Opisi algoritama

Zadatke, testne primjere i rješenja pripremili: Nikola Dmitrović, Karlo Franić, Gabrijel Jambrošić, Marin Kišić, Daniel Paleka, Ivan Paljak i Paula Vidas. Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima.

Zadatak: Steak

Pripremio: Nikola Dmitrović
Potrebno znanje: naredba učitavanja brojeva i riječi, naredba ispisivanja, naredba odlučivanja, naredba ponavljanjaj

Na prvo čitanje se čini da ćemo za redni broj dana u godini trebati uzeti u obzir i mjesec, tj. da ćemo prvo morati odrediti redni broj zadanog mjeseca, taj redni broj pomnožiti s 30 i na to dodati datum. Npr. dan `16 LIPANJ` je $30 \cdot 6 + 16 = 196$. dan u godini (lipanj je šesti mjesec).

Međutim, kako je dogovoreno da će svaki mjesec imati 30 dana, možemo zaključiti da se parnost/neparnost dana D iz teksta zadatka neće mijenjati u ovisnosti o mjesecu u kojem se nalazi. Primjerice, ako promatramo slučaj kada je $D = 16$, redni broj toga dana u siječnju će biti 16, u veljači 46, u ožujku 76, itd., sve parni brojevi kao što je i sam broj 16.

U testnim primjerima vrijednima 10 bodova nije bilo nužno koristiti naredbu ponavljanja.

Programski kod (pisan u `Python 3`):

```
N = int(input())
for i in range(N):
    D = int(input())
    M = input()
    if D % 2 == 0:
        print('BROKULA')
    else:
        print('MRKVA')
```

Zadatak: Duel

Pripremili: Karlo Franić i Ivan Paljak
Potrebno znanje: jednodimenzionalna polja

Za 6 bodova bilo je dovoljno provjeriti jesu li Fabijan i Patrik riješili oba ista zadatka ili je svatko riješio jedan koji drugi nije uspio riješiti.

Za dodatnih 12 bodova bilo je potrebno stvoriti dva polja (jedno za Patrika, a drugo za Fabijana) u kojima ćemo s 1 označiti riješen zadatak, a s 0 onaj koji nije riješen. Zatim smo mogli naredbom ponavljanja proći kroz oba polja te u svakom prebrojati koliko ima jedinica, a da se na istoj poziciji u drugom polju nalazi nula.

Za sve bodove u polje je bilno potrebno spremati indekse zadataka te naredbom ponavljanja za svaki zadatak koji je Patrik riješio provjeriti nalazi li se isti zadatak u popisu Fabijanovih zadataka. Isti proces tada ponavljamo za Fabijanove zadatke.

Također za zadatak postoji alternativno rješenje koje prebroji koliko su istih zadataka riješili. Odgovor na pitanje zašto nam je ta informacija korisna ostavljamo čitateljici za vježbu.

Zadatak: Emacs

Pripremila: Paula Vidas
Potrebno znanje: naredba ponavljanja, rad sa stringovima

Broj pravokutnika na slici jednak je broju gornjih lijevih kuteva pravokutnika. Polje (i, j) je gornji lijevi kut nekog pravokutnika ako u tom polju piše `'*'`, a u poljima $(i-1, j)$ i $(i, j-1)$ piše `'.'`. Naravno, moramo dodatno pripaziti na polja u prvom retku i prvom stupcu.

Alternativno, možemo BFS algoritmom pronaći broj povezanih područja znakova `'*'`.

Zadatak: Političari

Pripremio: Ivan Paljak
Potrebno znanje: iskorištavanje cikličnosti

Na ovom zadatku bilo je moguće osvojiti 50% (35) bodova naivnom implementacijom onog što piše u tekstu zadatka. Odnosno, bilo je potrebno ispravno implementirati pravila kojima se političari međusobno optužuju te tako saznati koji će se političar pojaviti u K-toj emisiji. Vremenska složenost ovog pristupa je $\mathcal{O}(K)$, a implementaciju možete vidjeti u izvornom kodu `politicari_brute.cpp`.

Za osvajanje svih bodova bilo je potrebno primijetiti da je proces cikličan. Budući sljedeći gost ovisi o isključivo tome tko je bio prethodni gost te tko je tog gosta optužio, zaključujemo da postoji $\mathcal{O}(N^2)$ različitih emisija (stanja). Dakako, dvije emisije smatramo jednakima ako u njima gostuje isti političar koji je optužen od strane iste osobe. U protivnom, emisije smatramo različitima.

