17. tekmovanje ACM v znanju računalništva za srednješolce

26. marca 2022

NALOGE ZA DRUGO SKUPINO

Odgovore lahko pišeš/rišeš na papir ali pa jih natipkaš z računalnikom ali pa oddaš del odgovorov na papirju in del prek računalnika. Vse te možnosti so enakovredne. Če oddajaš kaj na papirju, napiši na vsak oddani list svoje ime. Pri delu si lahko pomagaš s prevajalniki in razvojnimi orodji, ki so na voljo na tvojem računalniku, vendar bomo tvoje odgovore v vsakem primeru pregledali in ocenili ročno (ne glede na to, ali si jih oddal prek računalnika ali na papirju), zato manjše napake v sintaksi ali pri klicih funkcij standardne knjižnice niso tako pomembne, kot bi bile na tekmovanjih z avtomatskim ocenjevanjem.

Tekmovanje bo potekalo na strežniku `https://rtk.fri.uni-lj.si/`, kjer dobiš naloge in oddajaš svoje odgovore. Uporabniška imena in gesla vas bodo čakala na mizi v učilnici. Pri oddaji preko računalnika odpreš dotično nalogo v spletni učilnici in rešitev natipkaš oz. prilepiš v polje za programsko kodo. Med tipkanjem se rešitev na približno dve minuti samodejno shrani. Poleg tega lahko sam med pisanjem rešitve izrecno zahtevaš shranjevanje rešitve s pritiskom na gumb „Shrani spremembe“. Ker je vgrajeni urejevalnik dokaj preprost in ne omogoča označevanja kode z barvami, predlagamo, da rešitev pripraviš v kakšnem drugem urejevalniku na računalniku (Visual Studio Code, Geany, Lazarus) in jo nato prekopiraš v okno spletnega urejevalnika. Naj te ne moti, da se bodo barvne oznake kode pri kopiranju izgubile.

Ko si bodisi zadovoljen z rešitvijo ter si zaključil nalogo ali ko želiš začasno prekiniti pisanje rešitve naloge ter se lotiti druge naloge, uporabi gumb „Shrani spremembe“ in nato klikni na „Nazaj na seznam nalog“, da se vrneš v glavni meni. (Oddano rešitev lahko kasneje še spreminjaš.) Za vsak slučaj priporočamo, da pred oddajo shraniš svoj odgovor tudi v datoteko na svojem lokalnem računalniku.

Med reševanjem lahko vprašanja za tekmovalno komisijo postavljaš prek zasebnih sporočil na tekmovalnem strežniku (ikona oblačka zgoraj desno) ali pa vprašaš člane komisije, ki bodo prisotni v učilnicah. Prek zasebnih sporočil bomo pošiljali tudi morebitna pojasnila in popravke, če bi se izkazalo, da so v besedilu nalog kakšne nejasnosti ali napake. Zato med reševanjem redno preverjaj, če so se pojavila kakšna nova zasebna sporočila. Če imaš težave z računalnikom ali s povezavo s spletnim strežnikom za oddajo nalog in komunikacijo s tekmovalno komisijo, se nemudoma obrni na nadzornika v učilnici, ki bo zagotovil drug računalnik. **Če zaradi morebitnih težav pri oddajanju rešitev na strežnik želiš, da ocenimo odgovore v datotekah na lokalnem disku tvojega računalnika, o tem obvezno obvesti nadzorno osebo v svoji učilnici, še preden odideš iz nje.**

Svoje odgovore dobro utemelji. Če pišeš izvorno kodo programa ali podprograma, **OBVEZNO** tudi v nekaj stavkih z besedami opiši idejo, na kateri temelji tvoja rešitev. Če ni v nalogi drugače napisano, lahko tvoje rešitve predpostavljajo, da so vhodni podatki brez napak (da ustrezajo formatu in omejitvam, kot jih podaja naloga). Zaželeno je, da so tvoje rešitve poleg tega, da so pravilne, tudi učinkovite; bolj učinkovite rešitve dobijo več točk (s tem je mišljeno predvsem, naj ima rešitev učinkovit algoritem; drobne tehnične optimizacije niso tako pomembne). **Nalog je pet** in pri vsaki nalogi lahko dobiš od 0 do 20 točk. Liste z nalogami lahko po tekmovanju obdržiš.

Rešitve bodo objavljene na `https://rtk.ijs.si/`. Predvidoma nekaj dni po tekmovanju bodo tam objavljeni tudi rezultati.

1. Varnostno kopiranje

Podan imamo seznam poti do več map oz. imenikov (direktorijev) na računalniku, ki jih želimo kopirati na zunanji disk za varnostno kopijo. Toda na seznamu so tudi podvojene mape ter mape, ki so že podmape drugih map. Če imamo na seznamu mape `/home/user/`, `/home/user` in `/home/user/slike`, je to v resnici enako seznamu samo s `/home/user/`, saj se prvi dve poti nanašata na isto mapo, zadnja pa je podmapa in je ni treba kopirati posebej, saj bo skopirana, ko bomo kopirali njeno starševsko mapo.

