Opisi algoritama

Zadatke, testne primjere i rješenja pripremili: Fabijan Bošnjak, Nikola Dmitrović, Karlo Franić, Marin Kišić, Ivan Paljak i Stjepan Požgaj. Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima.

Zadatak: Ogledalo

Pripremio: Marin Kišić
Potrebno znanje: naredbe učitavanja i ispisivanja, rad s riječima

Možemo primijetiti da će se uvijek unositi četiri riječi i da je potrebno ispisati zadnju riječ bez zadnjeg znaka. Pola bodova moglo se osvojiti ispisivanjem podriječi koja se proteže od 18. do 22. znaka rečenice iz ulaza.

Programski kod (pisan u `Python 3`):

```
print(input().split(' ')[-1][:-1])
```

Zadatak: Ulica

Pripremio: Marin Kišić
Potrebno znanje: petlje, provjera parnosti

Napravit ćemo pomoćno polje `moze` i za svaki broj x iz ulaza ćemo postaviti `moze[x]=1`. Nakon toga, prebrojimo ima li više parnih ili neparnih brojeva u u ulazu. Na kraju, ako ima više parnih, krenemo od 2 i idemo tako dugo dok je `moze[trenutni_broj]=1` krećući se po parnim brojevima. Analogno za neparne, samo počinjemo od 1.

Zadatak: Datumi

Pripremio: Karlo Franić
Potrebno znanje: provjera palindromičnosti, ad-hoc

Za prvi podzadatak potrebno je uzeti prva dva znaka datuma i taj broj povećavati za 1 dok ne nađemo palindrom.

Za drugi podzadatak koristimo znanje prvog podzadatka, uz to svaki put kada dan dođe do zadnjeg u mjesecu (4. i 5. znak), postavimo dan na 1, a mjesec povećamo za 1.

Za treći podzadatak potrebno je povećavati i godinu svaki put kad dođemo do 31. dana u 12. mjesecu.

Za sve bodove bilo je potrebno primijetiti da palindromičnih datuma postoji 366. Svaki datum ima točno jednu godinu s kojom tvori palindrom. Te datume je moguće staviti u polje i nakon toga za svaki zadani datum iterativno po polju naći prvi datum koji je veći od zadanog.

Vremenska složenost je $\mathcal{O}(NK)$, gdje K predstavlja broj palindromičnih datuma. Zadatak se može riješiti i u složenosti $\mathcal{O}(N \log K)$, a to rješenje ostavljamo čitateljici za vježbu.

Zadatak: Birmingham

Pripremio: Marin Kišić
Potrebno znanje: osnove teorije grafova, pretraga u širinu (BFS)

Za pola bodova najprije trebamo odrediti polje `dist[a][b]` koje nam govori najmanju udaljenost između čvorora a i b. To možemo tako da iz svakog čvora posbeno pustimo BFS prema svim ostalim čvorovima. Nakon toga pustimo poseban BFS iz početnih čvorova koji se širi na sljedeći način: ako smo trenutno u čvoru a s udaljenosti d, onda u red ubacimo sve one čvorove koji još nisu posjećeni i koji su od a udaljeni za $\leq (d-1) \cdot K$ koraka. Njih nađemo tako da prođemo po nizu `dist[a]`.

Za sve bodove bilo je potrebno uočiti da, ako znamo udaljenost čvora x do nekog od početnih čvorova, onda odmah možemo reći i koje je rješenje za njega (formulom ili simulacijom, za implementacijske detalje pogledati službeno rješenje ili rješenja natjecatelja koji su osvojili sve bodove). Udaljenosti do početnih čvorova mogli smo odrediti tako da pustimo BFS iz početnih čvorova.

