult genereeritud test.
10. $N = 1000$. Juhuslikult genereeritud test.

Testid 1...10 à 2 punkti, kokku 20 punkti.

Kõigis testides on õige vastus ühene.

2. Filatelist (põhikooli variant)

1 sekund 40 punkti

Selle ülesande lahendamisel on kaks põhilist raskust:

1. Kuidas korraldada markide võrdlemine, kui sisendis on ühe margi kirjeldus laiali mitmel real ja faili järjest lugedes vaheliti teiste markide kirjeldustega?
2. Kuidas korraldada markide paaride läbivaatus?

Markide võrdlemiseks on kaks põhilist võimalust:

- Failis `filplah1.pas` toodud lahendus jagab iga sisendist loetud rea juppideks ja kleebib iga margi kirjelduse jupid kokku üheks pikaks sõneks.
- Failis `filplah2.pas` toodud lahendus loeb andmed sisse nii nagu nad failis antud on ja leiab kahe margi võrdlemisel jooksvalt kummalegi margile vastavad elemendid.

Ka paaride läbivaatuse korraldamiseks on kaks põhilist võimalust:

- Failis `filplah1.pas` toodud lahendus teeb duplikaatide paari esimese margi leidmiseks kaks eraldi kordust, ühe üle markide ridade, teise üle veergude, ja sama moodi ka paari teise margi leidmiseks. Sellise lähenemise raskus on, et paari teine mark võib olla kas esimesega samal real temast paremal või esimesest allpool ja sel juhul nii esimesest vasakul kui paremal kui ka täpselt esimese all. Kuna selles lahenduses me lõpetame töö kohe esimese ühesuguste paari leidmisel, siis saaks korduse teha ka natuke lihtsamalt (kontrollides ainult, et me ei võrdle marki iseendaga), aga mõnes teises ülesandes võib näidislahenduses toodud veidi keerulisem tingimus olla tõesti vajalik selleks, et iga paari töödeldaks täpselt üks kord.
- Failis `filplah2.pas` toodud lahendus vaatleb marke ühe pika rivina, nummerdades nad mõttes ridade kaupa vasakult paremale ja ülevalt alla $1 \ldots R \cdot V$. Sellises mõttelises jadas on muidugi lihtne kirjutada kordusi, mis vaatavad iga paari täpselt ühe korra, aga sellevõrra on keerulisem leida margi järjekorranumbrist tema rea- ja veeruindekseid (viimaste avaldised on veidi lihtsamad, kui kasutada massiive, mille indeksid algavad nullist, mitte ühest).

Muidugi oleks samavõrra õiged ka lahendused, mis teeks markide võrdlemise nii nagu on tehtud failis `filplah1.pas` ja paaride läbivaatuse nii nagu failis `filplah2.pas` või vastupidi.

Testid

1. 1×1 marki. Ainus mark on muidugi unikaalne, vastus `POLE`.
2. 1×10 marki. Kõik margid ühes reas.
3. 50×1 marki. Kõik margid ühes veerus.
4. 5×5 marki. Margid hästi sarnased, erinevused ainult alumises paremas nurgas.
5. 8×8 marki. Juhuslik negatiivne test, vastus `POLE`.
6. 20×7 marki. Duplikaatidest ülemine on alumisest paremal pool.
7. 30×9 marki. Duplikaadid samas veerus.
8. 50×10 marki. Maksimaalne test, duplikaadid samas reas (viimase rea kaks viimast).

Testid 1…8 à 5 punkti, kokku 40 punkti.

3. Maraton (põhikooli variant)

1 sekund 40 punkti

Kõige lihtsam viis seda ülesannet lahendada on vaadata läbi kõik võimalikud rajalõigud, lugeda igas lõigus kokku vabade kohtade arv ja väljastada piisava hulga vabade kohtadega lõikude hulgast parim. Selline lahendus on toodud failis `marplah1a.pas` ja leiab küll õiged vastused, kuid kulutab selleks sageli liiga palju aega. N kohaga rajal on võimalikke lõike umbes $N^2/2$. Kuna ühe lõigu keskmine pikkus on $N/3$, kulub kõigis lõikudes vabade kohtade loendamiseks kokku umbes $N^3/6$ operatsiooni. Seega suudab selline lahendus ajalimiiti ületamata töödelda juhte, kus N väärtus on mõnesaja piires.

Üks lihtne võimalus eelmise lahenduse töökiiruse parandamiseks on leida kohe alguses iga koha jaoks vabade kohtade arv raja algusest kuni selle kohani. Siis saame mistahes rajalõigul olevate vabade kohtade arvu nende loendamise asemel leida ühe tehtega: lahutades lõigu lõpukoha loenduri väärtusest lõigu alguskohale eelneva koha loenduri väärtuse. Selline lahendus on toodud failis `marplah1b.pas`, mis kulutab kokku umbes $N^2/2$ operatsiooni ja suudab töödelda juhte, kus N väärtus on mõne tuhande piires.

Teine võimalus esimese naiivse lahenduse parandamiseks on panna tähele, et parimas paigutuses istuvad sõbrad alati nii, et nende vahele ei jää ühtki vaba kohta. Tõepoolest, iga sellist lahendust, kus grupi sees on vaba koht, saaks parandada sellega, et üks kahest äärmisest grupiliikmest istub ümber sellele vabale kohale.

