Računalništvo (Fizika): 1. izpit

21. januar 2012

Čas reševanja: 120 minut. Doseženih 100 točk šteje za maksimalno oceno.

Rešitev je potrebno oddati na spletni učilnici kot eno ZIP ali 7z datoteko, ki vsebuje rešitve posameznih nalog v ločenih datotekah. Posamezne datoteke poimenuj `PriimekIme_Nx`, kjer je `x` zaporedna številka naloge.

Naloga 1 [25 točk]

Pripravi delovni zvezek v *Mathematici*, v katerem rešiš naslednjo nalogo:

Naj bo $f(x) = \dfrac{x^2 - 7x + 10}{x^2 - 9}$.

(a) Nariši graf funkcije f in pri tem računsko določi ničle in pole.

(b) Zapiši enačbo tangente t na graf funkcije f v točki $(1, f(x))$.

(c) Določi ploščino lika, ki ga oklepata premica t in graf funkcije f.

Naloga 2 [25 + 5 točk]

Dvojiško drevo je podano z razredom `Drevo`. V vsakem vozlišču drevesa je zapisano celo število ali pa znak $+$ ali znak $*$. Drevo tako na naraven način prikazuje izračun aritmetičnega izraza s seštevanjem in množenjem.

(a) Napiši konstruktor `__init__(self, a, levo, desno)`, kjer je `a` celo število ali $+$ ali $*$, levo in desno pa `None` ali `Drevo`. Če je kombinacija vhodnih parametrov nesmiselna, vrži izjemo!

(b) Napiši metodo `__repr__(self)`, ki izpiše račun, ki ga drevo predstavlja.

(c) Dodaj metodo `izracunaj(self)`, ki vrne vrednost računa, ki ga drevo predstavlja.

(d)* Dodaj metodo `poisciRacun(self, x)`, ki v drevesu poišče (kakšno) poddrevo, katerega račun, ki ga poddrevo predstavlja, ima vrednost `x`.

(obrni list)

Naloga 3 [25 + 5 točk]

Vaška klepetulja je najsrečnejša, če novico (ki je zapisana v obliki seznama besed) brž pove neki drugi osebi. Seveda pa da novici svojo osebno noto tako, da novico malo spremeni: eno besedo odvzame ali pa doda. Naj seznam $A = [a_0, \ldots, a_{n-1}]$ predstavlja originalno sporočilo, seznam $B = [b_0, \ldots, b_{m-1}]$ pa prejeto sporočilo. Naj bo $A_i = [a_0, \ldots, a_{i-1}]$ in $B_j = [b_0, \ldots, b_{j-1}]$. Definirajmo $d(i, j)$ - minimalno število vaških klepetulj za prenos sporočila A_i v B_j. Vaški učenjak je ugotovil, da velja zveza:

$$d(i,j) = \begin{cases} i + j & \text{če } ij = 0 \\ \min\{d(i-1,j)+1,\ d(i,j-1)+1\} & \text{sicer,} \end{cases}$$

kjer izraza v funkciji $\min$ nastopata zaradi odvzemanja ali dodajanja besede.

(a) Napiši metodo `kolikoKlepetulj(a, b)`, ki vrne najmanjše število klepetulj, preko katerih se je novica `a` spremenila v novico `b`. Da bi lahko izračunal to število, moraš znotraj funkcije izračunati matriko d dimenzije $n \times m$. V komentarju v rešitvi utemelji časovno zahtevnost postopka.

(b) Napiši funkcijo `spremembe(a, b)`, ki vrne matriko `s`, ki odraža, katera sprememba je bila izbrana pri izračunu $d(i, j)$.

(c)* Napiši funkcijo `seznamSprememb(a, b)`, ki vrne (kakšen) seznam, ki prikazuje, kako se je novica `a` pri klepetih spremenila v `b`.

Naloga 4 [25 + 5 točk]

Na ogromnem listu papirja so napisana vsa cela števila med 0 in $n = 2^k$ razen enega. Želimo ugotoviti, katero število manjka. Če imamo na voljo še en ogromen list papirja, gre lažje, sicer pa se bo potrebno bolj potruditi ...

(a) Predlagaj čim enostavnejši algoritem z optimalno časovno zahtevnostjo in časovno zahtevnost ter njeno optimalnost utemelji. (Števila so zapisana v seznam `s`. Napiši funkcijo `kajManjka(s)`, ki vrne manjkajoče število.)

(b) Naj bo sedaj število n tako veliko, da podatki zasedejo skoraj ves razpoložljiv pomnilnik in si ne moreš privoščiti dodatnih tabel ali drugih podatkovnih struktur velikosti $O(n)$. Sestavi algoritem po strategiji deli in vladaj, ki najde manjkajoči element, pri čemer časovna zahtevnost algoritma ni večja od $O(n \log n)$. (Če zmoreš bolje, dobiš dodatnih 5 točk.) Manjkajoče število naj vrne funkcija `kajManjkaV(s)`. Časovno zahtevnost utemelji!

NAMIG: Zgleduj se po algoritmu za quicksort.