Budući da je broj stanja znatno manji od rednog broja emisije koja nas zanima u tekstu zadatka, možemo simulirati proces iz zadatka sve dok se ponovno ne dogodi emisija koju smo već vidjeli. U tom trenutku, ako smo do sad zapamtili neke relevantne informacije, možemo metodama matematike i računanja odrediti tko će gostovati u K-toj emisiji.

Pretpostavimo da smo simulirajući proces primijetili da je i-ta emisija jednaka nekoj prethodnoj j-toj emisiji. U tom slučaju smo ušli u ciklus veličine $(i-j)$ te zaključujemo da je gost K-te emisije jednak gostu koji se pojavio u $(j + ((K-j)\%(i-j)))$-toj emisiji, gdje znakom % označavamo operator modulo.

Vremenska složenost ovakvog algoritma je $\mathcal{O}(N^2)$.

Zadatak: Matching

Pripremili: Paula Vidas i Daniel Paleka
Potrebno znanje: tournament stablo

Ako povežemo točke koje imaju jednaku jednu koordinatu, točke će biti podijeljene u *cikluse* i *puteve*. Za puteve je određeno koje točke moramo spojiti. Ako postoji put sa neparno mnogo točaka odgovor je odmah `"NE"`. Za cikluse moramo odrediti hoćemo li povući sve vodoravne ili sve okomite dužine (u nastavku teksta to zovemo *orijentacijom*).

Ciklusa ima najviše $\frac{N}{4}$, pa za podzadatak ($N \leq 40$) možemo isprobati svih $2^{\text{broj ciklusa}}$ mogućnosti za orijentacije ciklusa, i provjeriti postoji li neka mogućnost u kojoj se dužine ne sijeku.

Glavna ideja je sljedeća: na početku putevi određuju neke dužine koje moramo nacrtati, pa onda te dužine određuju orijentacije onih ciklusa koje sijeku i crtamo te dužine, itd. Ako u nekom trenutku nacrtamo dužinu koja siječe neku već nacrtanu ili dobijemo da neki ciklus ne može biti niti okomit niti vodoravan, onda je odgovor `"NE"`.

Za podzadatak $N \leq 2000$ to možemo napraviti u složenosti $O(N^2)$:

Nađemo puteve i cikluse, te dužine iz puteva ubacimo u *queue* dužina koje treba nacrtati. Sve dok queue nije prazan, uzimamo dužinu iz queuea. Provjerimo siječe li se s nekim već nacrtanim dužinama (moguće

je i da smo za neki ciklus nacrtali obje orijentacije, pa će se sijeći u točki, i tada je odgovor `"NE"`). Zatim prođemo po svih ciklusima kojima još nismo odredili orijentaciju, te ako dužina siječe ciklus, moramo ga orijentirati suprotno od dužine i ubaciti te nove dužine u queue. Na kraju se eventualno može dogoditi da za neke cikluse nismo odredili orijentaciju, pa njih sve orijentiramo jednako.

Za $N \leq 10^5$, moramo pametnije napraviti provjeru siječe li se dužina s nekim drugim dužinama. Pomoću tournament stabla napravit ćemo strukturu koja podržava:

- dodaj dužinu $(x_1, y) - (x_2, y)$
- makni dužinu $(x_1, y) - (x_2, y)$ (koja je nekada prije ubačena)
- odredi za neke x i y_1 najmanji $y_2 \geq y_1$ takav da postoji dužina s y koordinatom jednakom y_2 koja sadrži točku (x, y_2) (ili odredi da takav y_2 ne postoji)

U čvoru tournamenta pamtit ćemo skup svih y koordinata dužina kojima su x koordinate u intervalu za koji je odgovoran taj čvor. Prva dva upita su ubacivanje (odnosno izbacivanje) y u određene čvorove. Za odgovor na treći upit, nađemo takav y_2 za svaki čvor odgovoran za x (pomoću `lower_bound`-a u `std::set`-u), i uzmemo minimalni. Složenost sva tri upita je $\mathcal{O}(\log^2 N)$. Za implementaciju pogledajte službeni kod.

Dužina koja siječe dužinu $(x, y_1) - (x, y_2)$ postoji ako je odgovor na treći upit za x i y_1 manji ili jednak od y_2.