Järelikult võib vaadata läbi kõik vabad kohad ja proovida igas vabas kohas panna sõbrad istuma kõigile järjestikustele vabadele kohtadele sellest kohast alates. Selline lahendus on toodud failis `marplah2a.pas`. Kuna see lahendus vaatab läbi kõik N kohta ja võib juhtuda, et peaaegu igas kohas õnnestub paigutada kõik K sõpra, kulub tal kokku kuni $N \cdot K$ operatsiooni. See tähendab, et kui K on väike arv, suudab see lahendus töödelda ka juhte, kus N on maksimaalne, kuid suurte K väärtustega testides võib ta siiski ajahätta jääda.

Tegelikult pole meil vaja eelmisest paigutusest järgmise saamiseks kogu tööd uuesti teha, piisab vaid esimese sõbra koha vabastamisest ja talle viimase sõbra järel uue vaba koha leidmisest. Kui jooksva paigutuse algust ja lõppu meeles hoida, siis tuleb kõigi selliste nihutamiste peale kokku iga koha andmeid ainult kaks korda vaadata (esimene kord sõbrale koha otsimisel ja teine kord pärast sõbra koha vabastamist järgmise sõbra otsimisel). Selline lahendus on toodud failis `marplah2b.pas` ja on piisavalt efektiivne, et teenida maksimumpunktid.

Kuigi see võistluse tulemuse seisukohalt enam vajalik ei ole, saaks ka seda lahendust veel edasi arendada. Nimelt pole vabade kohtade arvestamiseks vaja eraldi muutujat igale kohale, piisab ainult nimekirjast, kus on järjestikustest vabadest kohtadest koosnevad lõigud. Selles nimekirjas navigeerimine on küll veidi keerulisem programmeerida, aga testides, kus neid lõike on oluliselt vähem kui N, võimaldab niisugune lähenemine hoida kokku nii mälu kui ka aega.

Testid

1. $N = 1$, $M = 0$, $K = 1$. Minimaalne test. Üksik "sõber" saab ainsa koha, mis raja ääres üldse on.
2. $N = 12$, $M = 4$, $K = 4$. Ainult üks koht, kuhu sõbrad vahetult üksteise kõrvale mahuvad, mujal jääks võõraid vahele.
3. $N = 13$, $M = 5$, $K = 4$. Kolm kohta, kuhu sõbrad mahuvad nii, et ainult üks võõras on vahel, valida tuleb neist finišile lähim.
4. $N = 15$, $M = 6$, $K = 4$. Piirjuht: parimad kohad vahetult finišis.
5. $N = 15$, $M = 6$, $K = 4$. Piirjuht: parimad kohad vahetult stardis.
6. $N = 15$, $M = 10$, $K = 5$. Piirjuht: sõbrad võtavad ära kõik vabad kohad.
7. $N = 30$, $M = 10$, $K = 5$. Väike juhuslik test, palju vabu kohti.
8. $N = 30$, $M = 20$, $K = 5$. Väike juhuslik test, vähe vabu kohti.
9. $N = 100$, $M = 50$, $K = 10$. Väike juhuslik test.
10. $N = 300$, $M = 100$, $K = 100$. Väike juhuslik test.
11. $N = 1000$, $M = 100$, $K = 100$. Mõõduka suurusega juhuslik test.
12. $N = 3000$, $M = 300$, $K = 300$. Mõõduka suurusega juhuslik test.
13. $N = 10\,000$, $M = 1000$, $K = 100$. Suur juhuslik test.
14. $N = 30\,000$, $M = 3000$, $K = 300$. Suur juhuslik test.
15. $N = 100\,000$, $M = 10\,000$, $K = 10\,000$. Väga suur juhuslik test.
16. $N = 1\,000\,000$, $M = 100\,000$, $K = 100\,000$. Maksimaalse suurusega juhuslik test.

Testid 1…10 à 1 punkt, 11…16 à 5 punkti, kokku 40 punkti.

4. Filatelist (gümnaasiumi variant)

1 sekund 20 punkti

Selle ülesande lahendamisel on kaks põhilist raskust:

1. Kuidas korraldada markide võrdlemine, kui sisendis on ühe margi kirjeldus laiali mitmel real ja faili järjest lugedes vaheliti teiste markide kirjeldustega?
2. Kuidas korraldada markide jagamine gruppideks?

Markide võrdlemiseks on kaks põhilist võimalust:

- Failis `filglah1.pas` toodud lahendus jagab iga sisendist loetud rea juppideks ja kleebib iga margi kirjelduse jupid kokku üheks pikaks sõneks.
- Failis `filglah2.pas` toodud lahendus loeb andmed sisse nii nagu nad failis antud on ja leiab kahe margi võrdlemisel jooksvalt kummalegi margile vastavad elemendid.

