1. Bensiinihind

Kõige lihtsam viis selle ülesande lahendamiseks on vaadelda järjest kõiki päevi ja uurida iga päeva kohta, kui kaua on sellest päevast alates bensiinihind järjest tõusnud (täpsemalt küll mittelangenud). Samamoodi võime sisendjada teist korda läbi vaadates leida ka pikima langusperioodi. Failis `bhlah0.pas` toodud lahendus nii teebki.

Eelmise lahenduse analüüsimisel torkab silma, et pikkade tõusude või langustega sisendi korral teeb see üsna palju tühja tööd. Nimelt, kui meil on juba teada, et päevast A alustades jätkub tõus kuni päevani L, siis mistahes vahepealse päeva B jaoks (ehk siis $A < B \leqslant L$ korral) on kohe teada, et päevast B alustades jätkub ka tõus kuni päevani L. Kuna intervall $B \ldots L$ peab kindlasti olema lühem kui intervall $A \ldots L$, on selge, et neid vahepealseid päevi pole mõtet vaadata ja intervalli $A \ldots L$ tuvastamise järel võime järgmise intervalli otsimist alustada päevast $L + 1$. Failis `bhlah1.pas` toodud lahendus nii teebki.

Edasisel analüüsimisel näeme, et iga langus $H_L > H_{L+1}$ eraldab kaht tõusuperioodi: üks lõpeb päevaga L, teine algab päevaga $L + 1$ (nii üks kui teine võivad olla ka vaid ühe päeva pikkused; kuna päeva jooksul bensiinihind ei muutu, võime iga päeva vaadelda nii 1-päevase tõusu- kui ka langusperioodina). See tähendab, et kui peame meeles viimase languse aega (maksimaalset E, mille korral $H_{E-1} > H_E$) ja iga sisendist loetud elementi talle vahetult eelnevaga võrreldes uue languse (minimaalse $L > E$, mille korral $H_L > H_{L+1}$) leiame, teame kohe, et intervall $E \ldots L$ on olnud järjekordne tõus. Pidades analoogiliselt meeles ka eelmise tõusu aega, saamegi failis `bhlah2.pas` toodud lahenduse, mis on võimeline analüüsima kuitahes pikki jadasid, kasutades selleks ainult paari abimuutujat.

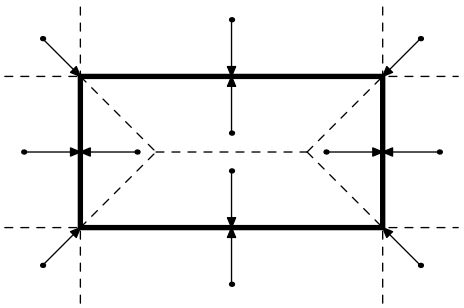
Testid

1. $N = 1$. Minimaalne test. 2 punkti.
2. $N = 7$. Väike lihtne test, üks tõus, üks langus. 2 punkti.
3. $N = 10$. Üks tõus, üks langus, vahepeal püsiv hind. 2 punkti.
4. $N = 16$. Mitmed tõusud, mitmed langused, kõik erineva pikkusega. 3 punkti.
5. $N = 26$. Mitmed tõusud, mitmed langused, pikkused korduvad. 3 punkti.
6. $N = 500$. Keskmise suurusega juhuslik test. 4 punkti.
7. $N = 10\,000$. Maksimaalse suurusega juhuslik test. 4 punkti.

Kokku 20 punkti.

2. Hiirekursor

Allolevalt jooniselt peaks olema üsna lihtne loendada, et selles ülesandes on kokku ainult kaksteist põhimõtteliselt erinevat võimalust, kuidas kursor ja aken teineteise suhtes paikneda võivad: kaheksa juhtu, kus kursor on aknast väljas, ja neli juhtu, kus kursor on akna sees.



