Univerzitetni programerski maraton 2019
3. kolo — rešitve nalog

Tomaž Hočevar
tomaz.hocevar@fri.uni-lj.si

3. oktober 2019

Papajščina

Implementirajte, kar zahteva naloga. Izpisujete lahko črko po črko in v primeru samoglasnika izpišete še dve dodatni, ali pa si pomagate z ukazom `replace` oz. njegovim ekvivalentom v vašem programskem jeziku.

Pot v f(x)

Izračunati moramo vrednosti polinoma na intervalu $[l, r]$ s korakom 0.1. Poiščemo največjo in najmanjšo vrednost ter izpišemo še par vrstic nad in pod intervalom - za vsako točko mreže lahko namreč določimo pravilen znak.

Težava nastane pri delu z decimalnimi števili, kjer pride do težav z natančnostjo, na kar vas prijazno opozori že primer. Pomagamo si lahko s primerjavami z upoštevano računsko napako, npr. $\epsilon = 10^{-9}$. Še bolj zanesljiva možnost pa je, da opravimo vse izračune s celimi števili.

Bashevski triki

Naloga testira znanje rekurzije, hkrati pa morate poskrbeti za cel kup grdobij, ki so v pomoč ilustrirane že v podanih primerih. Najprej morate poiskati prvi podniz, nato preveriti, ali ustreza pogojem za razvoj, iterirati čez vse možnosti z intervala in logiko rekurzivno ponoviti na preostanku niza.

Alternativna rešitev je, da dobro poznate svoj programski jezik in znate uporabiti orodja, ki so na voljo. Npr. v Pythonu lahko poiščete vse relevantne podnize z regularnim izrazom in preverite, kateri ustrezajo vsem pogojem. Namesto rekurzivnega sestavljanja rešitev pa uporabite funkcijo `product` iz modula `itertools`, ki to naredi namesto vas.

Jez

Na prvi pogled je težko določiti, kje in za koliko dvigniti jez, da bo njegov najnižji del čim višji. Bolj enostavno je preveriti, ali lahko jez dvignemo na višino H. Najbolj levi del jezu, ki je prenizek, moramo dvigniti za $H - h_i$. Hkrati se nam splača dvigniti tudi naslednjih $K - 1$ delov. Če potrebujemo kvečjemu M dvigov, smo uspešni, sicer pa ne. Nalogo lahko rešimo z bisekcijo po iskani višini H.

Pobeg iz projektivnega prostora

Izračunati moramo razdaljo med premico in več pravilnimi večkotniki. Če je število oglišč večkotnika majhno, lahko izračunamo kar razdalje med vsemi njegovimi oglišči in premico. Sicer pa lahko večkotnik aproksimiramo s krogom in izračunamo razdaljo med krogom in premico. V obeh primerih moramo poskrbeti še za situacijo, ko premica seka večkotnik.

Pomemben faktor je, kdaj se odločimo za katero od omenjenih možnosti. Večkotnik lahko aproksimiramo s krogom, če ima veliko oglišč ali pa če je dovolj majhen. Iz teh dveh podatkov lahko za vsak večkotnik izračunamo, ali je aproksimacija s krogom še dopustna (znotraj dovoljene napake).

Ptiči

Veje drevesa modeliramo kot vozlišča, izraščanje vej pa kot povezave v grafu, ki ima obliko drevesa s korenom, ki ustreza deblu. Učinkovito moramo ugotoviti, kje na poti od vozlišča u proti korenu prvič naletimo na vozlišče, ki ima prenizko nosilnost.

Iskanje lahko pospešimo tako, da si za vsako vozlišče vnaprej izračunamo minimume na poteh dolžine 2^i, podobno kot to naredimo pri računanju najnižjega skupnega prednika dveh vozlišč v drevesu. Tako potrebujemo največ $O(\log n)$ skokov za odgovor na posamezno poizvedbo, podatke pa si naračunamo v $O(n \log n)$.