Ka gruppide leidmiseks on kaks põhilist võimalust:

- Failis `filglah1.pas` toodud lahendus sorteerib margid nende kirjelduste järgi. Sorteeritud jadas satuvad ühesugused margid kõrvuti, mistõttu on selle väljastamise käigus gruppideks jagamine juba lihtne.
- Failis `filglah2.pas` toodud lahendus koostab ilmutatud kujul gruppide nimekirja. Iga järgmist marki on vaja võrrelda igast senisest grupist ainult ühe esindajaga (grupi kõik teised liikmed on ju samasugused) ja kui ta ühtegi olemasolevasse gruppi ei sobi, saab ta ise uue grupi esindajaks.

Pascalis lahendajate jaoks on selles ülesandes veel üks tehniline komplikatsioon: sisendi read võivad olla pikemad kui 255 märki, mis on FreePascali vaikimisi kasutatava `String`-tüübi jaoks pikkuse ülempiir. Failis `filglah1.pas` asendab kompilaatori direktiiv `{$LONGSTRINGS ON}` tavalise sõnetüübi pikemaid väärtusi toetava tüübiga (mille nimi muidu oleks `AnsiString` ja millel on võrreldes standardse `String`-tüübiga teisi miinuseid). Failis `filglah2.pas` töödeldakse andmeid märkhavaal, mistõttu sõnetüüpide piirangud seda lahendust ei mõjuta.

Testid

1. 1×1 marki à 1×1 pikselit. Ainus mark on muidugi unikaalne.
2. 1×10 marki à 1×2 pikselit. Kõik margid ühes reas, nii unikaalseid kui duplikaate (üks paar ja üks kolmik).
3. 20×1 marki à 3×1 pikselit. Kõik margid ühes veerus, üks duplikaatide paar.
4. 5×5 marki à 8×8 pikselit. Margid hästi sarnased, erinevused ainult alumises paremas nurgas, üks duplikaatide paar.
5. 20×7 marki à 10×10 pikselit. Kõik margid unikaalsed.
6. 20×10 marki à 10×10 pikselit. Kõik margid ühesugused.
7. 10×8 marki à 15×15 pikselit. Suur juhuslik test, üksikud duplikaatide paarid.
8. 30×8 marki à 30×30 pikselit. Maksimaalse suurusega margid, üksikud duplikaatide paarid.
9. 50×10 marki à 10×25 pikselit. Maksimaalne arv marke, üks duplikaatide grupp.
10. 50×10 marki à 30×30 pikselit. Maksimaalse suurusega test, üksikud duplikaatide paarid. Sisendi read pikemad kui 255 märki.

Testid 1…10 à 2 punkti, kokku 20 punkti.

5. Jada

1 sekund 30 punkti

Muidugi võib selle ülesande eest 10 punkti teenida programmiga, mis proovib antud arve kõigis võimalikes järjekordades kokku kleepida ja väljastab saadud tulemustest maksimaalse. Failis `jadalah1.pas` toodud lahendus nii teebki. Kuna N arvu võimalikke järjestusi on $N!$, suudab see lahendus ajalimiiti ületamata töödelda sisendeid N väärtustega umbes kümneni.

Paljude selliste sisendite korral väljuks tulemuste väärtused nii 32- kui ka 64-bitiste täisarvutüüpide määramispiirkonnast. Selle vastu tasub tähele panna, et antud arvude mistahes järjekorras üksteise otsa kleepimisel on tulemus alati sama pikkusega. Järelikult on võimalike tulemuste hulgas suurema arvulise väärtusega need, mille esimene number on suurem, nende hulgas omakorda on suuremad need, mille teine number on suurem, nende hulgas omakorda need, mille kolmas number on suurem jne. Seega pole meil tulemuste võrdlemiseks üldse vaja neid arvudeks teisendada ja nii vabanemegi standardsete täisarvutüüpide seatud piirangutest. Aga ajalimiidi ületamise vastu see siiski ei aita.

Parema lahenduse leidmiseks arendame eelmist mõttekäiku edasi: maksimaalse tulemuse saamiseks peame esimeseks elemendiks valida see, mille algusnumbrid on suurimad; olles esimese elemendi ära valinud, tuleb järgmiseks ülejäänute hulgast jälle valida see, mille algusnumbrid on suurimad. Seega tundub, et lahenduseks on elementide väljastame nende leksikograafilise kahanemise järjekorras.

Aga osutub, et eelnev väide kehtib vaid siis, kui ükski element ei ole ühegi teise prefiks — kui pole sellist elementide paari, kus teise elemendi lõpust mingi hulga numbrite kustutamisega on võimalik saada esimene element. Tõesti, leksikograafilises järjekorras pannakse prefiks oma ülemsõnedest (pikematest sama algusega sõnedest) ettepoole, aga näiteks hulga $\{2, 23\}$ järjestaks me selle reegli järgi järjekorda $(2, 23)$, millest saame tulemuse 223, kuigi 232 on ilmselt suurem arv. Ka prefiksireegli vastupidiseks muutmine ei paranda olukorda, sest annab vale tulemuse näiteks hulga $\{2, 21\}$ korral, millest siis saaksime järjekorra $(21, 2)$ ja sellest arvu 212, mis on ilmselt väiksem kui 221.