Failis `hklah1.pas` toodud lahendus kontrollib ükshaaval tingimusi läbi vaadates, millisega neist juhtudest on tegu ja lahendab igaühe individuaalselt.

Failis `hklah2.pas` toodud lahendus seevastu lähtub tähelepanekust, et otsitav punkt asub ülesande püstituse kohaselt ühel akna neljast äärest. Järelikult piisab, kui leiame igal akna äärel kursori algasukohale lähima punkti ja valime väljastamiseks neist neljast punktist omakorda kursorile lähima.

Testid

1. Kursor algseisus akna kohal. 2 punkti.
2. Kursor algseisus akna alumisel serval. 2 punkti.
3. Kursor algseisus aknast diagonaalis all vasakul. 2 punkti.
4. Kursor algseisus akna ülemisel vasakul nurgal. 2 punkti.
5. Kursor algseisus akna all parema serva pikendusel. 2 punkti.
6. Kursor algseisus akna sees vasaku serva lähedal. 2 punkti.
7. Kursor algseisus akna sees diagonaaljoonel ülemisest ja paremast servast võrdsel kaugusel. Vastus pole ühene. 2 punkti.
8. Kursor algseisus akna sees horisontaaljoonel ülemisest ja alumisest servast võrdsel kaugusel. Vastus pole ühene. 2 punkti.
9. Kursor algseisus akna sees ülemisest, vasakust ja paremast servast võrdsel kaugusel. Vastus pole ühene. 2 punkti.
10. Kursor algseisus ruudukujulise akna sees kõigist servadest võrdsel kaugusel. Vastus pole ühene. Suured arvud, kauguse ruut ei mahu 16-bitisesse muutujasse. 2 punkti.

Kokku 20 punkti.

3. Ajaintervallid

Üks võimalus seda ülesannet lahendada on panna tähele, et kuuel juhul seitsmest tähendab ajahetke allapoole "ümardamine" selle lõpust kindla pikkusega osa asendamist kindla konstandiga. Seitsmendal (5-minutise intervalli) juhul tuleb osata valida kahe erineva konstandi vahel. Sellel tähelepanekul põhinevad lahendused on toodud failides `ailah1.pas` ja `ailah1.c`.

Keskkondades, kus on olemas paindlikumad sisendi ja väljundi töötlemise vahendid, võib aasta, kuu, päeva, tunni, minuti ja sekundi väärtused "kujunduse" vahelt välja parsida, vajalikud neist nullida või ümardada ja siis tulemuse esialgsele kujule tagasi vormindada. Nii ongi tehtud failis `ailah2.c`.

Testid

1. Intervall minut, vaja sekundid nullida. 2 punkti.
2. Intervall 5 minutit, vaja minutid ümardada. 2 punkti.
3. Intervall 5 minutit, vaja sekundid nullida. 2 punkti.
4. Intervall tund, vaja minutid nullida. 2 punkti.
5. Intervall tund, vaja sekundid nullida. 2 punkti.
6. Intervall päev, vaja tunnid nullida. 2 punkti.
7. Intervall päev, vaja tunnid, minutid ja sekundid nullida. 2 punkti.
8. Intervall kuu, vaja päevad nullida. 2 punkti.
9. Intervall kuu, vaja päevad, tunnid, minutid ja sekundid nullida. 2 punkti.
10. Intervall aasta, vaja kõik madalamad järgud nullida. 2 punkti.

Kokku 20 punkti.

4. Kaubaveod

Selle ülesande lahendamiseks on võimalik kas otse kasutada või vähese vaevaga kohandada päris mitmeid erinevaid klassikalisi graafialgoritme.