Imat ćemo dvije strukture – za okomite i za vodoravne dužine – u kojima ćemo spremiti dužine još neorijentiranih ciklusa, te dvije strukture u kojima spremamo nacrtane dužine. Razradu detalja ostavljamo čitatelju na vježbu.

Zadatak: Putovanje

Pripremili: Gabrijel Jambrošić i Marin Kišić
Potrebno znanje: lca, manje na veće

Primjetimo da je pravi problem u zadatku saznati koliko puta ćemo proći svakom cestom. Kada to znamo, lako možemo odrediti želimo li kupiti višekratnu ili jednokratnu kartu za svaku cestu.

Za prvi podzadatak bilo je dovoljno samo nekim algoritmom poput dfs-a ili bfs-a pronaći put između X i $X + 1$ te brojač svih cesta na putu povećati za 1.

U drugom podzadatku nemamo stablo, nego lanac. Zapišimo taj lanac kao niz. Sada za svaki par X i $X + 1$ možemo u nekom nizu `dp` na poziciji `min(poz[X], poz[X+1])` povećati vrijednost za 1, a na poziciji `max(poz[X], poz[X+1])` smanjiti vrijednost za 1. Nakon što smo to napravili, za sve X od 1 do $(N - 1)$, imat ćemo varijablu `cnt`, prolaziti nizom `dp` te `cnt` mjenjati za vrijednost `dp[i]` (`cnt += dp[i]`). Primjetimo da zapravo sada u svakom trenu u varijabli `cnt` piše koliko puta smo tu cestu prešli.

Za sve bodove ćemo ovaj algoritam prilagoditi da radi na stablu. Ukorijenimo stablo u proizvoljnom čvoru. Neka je L prvi zajednički predak (*lowest common ancestor (LCA)*) od X i $(X + 1)$. Povećajmo sada `dp[X]` za 1, `dp[X+1]` za 1, te smanjimo `dp[L]` za 2. Sada kada bismo za neku cestu uzeli sumu svih `dp` vrijednosti u podstablu te ceste zapravo bismo dobili koliko puta smo prošli tu cestu.

Ovo rješenje je implementirano u `putovanje_lca.cpp`

Vremenska složenost je $\mathcal{O}(N \log N)$.

Postoji i alternativno rješenje iste vremenske složenosti koje koristi *manje na veće* trik, a implementacija tog rješenja se nalazi u `putovanje_stl.cpp`, a za idejske detalje pogledajte rješenje jako sličnog zadatka opisano na ovom linku.

Zadatak: Zapina

Pripremili: Marin Kišić i Paula Vidas
Potrebno znanje: dinamičko programiranje, prebrojavanje

Prvi podzadatak možemo riješiti tako da za svaki mogući raspored provjerimo postoji li sretan informatičar. Mogućih rasporeda ima N^N.

Drugi podzadatak možemo riješiti koristeći formulu uključivanja-isključivanja. Ako s $a(S)$, gdje je $S \subseteq \{1, 2, \ldots, N\}$, označimo broj rasporeda u kojima su svi informatičari iz skupa S sretni, tada je broj dobrih rasporeda jednak

$$\sum_{S \subseteq \{1,2,\ldots,N\}} (-1)^{|S|+1} a(S).$$

Neka je $S = \{s_1, s_2, \ldots, s_k\}$. Ako je $s_1 + s_2 + \cdots + s_k > N$ onda je očito $a(S) = 0$. Inače, vrijedi

$$a(S) = \binom{N}{s_1}\binom{N-s_1}{s_2}\cdots\binom{N-(s_1+s_2+\cdots s_{k-1})}{s_k}(N-k)^{N-(s_1+s_2+\cdots+s_k)}.$$

Povrhe možemo preprocessati koristeći relaciju $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$, u složenosti $\mathcal{O}(N^2)$.

Ukupna vremenska složenost je $\mathcal{O}(N2^N)$.

Treći podzadatak rješavamo dinamičkim programiranjem. Neka je $dp(n, k)$ broj raspodjela k zadataka na n informatičara u kojima je barem jedan informatičar sretan. Imamo dvije mogućnosti – hoćemo li n-tom informatičaru dati točno n zadataka (pa da bude sretan), ili nećemo. U prvom slučaju je, ako je $k \geq n$, broj načina jednak

$$\binom{k}{n}(n-1)^{(k-n)},$$

a u drugom

$$\sum_{0 \leq i \leq k, i \neq n} \binom{k}{i} dp(n-1, k-i).$$

Ukupna vremenska složenost je $\mathcal{O}(N^3)$.