Juba nende kahe näite pealt peaks olema üsna lihtne üldistada, et prefiks ja tema ülemsõne tulebki alati omavahel järjestada selle järgi, kummas järjekorras neid üksteise otsa kleepides on tulemus suurem. Samas peaks olema üsna ilmne, et kõigil muudel juhtudel annab see reegel tavalise leksikograafilise kahanemise järjekorra, seega võib seda kasutada kõigi elementide võrdlemiseks nende järjestamise käigus. Nii ongi tehtud failis `jadalah2.pas` toodud lahenduses.

Lihtsuse huvides kasutab see lahendus ruutkeerukusega sorteerimisalgoritmi. Selle ülesande üsna väikeste andmemahtude korral on see piisav, aga vajadusel saaks sama võrdlemisreeglit rakendada ka mõnes paremas sorteerimisalgoritmis ja saada sellega ka vastavalt efektiivsema lahenduse.

Testid

1. $N = 1$. Minimaalne test. Vastus alla 2^{31}.
2. $N = 3$. Kõik arvud võrdse pikkusega. Vastus alla 2^{31}.
3. $N = 5$. Erineva pikkusega arvud. Ükski arv pole ühegi teise prefiks, aga arvuliste väärtuste kahanemise järjekorras väljastamine annab vale vastuse. Vastus alla 2^{31}.
4. $N = 5$. Erineva pikkusega arvud. Prefiksi alati ettepoole järjestamine annab vale vastuse. Vastus alla 2^{31}.
5. $N = 5$. Erineva pikkusega arvud. Prefiksi alati tahapoole järjestamine annab vale vastuse. Vastus alla 2^{31}.
6. $N = 8$. Erineva pikkusega arvud. Korduvad arvud. Nullid. Prefiksi alati ettepoole või alati tahapoole järjestamine annab vale vastuse. Vastus alla 2^{63}.
7. $N = 30$. Juhuslik test.
8. $N = 100$. Juhuslik test.
9. $N = 300$. Juhuslik test.
10. $N = 1000$. Maksimaalne juhuslik test.

Testid 1...5 à 2 punkti, 6...10 à 4 punkti, kokku 30 punkti.

6. Jagajad

1 sekund 50 punkti

Lihtne lahendus sellele ülesandele oleks vaadata läbi kõik täisarvud alates ühest, loendada igaühe jagajate arv ja väljastada vastusena esimene sobiv. Nii ongi tehtud failis `jagalah1a.cpp` toodud lahenduses, mis kulutab arvu r jagajate loendamiseks r jagamistehet ja jõuab sekundiga töödelda juhud, kus vastus jääb umbes kümne tuhande piiresse, ning failis `jagalah1b.cpp` toodud lahenduses, mis lähtub tähelepanekust, et kui arv t on arvu r jagaja, siis on seda ka arv r/t, kulutab arvu r jagajate loendamiseks umbes $\sqrt{r}$ jagamist ja jõuab seega sekundiga töödelda juhud, kus vastus jääb umbes saja tuhande piiresse.

Parema lahenduse leidmiseks tuleb lähtuda aritmeetika põhiteoreemi nime all tuntud faktist, et mistahes positiivne täisarv r on täpselt ühel viisil esitatav korrutisena kujul

$$r = p_1^{a_1} \cdot p_2^{a_2} \cdot \ldots \cdot p_m^{a_m},$$

kus p_i on kasvavalt järjestatud algarvud ja a_i on positiivsed täisarvud. Jaguvuse põhiomadustest on üsna lihtne tuletada, et arvu r mistahes jagaja t peab sel juhul avalduma kujul

$$t = p_1^{b_1} \cdot p_2^{b_2} \cdot \ldots \cdot p_m^{b_m},$$

kus b_i on mittenegatiivsed täisarvud ja iga i korral $b_i \leqslant a_i$. See aga omakorda tähendab, et arvu r erinevate jagajate arv kokku on $(a_1 + 1) \cdot (a_2 + 1) \cdot \ldots \cdot (a_m + 1)$.

Eelnevat tähelepanekut teistpidi pöörates saame, et kui jagajate arv K on esitatav korrutisena $K = c_1 \cdot c_2 \cdot \ldots \cdot c_m$, kus $c_1 \geqslant c_2 \geqslant \ldots \geqslant c_m > 1$, siis vähim arvu K sellele tegurdusele vastav täpselt K jagajaga arv on

$$x = p_1^{c_1-1} \cdot p_2^{c_2-1} \cdot \ldots \cdot p_m^{c_m-1},$$

kus p_i on kasvavavalt järjestatud m esimest algarvu $(2, 3, 5, 7, \ldots)$. Tõepoolest, kui oletada, et eeltoodud korrutises on p_i hulgas mõni suurem algarv, siis saaksime korrutist vähendada, asendades selle suurema algarvu väiksema veel kasutamata algarvuga. Seega peavad p_i tõesti olema m esimest algarvu. Kui nüüd oletada, et nende algarvude kasvavalt järjestamisel ei ole astmenäitajad järjestatud mittekasvavalt, s.t. kui mingite i ja j korral $p_i < p_j$ ja $c_i < c_j$, siis saaksime c_i ja c_j väärtuste vahetamisel tulemuse

$$x' = x \cdot \frac{p_i^{c_j-1} \cdot p_j^{c_i-1}}{p_i^{c_i-1} \cdot p_j^{c_j-1}} = x \cdot (p_i/p_j)^{c_j-c_i},$$

mis eeltoodud tingimustel oleks väiksem kui x.