Üks võimalus on kasutada korduvalt graafi sidususe tuvastamist, võttes iga kord arvesse erinevat servade alamhulka. Peaks olema selge, et kui arvestades servi alates kaalust c on graaf sidus ja arvestades servi alates kaalust $c+1$ mittesidus, siis ongi c meie otsitav vastus. Sellel ideel põhinev lahendus, mis proovib järjest c väärtusi $0, 1, 2, \ldots$ on toodud failis `kvlah1a.c`. Seda lahendust võib optimeerida, kasutades sobiva c väärtuse leidmiseks lineaarse otsingu asemel kahendotsingut, nagu on tehtud failis `kvlah1b.c`. Muidugi võiks graafi sidususe kontrolli realiseerida sügavuti läbimise asemel ka laiuti läbimisega.

Teine võimalus oleks modifitseerida Floyd-Warshalli algoritmi[1]. Originaalis leiab see algoritm n tipuga graafis iga tipu kauguse igast teisest järgmiselt:

- alustame $n \times n$ maatriksist A, kus A_{ij} näitab otse tipust i tippu j viiva serva pikkust (kui tipust i tippu j serva ei ole, siis $A_{ij} = \infty$);
- eeldades, et meil on iga tipupaari jaoks juba teada kaugused mööda teid, mis võivad vahepeatustena läbida ainult tippe $1 \ldots k-1$, konstrueerime sellest kaugused, mis võivad läbida ka tippu k; selleks vaatleme kõiki paare (i,j) ja asendame

  $$A_{ij} \leftarrow \min(A_{ij},\, A_{ik} + A_{kj});$$

  teisisõnu, vaatleme lühimat nende teede hulgas, mis tippe $k \ldots n$ ei kasuta ja võrdleme seda parima teega, mis kasutab ka tippu k;
- rakendades eelmist sammu $k = 1, 2, \ldots, n$ korral, saamegi lõpuks maatriksisse A lühimad teed kõigi tipupaaride vahel.

Meie ülesande lahendamiseks võib seda muuta järgmiselt:

- algseisus on A_{ij} linnast i linna j viiva tee kandevõime (kui teed ei ole, siis $A_{ij} = 0$);
- asendussammuks võtame

  $$A_{ij} \leftarrow \max(A_{ij},\, \min(A_{ik}, A_{kj}));$$

  ehk siis, vaatleme maksimaalse kandevõimega teed linnast i linna j, mis kasutab ainult linnu $1 \ldots k-1$ ja võrdleme seda parima teega, mis läbib ka linna k; ilmselt määrab marsruudi $i \to k \to j$ kandevõime nõrgem lõikudest $i \to k$ ja $k \to j$).

Selline lahendus ongi toodud failis `kvlah2.c`.

Veel võiks selle ülesande lahendamiseks kohandada kõiki minimaalse toese leidmise algoritme, mis lisavad uusi servi pooleliolevasse toesesse nende kaalude kasvamise järjekorras. Sellised on näiteks Kruskali[2] ja Primi[3] algoritmid. Muidugi tuleks nende algoritmide kasutamisel meie ülesande jaoks servad järjestada kaalude kahanemise järjekorras.

Veel võiks lisada, et esmapilgul lihtsa lahendusena tunduv "min-max" algoritm — leiame igast linnast väljuvate teede kandevõimete maksimumi ja väljastame vastusena minimaalse nende maksimumide hulgast — ei ole tegelikult lahendus. See annab vale tulemuse juba nelja linna ja kolme tee korral, kui näiteks linnade 1 ja 2 ning 3 ja 4 vahel olevate teede kandevõimed ületavad linnade 2 ja 3 vahel oleva tee oma.