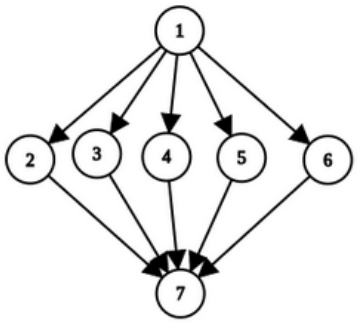
Zadatak: Konstrukcija

Pripremio: Stjepan Požgaj
Potrebno znanje: ad-hoc

Moramo konstruirati graf s N čvorova i M bridova tako da je $tns(1, N) = K$, gdje je $tns(x, y) = \sum_{C \in S_{x,y}} sgn(C)$. Neka je $[a, b>$ skup čvorova x takvih da postoji put od a do x, postoji put od x do b i x je različit od b. Svakom uređenom nizu $C \in S_{x,y}$ možemo maknuti zadnji član, te tako dobiti uređeni niz C' koji počinje u x, završava u nekom čvoru $[x, y>$ i čija je duljina za jedan manja od duljine uređenog niza C pa vrijedi $sgn(C) = -sgn(C')$. Zbog toga je $tns(x, y) = -\sum_{z \in [x,y>} tns(x, z)$.

Naš graf gradit ćemo po razinama. Na prvoj razini će uvijek biti samo čvor 1, a na zadnjoj razini samo čvor N. U prva dva podzadatka će svi čvorovi (osim, naravno, N) s neke razine imati usmjerene bridove prema svim čvorovima na sljedećoj razini. U općenitom rješenju ćemo trebati malu modifikaciju.

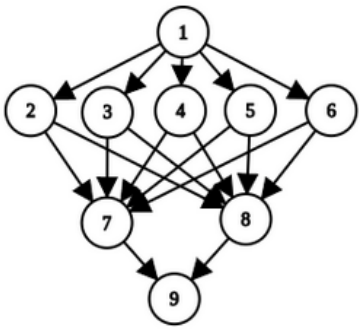
Graf koji je rješenje za prvi podzadatak $1 \leq K < 500$ imat će tri razine. Na prvoj samo čvor s oznakom 1, na drugoj $K + 1$ čvorova, a na zadnjoj čvor s oznakom $K + 3$. Primjerice, za $K = 4$ graf izgleda kao na slici dolje.



U ovom primjeru, koristeći rekurzivnu relaciju za $tns(1, x)$, vidimo da je $tns(1, 1) = 1$, $tns(1, 2) = \cdots = tns(1, 6) = (-1) \cdot 1$, $tns(1, 7) = (-1) \cdot (1 + 5 \cdot (-1)) = 4$.

Slično rješavamo i drugi podzadatak gdje vrijedi $-300 < K \leq -1$. Primjerice, za $K = -4$ graf izgleda kao na slici dolje.

Promatrajući konstrukcije za prva dva podzadatka možemo uočiti da, dodavanjem M čvorova na novu razinu, sumu svih vrijednosti $tns(1, x)$ množimo s $-(M - 1)$. Također je iz rekurzivne relacije jasno da je $tns(1, N)$ jednak sumi svih vrijednost $tns(1, x)$, gdje je $1 \leq x < N$, pomnoženoj s -1. Ako je broj K oblika $\pm 2^i$, dodavanjem tribrida na novu razinu, sumu vrijednosti $tns(1, x)$ množimo s -2 pa tako s $3 + (i - 1) \cdot 9$ bridova možemo dobiti vrijednost K, do na predznak. Ako nam predznak nije dobar, na



sljedeću razinu dodajemo 2 čvora da bismo sumu vrijednosti pomnožili s -1. Sad još moramo odrediti kako apsolutnu vrijednost trenutne sume vrijednosti povećati za 1. Pomoću toga možemo konstruirati graf algoritmom analognim onom u kojem množenjem s 2 i eventualnim dodavanjem jedinice iz broja u binarnom zapisu dobivamo njegovu vrijednost u dekadskom zapisu. Ako je trenutna suma vrijednosti negativna, jednostavno joj možemo oduzeti 1 tako da na trenutnu razinu dodamo čvor koji je spojen samo sa čvorom 1. Isto tako, ako je ona pozitivna, možemo dodati 2 čvora na novu razinu spojena sa svima s prethodne da tu vrijednost učinimo negativnom te tada na isti način kao u prethodnom slučaju oduzeti 1.

Ovako opisan algoritam za dani K konstruira graf s manje od $16 \cdot \log_2(K)$ bridova.