Järelikult piisab vähima täpselt K jagajaga arvu leidmiseks, kui vaatame läbi kõik K esitused mittekasvavate tegurite korrutisena ja valime kõigile tegurdustele vastavate x väärtuste hulgast minimaalse. Failis `jagalah2.cpp` toodud lahendus täpselt seda teebki.

Selle lahenduse tööaja hindamiseks märgime, et kui tulemus on väiksem kui 2^{63}, siis ei saa algteguri 2 aste selles olla suurem kui 62 (sest 2^{63} pole enam väiksem kui 2^{63}), 3 aste suurem kui 24 (sest 3 aste ei ole suurem kui 2 aste ja $2^{25} \cdot 3^{25}$ on juba suurem kui 2^{63}). Analoogiliselt arutledes saamegi, et variantide koguarv ei saa olla suurem kui 34 283 520 (ja tegelikult peab olema sellest tublisti väiksem, sest kui 2 aste on tõesti 62, siis kõigi teiste tegurite astmed peavad olema nullid, muidu oleks tulemus ikkagi suurem kui 2^{63}; analoogiliselt ka kõigi ülejäänud piirväärtuste puhul).

Testid

1. $K = 1$. Erijuht, täpselt ühe teguriga arve on täpselt üks: 1.

2.–10. Väikesed testid, vastused alla 10 000, saab proovimise ja jagajate loendamisega lahendada.

11.–15. Keskmised testid, vastused alla 1 000 000, saab veerand tunni jooksul jagajate loendamisega oma arvutis ära lahendada ja vastused tabelina programmi teksti sisse panna; piisab 32-bitistest muutujatest.

16.–25. Suured testid, lahendamiseks on vajalik tegurite analüüs ja 64-bitiste arvude kasutamine.

Testid 1. . . 10 à 1,5 punkti, 11. . . 15 à 2 punkti, 16. . . 25 à 2,5 punkti, kokku 50 punkti.

7. Kataloogipuu

1 sekund 20 punkti

Kataloogipuud hakkame koostama ridahaaval, vajadusel juba puusse pandud ridu uuendades.

Kõige keerulisem ongi kataloogi lisamine puusse, milleks tuleb teha järgmised toimingud:

1. Loeme kokku rea alguses olevad tühikud.
2. Leiame uue kataloogi sügavuse puus. Selleks otsime kõige viimase rea, mille taane ei ole suurem kui lisataval kataloogil.
   (a) Kui leitud rea taane on väiksem, siis uue kataloogi sügavus on leitud reale vastava kataloogi sügavusest ühe võrra suurem.
   (b) Vastasel juhul on uue kataloogi sügavus leitud kataloogi sügavusega võrdne.
   (c) Kui ühtegi sellist rida ei leita, siis kataloogi sügavus on 0 (ehk see ei kuulu ühegi teise kataloogi alla).
3. Nüüd tuleb olemasolevat kataloogipuud täiendada. Alustame viimasest kataloogist ja liigume ettepoole.
   (a) Kui vaadeldav kataloog on puusse lisatavast kataloogist väiksema sügavusega (taandega), siis võime puu täiendamise lõpetada.
   (b) Kui vaadeldav kataloog on puusse lisatava kataloogiga samal sügavusel (sama taandega), siis asendame vaadeldava kataloogi ees oleva sümboli `*` sümboliga `+` ja lõpetame puu täiendamise.
   (c) Kui vaadeldav kataloog on väiksema sügavusega ja lisatava kataloogi sügavus ei ole 0 (st, kui tegemist on alamkataloogiga), siis lisame kataloogi ette vertikaalse joone.
4. Lõpuks lisame uue kataloogi puusse. Kui lisatav kataloog on alamkataloog, siis selle nime ette lisame sümbolid `*-`.

Pascalis lahendajate jaoks on selles ülesandes veel üks tehniline komplikatsioon: väljundi read võivad olla pikemad kui 255 märki, mis on FreePascali vaikimisi kasutatava `String`-tüübi jaoks pikkuse ülempiir. Failis `katalah1.pas` asendab kompilaatori direktiiv `{$LONGSTRINGS ON}` tavalise sõnetüübi pikemaid väärtusi toetava tüübiga (mille nimi muidu oleks `AnsiString` ja millel on võrreldes standardse `String`-tüübiga teisi miinuseid). Teine võimalus oleks hoida taanete infot kataloogide nimedest eraldi mingis muud tüüpi massiivis.