[1] `http://en.wikipedia.org/wiki/Floyd-Warshall_algorithm`
[2] `http://en.wikipedia.org/wiki/Kruskal's_algorithm`
[3] `http://en.wikipedia.org/wiki/Prim's_algorithm`

Testid

1. $N = 2$, $M = 1$. Minimaalne test. 4 punkti.
2. $N = 5$, $M = 4$. Teedevõrk on puu. 4 punkti.
3. $N = 5$, $M = 6$. Pärast kordsete eemaldamist on teedevõrk puu. 4 punkti.
4. $N = 6$, $M = 6$. Teedevõrk on tsükkel. 4 punkti.
5. $N = 6$, $M = 15$. Teedevõrk on täisgraaf. 4 punkti.
6. $N = 6$, $M = 15$. Teedevõrk on täisgraaf, aga vastus ei ole min-max. 4 punkti.
7. $N = 30$, $M = 100$. Keskmise suurusega juhuslik test, kordseid teid ei ole. 4 punkti.
8. $N = 30$, $M = 100$. Keskmise suurusega juhuslik test, kordsed teed. 4 punkti.
9. $N = 100$, $M = 1000$. Maksimaalse suurusega juhuslik test, kordseid teid ei ole. 4 punkti.
10. $N = 100$, $M = 1000$. Maksimaalse suurusega juhuslik test, kordsed teed. 4 punkti.

Kokku 40 punkti.

5. Rubiku kuubik

Rubiku kuubikul on mitmeid invariante, mis kõigi lubatud teisenduste (nii käikude kui ka terve kuubiku pööramise) käigus ei muutu:

- Ükski ruut kuubiku pinnal ei muuda oma värvi. Järelikult peab mistahes korrektses seisus alati igat värvi ruute olema täpselt neli. (Sellega saab tuvastada ühe vigase sisendi.)
- Ükski ühise servaga ruudupaar kuubiku pinnal ei lähe lahku. Järelikult peab mistahes korrektses seisus alati igal värvil olema nelja erinevat värvi naabreid, igaüht kaks korda. (Sellega saab tuvastada kolm vigast sisendit.) Enamgi veel, värvipaarid peavad olema täpselt samad, mis nõutaval lõppseisul. (Sellega saab tuvastada neli vigast sisendit.)
- Ükski ühise tipuga ruudukolmik kuubiku pinnal ei lähe lahku. Järelikult peavad mistahes korrektses seisus alati esinema samad värvikolmikud, mis nõutaval lõppseisul. (Sellega saab tuvastada neli vigast sisendit.) Enamgi veel, värvid peavad igas kolmikus säilitama oma järjekorra, näiteks tipust vaadates päripäeva lugedes. (Sellega saab tuvastada viis vigast sisendit.)
- Lisaks eelnevale ei ole võimalik muuta ainult ühe väikese kuubi orientatsiooni. Loendades kolmandikpöörded, mille võrra kõik väikesed kuubid kokku oma normaalasendist kõrvale kalduvad, peab see arv alati jaguma kolmega[4]. Seda tunnust ei saa kasutada sisenditel, millel väikeste kuupide hulk ei lange kokku nõutava lõppseisu omadega (kui mingit kuupi üldse pole, ei saa ka tuvastada, kas ta on oma õiges asendis või keeratud päri- või vastupäeva), aga viimase vigase sisendi saab selle tunnuse alusel tuvastada küll.

Kõigi nende tingimuste kontrollimine on realiseeritud failis `rklah0.pas`.

Positiivsed testid võib lahendada kas laiuti otsimisega (igast seisust genereerime kõik uued, mis on sellest saavutatavad ühe käiguga ja mis pole veel varem esinenud), nagu on tehtud failides `rklah1.cpp` ja `rklah1.pl`, või iteratiivse süvendamisega (variantide läbivaatuse üldistus: alguses vaatame läbi kõik seisud, mis on saavutatavad ühe käiguga; kui otsitavat nende hulgas ei ole, vaatame läbi seisud, mis on saavutatavad kahe käiguga jne), nagu on tehtud failis `rklah2.c`. Mõlemal juhul saab vigase sisendi tuvastada selle järgi, et otsing jõuab olukorda, kus uusi seise enam ei ole, aga lahendust ka veel ei ole. Muidugi saab selle põhjal ainult tuvastada, et sisend on vigane, mitte aga aru saada, mis seal valesti on.