Napiši program, ki prebere seznam poti in izpiše le tiste izmed njih, ki jih je treba skopirati, da se izognemo nepotrebnemu kopiranju. Pri tem ni pomemben vrstni red izpisanih poti, važno je le, da so podmnožica podanih poti in da jih je čim manj. Če obstaja več enako dobrih rešitev, je vseeno, katero izpišeš.

Predpostaviš lahko, da so poti na voljo v datoteki `poti.txt` in vsaka pot je v svoji vrstici. Zagotovljeno je, da se vse poti začnejo s poševnico „`/`“, imena map pa bodo vsebovala le male črke angleške abecede, številke ter podčrtaj „`_`“. Vse poti bodo veljavne (ni ti treba npr. skrbeti zaradi poti oblike `abc//def`). Dolžina vsake poti bo manjša od 4096 znakov.

Poti je lahko veliko, zato naj bo tvoj program čim bolj učinkovit.

Primer vhoda:

```
/home/admin/config
/home/user/slike
/home/user/slike_stare
/home/user/dokumenti/sola/
/home/user/dokumenti/sola/slo/spisi/
/home/admin/config/
/home/user/minecraft/savegames/svet1
/home/user/minecraft/savegames/svet2
/home/user/slike/2019/smucanje/
/var/www/web/
/home/admin/config/
```

Možen izhod (vrstni red ni pomemben):

```
/home/user/slike
/home/user/slike_stare
/home/user/dokumenti/sola/
/home/admin/config/
/home/user/minecraft/savegames/svet1
/home/user/minecraft/savegames/svet2
/var/www/web/
```

2. Luči

Dano je zaporedje n luči; znano je njihovo začetno stanje (katere so prižgane in katere ugasnjene) ter želeno končno stanje. Osnovna operacija, ki jo lahko nad lučmi izvajamo, je, da si izberemo števili i in j (z območja $1 \leq i \leq j \leq n$) in spremenimo stanje vseh luči od vključno i-te do vključno j-te (torej ugasnemo tiste med njimi, ki so bile prej prižgane, in prižgemo tiste, ki so bile prej ugasnjene).

Napiši podprogram (oz. funkcijo) Luci(n, zacetno, koncno), ki izpiše zaporedje čim manj takšnih operacij, s katerimi lahko luči iz začetnega stanja spravimo v želeno končno stanje. Za vsako operacijo naj izpiše števili i in j, torej številko prve in zadnje spremenjene luči (luči so oštevilčene od 1 do n). Če obstaja več enako dobrih rešitev z najmanjšim možnim številom operacij, je vseeno, katero izpiše. Kot parametra zacetno in koncno naj tvoj podprogram sprejme niza n znakov, ki opisujeta začetno in končno stanje luči (znak `'P'` predstavlja prižgano luč, znak `'U'` pa ugasnjeno).

Dobro tudi **utemelji**, zakaj tvoja rešitev res najde najmanjše možno število operacij.

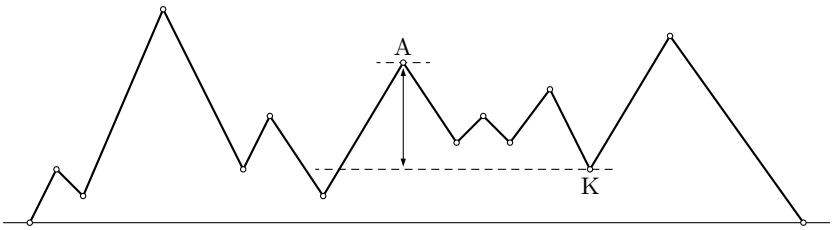
Primer: če imamo $n = 7$ luči in je začetno stanje `UUPPPUP`, končno pa `UPPUPPU`, potrebujemo vsaj tri operacije. Eno od možnih zaporedij treh operacij je: $i = 2$, $j = 5$ (dobimo `UPUUUUP`); $i = 4$, $j = 7$ (dobimo `UPUPPPU`); $i = 3$, $j = 4$ (dobimo `UPPUPPU`).

3. Planinarjenje

Za planince je pri izbiri ciljev ponavadi bolj zanimiv vrh, ki se izraziteje dviga višje od svoje okolice, manj pomembna pa je njegova absolutna nadmorska višina.

V ta namen je vpeljan pojem *topografske prominence* vrha (ali njegova *relativna višina*). Ta je določena z višinsko razdaljo (razliko višin) med vrhom in najnižjo tako plastnico (izohipso) terena, ki obkroža ta vrh in hkrati ne obkroža kakšnega višjega vrha. Z drugimi besedami: če bi se morska gladina dvignila do tiste plastnice, bi bil ta vrh najvišja točka otoka.