Testid

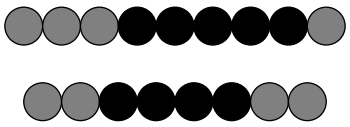
1. Minimaalne test ühe kataloogiga. Taandeid pole, seega töötavad ka need lahendused, mis eeldavad, et taande samm sisendis on üks tühik.
2. Maksimaalse sügavusega test. Väljundi read pikemad kui 250 märki. Taande samm sisendis on üks tühik.
3. Mitme vahetu alamkataloogiga kataloogid, vertikaalseid jooni ei ole. Taande samm sisendis on üks tühik.
4. Mitme vahetu alamkataloogiga kataloogid, vertikaalseid jooni ei ole samal real topelt. Taande samm sisendis on üks tühik.
5. Mitme vahetu alamkataloogiga kataloogid, vertikaalseid jooni on samal real mitu. Taande samm sisendis on üks tühik.
6. Sama test, mis eelmine, aga taanete samm on varieeruv.
7. Esimene kataloog ei ole kõige väiksema taandega. Ebaloogiline, aga ülesande tekstis antud definitsiooni järgi lubatud olukord.
8. Sisend, mis juba on kataloogipuu. Lahendused, mis kasutavad järgmiste ridade töötlemisel eelmistele ridadele nimede ette lisatud graafilist infot, võivad segadusse sattuda.
9. Maksimaalne kataloogide arv, palju tühikuid.
10. Maksimaalne kataloogide arv, pikad nimed.

Testid 1. . . 10 à 2 punkti, kokku 20 punkti.

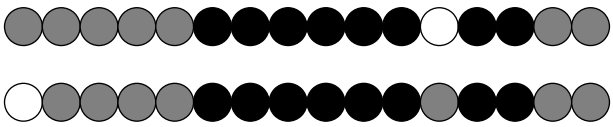
8. Maraton (edasijõudnute variant)

1 sekund 40 punkti

Selle ülesande lahendamisel on vaja kõigepelt panna tähele, et parimas paigutuses istuvad sõbrad alati nii, et nende vahele ei jää ühtki vaba kohta. Intuitiivselt tundub see ilmne, aga intuitivselt tundub ka, et lühem paigutus (see tähendab selline, kus minimaalse ja maksimaalse kohanumbri vahe on väiksem) peaks alati pikemast parem olema. Aga näiteks alloleval joonisel (kus hallid ringikesed tähistavad sõpru ja mustad võõraid nende vahel) on ülemisel real kujutatud paigutuse hinne 25 ja alumisel real kujutatud paigutuse hinne 26. Sellega oleme ka näidanud, et ülesande edasijõudnute rühma variant tõesti erineb põhikooli rühma omast ja põhikooli lahendus edasijõudnute omaks ei sobi.



Kui hakata otsime tõestust väitele, et rühma sisse vabade kohtade on alati kahjulik, siis nii lihtsalt, nagu põhikooli rühma variandis, läbi ei saa. Näiteks alloleval joonisel (kus lisaks eelnevale tähistavad valged ringikesed vabu kohti) on ülemisel real kujutatud paigutuse hinne 146 ja alumisel real kujutatud (eelmisest vasakpoolseima sõbra vabale kohale kolimise järel saadud) paigutuse hinne 148.



Kõigele eelnevale vaatamata väide siiski kehtib. Oletame, et meie K sõpra on paigutatud nii, et vabast kohast vasakul on k_1 ja sellest paremal $k_2 = K - k_1$ sõpra ja et $k_1 \geqslant k_2$ (nagu joonisel toodud näites; kui tegelikult on gruppide suurused vastupidi, tuleb järgnevas arutelus lihtsalt vasak ja parem pool ära vahetada). Tähistagu nüüd l vaba koha kaugust parempoolse grupi kõige vasakpoolsemast liikmest (joonisel toodud näites $l = 3$). Vaatame nüüd, mis juhtub paigutuse hindega selle sõbra vabale kohale istumisel: ta liigub kõigile vasakpoolse grupi liikmetele l kohta lähemale ja kõigist paigale jäänud parempoolse grupi liikmetest l kohta kaugemale; kui ülejäänud sõbrad ei liigu, siis nende omavahelised kaugused ei muutu ja siis on uue paigutuse hinne $h' = h - l \cdot k_1 + l \cdot (k_2 - 1) = h - l \cdot (k_1 - k_2 + 1)$, kus h on esialgse paigutuse hinne; kuna $k_1 \geqslant k_2$, siis $k_1 - k_2 + 1 > 0$ ja seega $h' < h$, järelikult selline ümberistumine annab parema paigutuse. Kui see liige on ümber istunud, jagab temast vabaks jäänud koht sõbrad kahte gruppi suurusega vastavalt $k_1 + 1$ ja $k_2 - 1$, kus juba kindlasti $k_1 + 1 > k_2 - 1$ ja järelikult võib jälle hinnet parandada, kui parempoolse grupi vasakpoolseim liige ümber istub. Niimoodi ükshaaval istubki kogu parempoolne grupp igaüks eelmise sõbra kohale. Kuna igal sammul paigutuse hinne paraneb, peab ta seda tegema ka kõigi sammude summas. Trikk on siin selles, et kui ümber istub suurema grupi liige, siis see võib olla kahjulik, aga kui ümber istub väiksema (või kahe võrdse suurusega grupi korral ükskõik kumma) grupi liige, siis see on alati kasulik.

Eelneva kokkuvõttes teame, et parima paigutuse otsimisel võime keskenduda ainult neile, milles sõbrad istuvad järjest täis kõik vabad kohad ja küsimus on ainult selles, millise numbriga kohast alustada. Failis `marelah1.pas` toodud lahendus proovib lihtsalt kõik võimalused läbi. Kuna võimalikke alguskohti on maksimaalselt N ja ühe paigutuse hinde arvutamiseks vaadatakse läbi kõik paarid, kulub kõigi paigutuste hindamiseks ja neist parima valimiseks kokku $O(N \cdot K^2)$ operatsiooni, mis tähendab, et sellise lahenduse eest saaks umbes 10 punkti.

