Računalništvo (Fizika): 2. izpit

20. junij 2014

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

Navodila: 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. Pri reševanju nalog je dovoljena uporaba interneta (brskanje po referencah). Uporaba komunikacijskih orodij (email, facebook, ipd.) je prepovedana. Strogo prepovedana je kakršna koli medsebojna komunikacija in komunikacija s tretjimi osebami na kakršen koli način (tudi preko interneta).*

Naloga 1 [25+5 točk]

Pripravi delovni zvezek v *Mathematici*, v katerem rešiš naslednjo nalogo. Podana je parametrična krivulja:

$$x = t^2 - 5t + 6,$$
$$y = t^3 - 4t.$$

(a) Nariši graf krivulje za parameter $t \in [-3, 3]$.

(b) Izračunaj vsa presečišča krivulje z osema x in y.

(c) Izračunaj tangento na krivuljo v točki $t = -1$ in jo nariši.

(d) **Dodatna naloga [+5 točk]**: Izračunaj vsoto ploščin treh omejenih površin, ki jih objema na eni strani krivulja na drugi strani pa kaka od koordinatnih osi.

Naloga 2 [25 + 5 točk]

Sestavi razred `Rotacija`, ki predstavlja rotacijo v $\mathbb{R}^n$ okoli izhodišča podano z osjo v in kotom ϕ. Vektorje bomo podajali kot sezname treh števil.

(a) Napiši konstruktor `__init__(self, v, phi)`. Podan vektor ni nujno enotski. Če je vektor, ničelen naj konstruktor vrže izjemo.

(b) Napiši metodo `os(self)`, ki vrne os rotacije kot normaliziran vektor ter metodo `inverz(self)`, ki vrne inverzno rotacijo.

(c) Napiši ustrezno metodo, ki bo omogočila, da funkcijski klic rotacije nad vektorjem vrne zavrten vektor. Če funkcijski klic uporabimo nad kakršnim koli drugim objektom, naj klic vrže izjemo. Primer uporabe:

```
>>> R = Rotacija([0,0,1], math.pi);
>>> R([1,0,0])
[-1,0,0]
```

(d) **Dodatna naloga [+5 točk]**: Sestavi generator `rotacije(R, n)`, ki za dano rotacijo R vrača $R^0, R^1, \ldots, R^{n-1}$ v tem vrstnem redu.

(Obrni!)

Naloga 3 [10 + 15 točk]

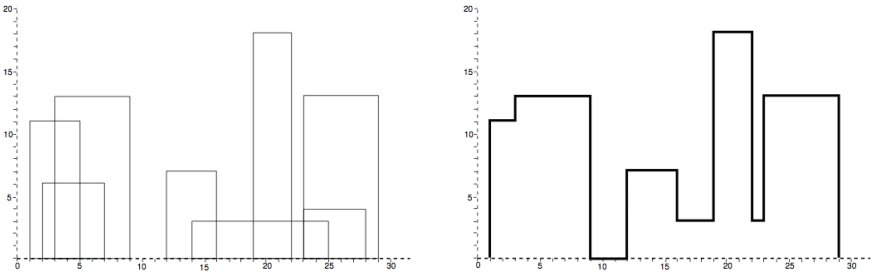
Predstavljajmo si, da je obzorje mesta opisano z realno premico. Stavbo na obzorju opišemo s trojico (l, h, r), pri čemer predstavlja vrednost l koordinato na realni premici, kjer se začne stavba, h višino stavbe in $r > l$ koordinato, kjer se zaključi stavba. Oglejmo si množico stavb:

$$[(3, 13, 9), (1, 11, 5), (12, 7, 16), (14, 3, 25), (19, 18, 22), (2, 6, 7), (23, 13, 29), (23, 4, 28)].$$

Obris stavb opišemo z zaporedjem parov, ki definira odsekoma konstantno funkcijo:

$$[(1, 11), (3, 13), (9, 0), (12, 7), (16, 3), (19, 18), (22, 3), (23, 13), (29, 0)].$$

Pri tem je seveda pomembno, da so prve komponente parov v zaporedju podane v strogo naraščajočem zaporedju! Stavbe in njihov obris vidimo grafično ponazorjene na spodnji sliki.



(a) Napiši funkcijo `zlij(ob1, ob2)`, ki zlije dva obrisa na obzorju velikosti n_1 in n_2 na čimbolj učinkovit način. Kakšna je njegova časovna zahtevnost?

(b) S pomočjo metode deli in vladaj razvij in utemelji algoritem za izračun obrisa n stavb, katerega časovna zahtevnost je največ $O(n \log n)$. Napiši funkcijo `obzorje(stavbe)`, ki implementira ta algoritem.

Opomba: nalogo oddaj v eni datoteki v Pythonu. Kodo ustrezno komentiraj in v komentarju razloži osnovne ideje algoritmov.

Naloga 3 [5 + 10 + 10 + 10 = 35 točk]

Za dano zaporedje $A = [a_1, a_2, \ldots, a_n]$ označimo z $A_{ij} = [a_i, a_{i+1}, \ldots, a_j]$ strnjeno podzaporedje. Pravimo, da je A_{ij} *strnjeno palindromsko podzaporedje* zaporedja A, če velja $a_{i+h} = a_{j-h}$ za $0 \leq h \leq j - i$. Na primer, če je $A = [a, c, c, a, b, a]$, potem sta $A_{14} = [a, c, c, a]$ in $A_{46} = [a, b, a]$ strnjeni palindromski podzaporedji.

(a) Predlagaj algoritem in napiši funkcijo `maxPalindrom(A)`, ki vrne par (i, j), ki predstavlja indeksa začetka in konca najdaljšega strnjenega palindromskega podzaporedja. Kakšna je časovna zahtevnost?

(b) Opustimo pogoj strnjenosti. Najdaljše palindromsko podzaporedje zaporedja $A = [a, c, a, c, b, a]$ je potem $[a, c, a, c, a]$. Naj P_{ij} predstavlja dolžino najdaljšega palindromskega podzaporedja v strnjenem podzaporedju A_{ij}. S pomočjo metode dinamičnega programiranja izpelji in utemelji (v komentarju datoteke v Pythonu) rekurzivno zvezo P_{ij}. (*Namig: loči primere glede na enakost/neenakot a_i in a_j.*)

(c) Implementiraj funkcijo `P(A)`, ki izračuna tabelo vrednost P_{ij}, za $1 \leq i, j \leq n$. Časovna zahtevnost naj ne presega $O(n^2)$.

(d) **Dodatna naloga [+10 točk]**: Napiši funkcijo `maxPalindrom2(A)`, ki vrne seznam, ki predstavlja eno izmed najdaljših palindromskih podzaporedij.