Problem si poenostavimo v dve dimenziji. Na spodnji sliki je prominenca vrha A razlika med nadmorskima višinama točk A in K (če bi se voda dvignila do višine točke K oz. tik nad njo, bi bila točka A najvišji vrh svojega otoka):



Da se ne trudimo z iskanjem vrhov in dolin, so podatki že pripravljeni tako, da si v seznamu (ali vhodni datoteki) sledijo izmenoma nadmorske višine dolin in vrhov. (Na gornji sliki so to točke, označene s krožci ∘, tako da bi za ta primer dobili seznam 15 višin.) Podatkov je liho število, seznam pa se začne in konča z dolino na nadmorski višini 0 (vse ostale višine so večje od 0).

Napiši program ali podprogram (funkcijo), ki bo za vsak vrh iz seznama ugotovil in izpisal njegovo prominenco. (Na primeru iz gornje slike bi moral program torej izpisati rezultate za 7 vrhov.) Tvoj (pod)program lahko seznam dobi kot parameter ali globalno spremenljivko (vektor, tabelo ali kaj podobnega), lahko pa ga prebere s standardnega vhoda ali iz vhodne datoteke (kar ti je lažje).

4. Sedežni red

Zloglasni 5. c je dobil novo učiteljico. Ta se je odločila, da bo v razredu naredila red, in sicer tako, da bo otroke posedla na točno določen način. Po novih pravilih noben otrok ne sme sedeti za otrokom, ki je strogo višji kot on sam, prav tako pa drug poleg drugega v isti vrsti ne smeta sedeti dva dečka ali dve deklici, ker bi bilo v tem primeru zagotovo preveč klepetanja.

Primera dobrih sedežnih redov za 6 otrok, kjer črka predstavlja spol, številka pa višino:

```
143M   150Z
139Z   128M
129M   127Z
  (TABLA)

139Z   154M
135Z   148M
135Z   129M
  (TABLA)
```

Dva primera slabih postavitev:

```
130Z   154M
135Z   148M
130Z   129M
  (TABLA)
```

(V gornji postavitvi je učenka levo v zadnji vrsti nižja od učenke pred njo.)

```
150Z   154M
145Z   148Z
141M   129M
  (TABLA)
```

(V tej postavitvi v prvi in drugi vrsti drug poleg drugega sedita otroka istega spola.)

Opiši postopek (ali napiši (pod)program oz. funkcijo, če ti je lažje), ki prejme podatke o višinah in spolu vseh u otrok v razredu ter jih razporedi v n vrst in m stolpcev tako, kot si je to zaželela učiteljica (ali pa ugotovi, da takšen razpored sploh ne obstaja). Za število stolpcev m predpostavi, da je sodo. Če je možnih več različnih pravilnih postavitev, je vseeno, katero izmed njih najde tvoj postopek. S podrobnostmi branja vhodnih podatkov in izpisa rezultatov se ti ni treba ukvarjati.

5. Žabe

Nad močvirjem leta r rojev muh. Vsak roj je sestavljen iz zelo velikega števila muh. Gibanje posameznega roja opišemo z zaporedjem koordinat in časov, ko se roj na svoji poti za trenutek ustavi. Tako bi postanke i-tega roja muh opisali s seznamom trojic $(x_{i,j}, y_{i,j}, t_{i,j})$, ki predstavljajo j-ti postanek i-tega roja na koordinati $(x_{i,j}, y_{i,j})$ ob času $t_{i,j}$. Predpostavimo lahko, da so postanki na poti posameznega roja podani po naraščajočih časih, torej $t_{i,j} < t_{i,j+1}$.

Poleg muh se v močvirju nahaja tudi z žab, ki lovijo muhe. Žaba lahko ulovi muho iz roja samo v trenutku, ko se roj ustavi, če se ravno takrat nahaja na istem mestu kot roj. Poznamo tudi lokacije žab: i-ta žaba se trenutno (ob času 0) nahaja na koordinati (a_i, b_i). Vse žabe se lahko premikajo po močvirju s hitrostjo 1 enote na sekundo (lahko pa tudi stojijo pri miru). Žaba se lahko med poljubnima dvema točkama premika v ravni črti, torej za merjenje razdalje uporabi evklidsko razdaljo (Pitagorov izrek).

Žabe načrtujejo lov na muhe. Ulovile bodo nekaj muh, nato pa se zbrale v koordinatnem izhodišču $(0, 0)$, kjer si bodo plen razdelile. Vsaka žaba bo ujela največ eno muho. Ker so muhe dobro rejene, je žabam dovolj, če vse skupaj ujamejo k muh (velja $k \leq z$), ki si jih nato razdelijo.

Opiši in utemelji **postopek**, ki bo določil najkrajši čas, v katerem se lahko vse žabe zberejo v izhodišču in pri tem skupaj ulovijo k muh. Koordinate in časi so realna števila.