Lahenduse parandamiseks uurime lähemalt, kuidas mõjutab hinnet ühe sõbra lisamine osalisse paigutusse. Täpsemalt, olgu meil juba paigutatud x sõpra ja olgu järgmine vaba koht eelmisest kaugusel l. Siis on uue sõbra $x+1$ kaugus igast juba varem paigutatud sõbrast l võrra suurem kui sõbral x. Seega, kui me teaks sõbra x paigutamise hinda h, saaks me sõbra $x+1$ paigutamise hinna arvutada kujul $h' = h+x \cdot l$. Tehes selle loogika läbi jooksvalt alates esimesest sõbrast (kelle puhul $x = 0$ ja $h = 0$), saamegi paigutuse hinna arvutada sõprade loetelu ühe läbimisega. Sellel ideel põhinev lahendus on toodud failis `marelah2.pas` ja kulutab kõigi paigutuste hindamiseks kokku $O(N \cdot K)$ operatsiooni, mis tähendab, et selle eest saaks umbes 20 punkti.

Aga tegelikult pole vaja eelmisest paigutusest järgmise saamiseks kogu tööd uuesti teha, piisab vaid esimese sõbra koha vabastamisest ja talle viimase sõbra järel uue vaba koha leidmisest. Sellise lahenduse efektiivseks realiseerimiseks on vaja kahte tähelepanekut. Neist esimene: eelmises lõigus kirjeldatud uue sõbra lisamise loogika on rakendatav ka tagurpidi, sõbra eemaldamise loogikana (kui meil on teada h', x ja l, saame nende kaudu avaldada h). Kui me tahame sõpru paigutusse lisada paremale, siis peame neid eemaldama vasakult. Et see oleks võimalik, ongi vaja teist tähelepanekut: kui kõige esimese võimaliku paigutuse summa arvutada lisaks välja ka paremalt vasakule, siis selle arvutuse lõpuks on meil käes esimese sõbra paigutusest eemaldamiseks vajalik info kujul, mis võimaldab mitte ainult eemaldamise loogikat vasakpoolsele sõbrale rakendada, vaid ka jooksvalt sama infot järgmise sõbra jaoks välja arvutada.

Seega, kui jooksva paigutuse alguse ja lõpu juures vastavalt esimese sõbra eemaldamise ja viimase lisamise hinda meeles pidada ja jooksvalt uuendada, tuleb kõigi selliste nihutamiste peale kokku iga koha andmeid ainult kaks korda vaadata (esimene kord sõbrale koha otsimisel ja teine kord pärast sõbra koha vabastamist järgmise sõbra otsimisel). Selline lahendus on toodud failis `marelah3.pas` ja on piisavalt efektiivne, et teenida maksimumpunktid.

Kuigi see võistluse tulemuse seisukohalt enam vajalik ei ole, saaks ka seda lahendust veel edasi arendada. Nimelt pole vabade kohtade arvestamiseks vaja eraldi muutujat igale kohale, piisab ainult nimekirjast, kus on järjestikustest vabadest kohtadest koosnevad lõigud. Selles nimekirjas navigeerimine on küll veidi keerulisem programmeerida, aga testides, kus on palju hõivatud kohti, võimaldaks niisugune lähenemine veel mõnevõrra aega kokku hoida.

Testid

1. $N = 1$, $M = 0$, $K = 1$. Minimaalne test. Üksik "sõber" saab ainsa koha, mis raja ääres üldse on.
2. $N = 12$, $M = 4$, $K = 4$. Ainult üks koht, kuhu sõbrad vahetult üksteise kõrvale mahuvad, mujal jääks võõraid vahele.
3. $N = 13$, $M = 5$, $K = 4$. Kolm kohta, kuhu sõbrad mahuvad nii, et ainult üks võõras on vahel, nende hinded on võrdsed, valida tuleb asukoha järgi.
4. $N = 15$, $M = 6$, $K = 4$. Piirjuht: parimad kohad vahetult finišis.
5. $N = 15$, $M = 6$, $K = 4$. Piirjuht: parimad kohad vahetult stardis.
6. $N = 15$, $M = 10$, $K = 5$. Piirjuht: sõbrad võtavad ära kõik vabad kohad.
7. $N = 20$, $M = 9$, $K = 4$. Kolm kohta, kuhu sõbrad mahuvad nii, et ainult kaks võõrast on vahel, neist keskmise hinne on parim.
8. $N = 29$, $M = 18$, $K = 4$. Kontranäide: lühemale lõigule paigutamine ei anna tingimata paremat hinnet.
9. $N = 100$, $M = 30$, $K = 30$. Väike juhuslik test.
10. $N = 1000$, $M = 100$, $K = 100$. Väike juhuslik test.
11. $N = 5000$, $M = 500$, $K = 500$. Mõõduka suurusega juhuslik test.
12. $N = 30\,000$, $M = 3000$, $K = 300$. Mõõduka suurusega juhuslik test.
13. $N = 100\,000$, $M = 10\,000$, $K = 10\,000$. Suur juhuslik test.
14. $N = 1\,000\,000$, $M = 100\,000$, $K = 100\,000$. Maksimaalse suurusega juhuslik test.

