

Zadatke, test primjere i rješenja pripremili: Alen Rakipović <alen.rakipovic@gmail.com>, Goran Gašić <goran.gasic@gmail.com>, Ivo Sluganović <ivo.sluganovic@gmail.com>, Tomislav Gudlek <tgudlek@gmail.com>, Ivan Mandura <ivan.mandura93@gmail.com>, i Ante Đerek <ante.derek@gmail.com>.

Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima koji nužno ne odgovaraju u svim detaljima ovdje opisanim algoritmima.

DISKID

Predložio: Ante Đerek

Potrebno znanje: simulacija, jednodimenzionalna polja, cijelobrojno dijeljenje, osnovne petlje

Zadatak se rješava direktnom simulacijom odnosno implementacijom postupka računanja DiskId-a koji je opisan u tekstu zadatka - najprije izračunamo tri dijela X, Y, Z te ih potom pretvorimo i ispišemo u heksadekatskom obliku sa potrebnim brojem vodećih nula

Računanje brojeva Y i Z je lagano, potrebna je jedna 'for' petlja za prvi, a drugi je jednostavno broj N. Računanje broja X je samo nešto teže jer trebamo izračunati zbroj dekatskih znamenki za svaki početak pjesme. Zbroj dekadskih znamenaka broja možemo naći tako da unutar petlje ponavljamo postupak uzimanja ostataka pri djeljenju sa 10, te potom podijelimo dani broj sa 10 - tako jednu za drugom dobivamo dekatske znamenke odostraga. Pametna implementacija će ovaj dio napisati u posebnoj funkciji koju će onda pozivati unutar for petlje.

Ispis heksadekatskih znamenaka broja možemo obaviti na više načina: Ako radimo u jeziku C ispis obavljam trivijalno korištenjem format stringova - na primjer, broj X možemo ispisati u heksadekatskom obliku sa točno dvije znamenke (tj. tako da se doda vodeća nula ako je potrebna) sa `printf("%02x", X)`. Bez format stringova pretvaranje i ispis nije puno teže: Kako bi dobili na primjer četiri znamenke (uljučujući možebitne vodeće nule) broja Y, potrebno je četiri puta izračunati ostatak pri djeljenju sa 16 te podijeliti Y sa 16. Dakle, to je postupak koji je istovjetan gore opisanom postupku rastavka broja na dekatske znamenke kako bi izračunali zbroj znamenki - samo koristimo bazu 16 umjesto 10.

Za one koji žele više

Pretpostavi da nije poznat poredak pjesama na albumu nego samo njihovo trajanje. Zadano je 1000 diskid-ova kandidata, odredi barem jedan kojemu odgovara neki raspored (permutacija) pjesama. Preuzmi bazu diskid-ova sa <http://ftp.freedb.org/pub/freedb/> te napravi program koji automatski organizira tvoju kolekciju digitalne muzike.

KVADRATI

Predložio: Goran Žužić

Potrebno znanje: rekurzija, ispitivanje svih kombinacija, analiza složenosti

Činjenica da je najveći mogući broj koraka 3 nas vodi na ideju potpunog pretraživanja:

1. na sve moguće načine pokušamo postaviti jedan kvadrat
 - a. za svaki izgenerirani način provjerimo gradi li on konačnu sliku
2. ukoliko nismo uspjeli na sve moguće načine pokušavamo postaviti 2 kvadrata
 - a. za svaki izgenerirani način provjerimo gradi li on konačnu sliku
3. ukoliko nismo uspjeli na sve moguće načine pokušavamo postaviti 3 kvadrata
 - a. za svaki izgenerirani način provjerimo gradi li on konačnu sliku

Broj načina na koji možemo postaviti jedan kvadrat je nešto manji od N^3 (N^2 načina da postavimo gornji lijevi vrh te najviše N načina da odaberemo duljinu stranice), dok za provjeru gradi li tako izgenerirani način konačnu sliku trebamo otprilike N^2 operacija (prolaz po cijeloj slici). Dok će ovaj pristup raditi zadovoljavajuće dobro za prva dva koraka, njegova složenost u trećem koraku je otprilike $N^3 N^3 N^2 = N^{11} \sim 10^{10}$, što je otprilike 2 reda veličine previše da bi algoritam bio dovoljno brz.

Kao optimizaciju toga postupka dovoljno je uočiti da se, primjerice, prvi korak algoritma (određivanje može li se slika izgraditi od jednog kvadrata) može implementirati u N^2 koraka umjesto u $N^3 N^2$ na sljedeći način: odredimo gornji lijevi i donji desni označeni vrh te potom provjerimo A) opisuju li ta dva vrha kvadrat te B) jesu li svi preostali označeni vrhovi unutar tog pronađenog kvadrata. Potpuno analogna ideja se može upotrijebiti i za preostale korake što ukupnu složenost spušta s N^{11} na N^8 , što je dovoljno dobro da se dobiju svi bodovi.

Za one koji žele više

Ukoliko bi broj kvadrata koje je Mirko odabrao bio mnogostruko veći (npr. veći od 100), nije jako vjerojatno da optimalno rješenje za takav problem uopće postoji (pod pretpostavkom da nam rješenje koje se doslovno izvršava milijardu godina nije adekvatno). Ipak, u takvim slučajevima možemo daleko doći sa suboptimalnim heuristikama: brzim rješenjima koje nam u relativno kratkom vremenu daju rješenje koje je lošije od optimalnog za određeni postotak. Sličan zadatak ovome je dan na Međunarodnoj informatičkoj olimpijadi 2002. godine u Koreji (jedina razlika, osim mnogo većih ograničenja, je što umjesto kvadrata postavljamo pravokutnike) - zadatak XOR (<http://ioinformatics.org/locations/ioi02/contest/day1/xor/xor.pdf>). Za efikasni heuristički pristup prema takvom vrlo teškom zadatku pogledate elegantno rješenje predloženo nakon spomenute olimpijade: <http://olympiads.win.tue.nl/ioi/ioi2002/contest/report.pdf> (stranice 32.-48.)

PAKETI

Predložio: Ivo Sluganović

Potrebno znanje: binarno pretraživanje

Primjetimo najprije da formula koja određuje mogu li čokolade stati u kutiju određenog kapaciteta ima svojstvo da je 'rastuća', odnosno ako dodajemo nove čokolade vrijednost formule će uvijek rasti, a nikada padati.

Pretpostavimo da nam je unaprijed zadan kapacitet K i želimo odrediti koliko možemo najviše napraviti poklon paketa sa zadanim čokoladama. Zbog navedenog svojstva formule, ovo možemo izračunati *pohlepnim* algoritmom: dodajemo čokolade u trenutnu kutiju dok možemo, kada premašimo kapacitet uzimamo novu kutiju. Na ovaj način pomoću jedne for petlje možemo implementirati funkciju **broj_paketa(K)** koja za zadani K vraća najveći broj paketa.

Međutim u zadatku se traži obrnuta stvar, potrebno je naći minimalni K tako da je moguće zapakirati P paketa. Jedno rješenje se odmah nameće, isprobavamo jedan po jedan broj kao kandidat za K , pozivamo funkciju **broj_paketa** te stajemo kada ona vrati rezultat koji je veći od P . Ovo rješenje je presporo za ograničenja u zadatku ali ipak dovoljno za 50% bodova.

Za potpuno rješenje zadatka potrebno je primjetiti da je **broj_paketa** padajuća funkcija te najmanji mogući kapacitet K pronaći metodom binarnog pretraživanja.