Zadatak: Skandi

Pripremili: Fabijan Bošnjak i Ivan Paljak
Potrebno znanje: maksimalno sparivanje u bipartitnom grafu, minimalni vršni pokrivač, Königov teorem

Za osvajanje bodova na prvom podzadatku bilo je dovoljno primijetiti da je, zbog malog broja jedinica, broj pitanja u skandinavci vrlo malen. Bilo je dovoljno na sve moguće načine pretpostaviti na koja ćemo pitanja ponuditi odgovor te potom, za svaki način, provjeriti je li cijela skandinavka ispunjena. Vremenska složenost je $\mathcal{O}(2^Q NM)$ gdje sa Q označavamo ukupan broj pitanja u skandinavci.

Ograničenja drugog podzadatka će iskusnog natjecatelja odmah navesti na pravi put. Matrica je duguljasta, naziru se bitmaske, mora da se radi o dinamičkom programiranju. I bi tako, kretat ćemo se matricom „redak po redak" te ćemo pamtiti u kojim smo stupcima odlučili odgovoriti na okomita pitanja čiji se odgovori protežu kroz trenutni redak. Dakle, stanje označavamo sa `dp[red][maska]` koje nam govori na koliko najmanje pitanja je potrebno odgovoriti ako smo popunili sve retke skandinavke do retka $red - 1$, nalazimo se u retku red, a u masci su navedeni nezavršeni odgovori na okomita pitanja. Razradu prijelaza kao i rekonstrukciju ostavit ćemo čitateljici za vježbu. Implementaciju ovog rješenja vremenske složenosti $\mathcal{O}(N2^M)$ možete vidjeti u izvornom kodu `skandi_dp.cpp`.

Ponekad je korisno formalnije zapisati ono što se od nas u zadatku traži. U ovom slučaju, taj će nas pristup prirodno odvesti do potpunog rješenja. Označimo *vodoravna* pitanja sa $a_1, a_2, \ldots, a_p$, a okomita pitanja sa $b_1, b_2, \ldots, b_q$. Neka je a_i (odnosno b_j) jednak 1 ako smo na to pitanje ponudili odgovor, odnosno neka je jednak 0 ako to nismo napravili. Primijetimo da za svako prazno polje x moramo odabrati barem jedno od točno dva pitanja u čijem se odgovoru to polje nalazi. Dakle, za svako prazno polje postoje neka dva pitanja a_i i b_j za koja mora vrijediti $a_i \vee b_j = 1$.

Modelirajmo sada ovaj problem bipartitnim grafom u kojem se s jedne strane nalaze čvorovi koji predstavljaju vodoravna, a s druge strane čvorovi koji predstavljaju okomita pitanja. Povežimo neka dva čvora a_i i b_j bridom ako postoji neko prazno polje koje uvjetuje da vrijedi $a_i \vee b_j = 1$. Ono što želimo napraviti jest odrediti najmanji mogući skup čvorova takav da je svaki brid u grafu incidentan (susjedan) barem jednom čvoru iz tog skupa. Dakako, radi se poznatom problemu traženja minimalnog vršnog pokrivača (*engl.* minimum vertex cover). Budući da je graf bipartitan, možemo iskoristiti Königov teorem i zadatak riješiti u polinomijalnoj složenosti.

Zadatak: Trener

Pripremio: Karlo Franić
Potrebno znanje: teorija grafova, dinamičko programiranje

Za prvi podzadatak potrebno je provjeriti svaku mogućnost. Složenost je $\mathcal{O}(K^N)$.

Za druga dva podzadatka bilo je potrebano izgraditi acikličan usmjereni graf (DAG) i na njemu prebrojiti broj puteva od čvorova s prve razine do čvorova sa zadnje razine. To prebrojavanje može se rješiti dinamičkim programiranjem i prepuštamo ga čitateljici za vježbu. Objasnimo sada kakav točno DAG želimo izgraditi.

Na svakoj od N razina imat ćemo čvor za svako različito prezime koje se pojavljuje na toj razini. Vrijednost čvora bit će broj istih prezimena koje on predstavlja. Veze ćemo graditi od čvorova na razini i prema čvorovima na razini $i+1$. Veza od A prema B će postojati ako prezime koje predstavlja čvor A se razlikuje u točno jednom slovu od prezima koje predstavlja čvor B. U drugom podzataku te veze smo mogli ručno izraditi, međutim za sve bodove bilo je potrebno korištenje efikasnih ugrađenih struktura podataka (npr. `std::map` ili `std::unordered_map` u jeziku C++) ili ručno hashiranje podataka.