Testid 1…10 à 1 punkt, 11…12 à 5 punkti, 13…14 à 10 punkti, kokku 40 punkti.

9. Kombinatsioonid

1 sekund 40 punkti

Lihtne viis selle ülesande lahendamiseks on arvutada $C(N,K)$ välja vahetult definitsiooni järgi. Selline lahendus on toodud failis `komblah1a.cpp` ja teeniks 10 punkti.

Üks võimalus selle lahenduse parandamiseks on $C(N,K)$ avaldist taandada

$$\begin{aligned} C(N,K) &= \frac{N!}{K!\cdot(N-K)!} \\ &= \frac{1\cdot 2\cdot\ldots\cdot(N-1)\cdot N}{1\cdot 2\cdot\ldots\cdot(K-1)\cdot K\cdot(N-K)!} \\ &= \frac{(K+1)\cdot(K+2)\cdot\ldots\cdot(N-1)\cdot N}{1\cdot 2\cdot\ldots\cdot(N-K-1)\cdot(N-K)} \\ &= (K+1)/1\cdot(K+2)/2\cdot\ldots\cdot(N-1)/(N-K-1)\cdot N/(N-K) \end{aligned}$$

ja arvutada saadud tulemuse järgi. Paneme tähele, et jagamised tulevad kõik kindlasti täpselt välja: $K+1$ jagub arvuga 1; $K+1$ ja $K+2$ hulgas on kindlasti üks, mis jagub arvuga 2; $K+1$, $K+2$ ja $K+3$ hulgas on kindlasti üks, mis jagub arvuga 3 j.n.e. Selline lahendus on toodud failis `komblah1b.cpp` ja teeniks 20 punkti.

Teine võimalus $C(N,K)$ väljaarvutamist parandada on lähtuda faktist, et kui nummerdada Pascali kolmnurga read $0\ldots$ ja i. rea elemendid $0\ldots i$, siis on kolmnurga N. rea K. element parajasti $C(N,K)$. Sellel tähelepanekul põhinev lahendus on failis `komblah1c.cpp` ja teeniks samuti 20 punkti.

Parema lahenduse leidmiseks lähtume aritmeetika põhiteoreemi nime all tuntud faktist, et mistahes positiivne täisarv A on täpselt ühel viisil esitatav korrutisena kujul

$$A = p_1^{a_1}\cdot p_2^{a_2}\cdot\ldots\cdot p_m^{a_m},$$

kus p_i on kasvavalt järjestatud algarvud ja a_i on positiivsed täisarvud. Jaguvuse põhiomadustest on üsna lihtne tuletada, et arvu A mistahes jagaja B peab sel juhul avalduma kujul

$$B = p_1^{b_1}\cdot p_2^{b_2}\cdot\ldots\cdot p_m^{b_m},$$

kus b_i on mittenegatiivsed täisarvud ja iga i korral $b_i \leqslant a_i$. See omakorda tähendab, et arv B^L on arvu A jagaja ainult juhul kui iga i korral $L\cdot b_i \leqslant a_i$, ehk $L \leqslant \min\{\lfloor a_i/b_i\rfloor\}$, kus kirjutis $\lfloor a_i/b_i\rfloor$ tähendab jagatise a_i/b_i täisosa ehk ümardamist allapoole lähima täisarvuni.

Teiselt poolt, kui p on algarv, siis esineb ta arvu $N!$ tegurina kõigepealt ühe korra igas p kordses teguris (mida on $\lfloor N/p\rfloor$), seejärel lisaks veel ühe korra igas p^2 kordses teguris (mida on $\lfloor N/p^2\rfloor$), j.n.e., kokku seega astmel

$$\lfloor N/p\rfloor + \lfloor N/p^2\rfloor + \ldots.$$

Taandamiste tõttu esineb algarv p seega arvu $C(N,K)$ tegurina astmel

$$(\lfloor N/p\rfloor + \lfloor N/p^2\rfloor + \ldots) - (\lfloor K/p\rfloor + \lfloor K/p^2\rfloor + \ldots) - (\lfloor (N-K)/p\rfloor + \lfloor (N-K)/p^2\rfloor + \ldots).$$

Eelnevatel ideedel põhinev lahendus on toodud failis `komblah2.cpp` ja see on piisavalt efektiivne, et teenida maksimumpunktid.

Testid

1.–10. Väikesed testid. $N < 20$, seega $N! < 2^{63}$. Lahendatavad $C(N,K)$ vahetult definitsiooni järgi välja arvutades ja korduvalt M-ga jagades.

11.–15. Keskmised testid. $C(N,K) < 2^{63}$. Lahendatavad $C(N,K)$ ettevaatlikult välja arvutades.

16.–25. Suured testid. Lahendatavad tegurite kordsuse analüüsiga.

Testid $1\ldots 10$ à 1 punkt, $11\ldots 15$ à 2 punkti, $16\ldots 25$ à 2 punkti, kokku 40 punkti.