Töömahu hindamiseks paneme tähele, et etteantud väikestest kuupidest saame kuubikut konstrueerida järgmiselt:

- esimeseks nurgatükiks valime ühe kindla väikese kuubi mingis kindlas orientatsioonis (sest me võime kuubikut keerata suvalisse suunda ja see ei anna uut seisu);
- järgmise tüki paigaldamiseks on 7 erinevat kohta ja igas kohas 3 erinevat orientatsiooni;
- järgmise tüki jaoks 6 kohta ja 3 orientatsiooni;
- jne...

Kokku saame seega $7! \cdot 3^7 = 11\,022\,480$ võimalust väikestest kuupidest kuubikut kokku laduda. Nagu eelnevast teada, ei ole Rubiku kuubiku sihipärasel kasutamisel viimase väikese kuubi orientatsioon enam vabalt valitav, seega on tegelikult saavutatavate seisude arv $7! \cdot 3^6 = 3\,674\,160$.

Kuna vigaste sisendite ilmutatud tuvastamine on palju efektiivsem kui otsingu lõpuni töötada laskmine (sõltuvalt realisatsioonist võtab see aega mõni minut kuni tund), siis oleks kiirema lahenduse saamiseks kasulikum alustada just negatiivsete testide tuvastamisest.

[4] `http://www.ryanheise.com/cube/cube_laws.html`

Testid

1. Vigane algseis. Võimatu nurgakolmik. Sisendis esinb tükk `ABD`, aga nõutaval lõppseisul ei ole tahkudel `A`, `B` ja `D` ühist tippu.
2. Vigane algseis. Kuubiku saab panna kokku seisu `AAAABBBBEEEEDDDDCCCCFFFF`. Nõutava lõppseisuga võrreldes on kaks vastastahku omavahel vahetatud.
3. Korrektne algseis. Optimaalne lahendus 5 käiguga.
4. Vigane algseis. Kuubiku saab panna kokku seisu `AAAAFFFFCCCCDDDDEEEEBBBB`. Nõutava lõppseisuga võrreldes on kaks naabertahku omavahel vahetatud.
5. Vigane algseis. Sisendis on `C` ruute 3 ja `F` ruute 5 tükki.
6. Korrektne algseis. Optimaalne lahendus 10 käiguga.
7. Korrektne algseis. Optimaalne lahendus 11 käiguga (mis on ka maksimaalne võimalik).
8. Korrektne algseis. Optimaalne lahendus 1 käiguga.
9. Vigane algseis. Võimatu nurgakolmik. Sisendis esineb tükk `AAF`, aga nõutaval lõppseisul on iga värv tervenisti ühel tahul.
10. Korrektne algseis. Optimaalne lahendus 7 käiguga.
11. Vigane algseis. Kuubiku saab panna kokku seisu `EAAAABBBCCCCDDDDEBEEFFFF`. Nõutava lõppseisuga võrreldes on tükk $\widehat{\text{ABE}}$ oma kohal pöördunud.
12. Korrektne algseis. Optimaalne lahendus 4 käiguga.

Hindamine.

- Sisendite klassifitseerimine korrektseteks ja ebakorrektseteks: 8 punkti igale võistlejale, kes esitas õiged vastused vähemalt kahele korrektsele ja kahele ebakorrektsele testile ning ei esitanud ühtegi vale vastust.
- Ebakorretsete sisendite ebakorrektsuse põhjendused: 4 punkti iga veenva põhjenduse eest, kokku $6 \cdot 4 = 24$ punkti.
- Korretsete sisendite lahendamine: 4 punkti iga lahendatud testi eest, kokku $6 \cdot 4 = 24$ punkti.
- Korretsete sisendite lahendamine minimaalse sammude arvuga: 4 lisapunkti iga optimaalselt lahendatud testi eest, kokku $6 \cdot 4 = 24$ punkti.

Kokku 80 punkti.