Podatkovne strukture in algoritmi (2024/25)

1. domača naloga – programerski del

Povezan seznam in dvojiško iskalno drevo

Povezan seznam sestoji iz zaporedja vozlišč, v katerih so shranjeni elementi (ključi). Vsako vozlišče vsebuje kazalec na drugo vozlišče. Prvemu vozlišču pravimo glava, preostanku pa rep. *Iskanje v seznamu* poteka rekurzivno z začetkom v glavi: če elementa ni v glavi, ga iščemo v repu. Element *vstavljamo* na konec seznama. Če želimo *izbrisati element*, ga najprej poiščemo, nato pa ustrezno popravimo kazalce.

Dvojiško iskalno drevo je urejeno drevo s korenom, kjer ima vsako vozlišče največ dva otroka. Elementom, ki jih hranimo v vozliščih, pravimo ključi. V urejenem drevesu za poljubno vozlišče velja, da so v levem poddrevesu vozlišča vsi elementi (ključi) manjši, v desnem poddrevesu vozlišča pa vsi elementi (ključi) večji od elementa (ključa), shranjenega v vozlišču. *Iskanje v drevesu* poteka rekurzivno od korena navzdol: če je iskani element (ključ) manjši od elementa (ključa) v trenutnem vozlišču, zavijemo v levo poddrevo, če ni, v desnega. Če je enak, vrnemo iskano vozlišče. Element *vstavljamo* v levo poddrevo, če je novi element manjši ali v desno, če je večji. Pri *brisanju elementa* najprej poiščemo vozlišče, v katerem je le-ta shranjen. Nato ločimo naslednje primere: (i) če to vozlišče nima naslednikov, ga enostavno zbrišemo; (ii) če ima določeno le eno poddrevo, vozlišče (in element) zamenjamo z njegovim edinim otrokom; (iii) če pa ima vozlišče obe poddrevesi, ga zamenjamo s tistim vozliščem v desnem podrevesu, ki hrani minimalni element (le-ta se nahaja skrajno levo v desnem poddrevesu).

Na strežniku za oddajanje nalog (https://marmoset.famnit.upr.si) je naložena prva programerska domača naloga, pri kateri je potrebno sprogramirati rekurzivni podatkovni strukturi seznam in dvojiško iskalno drevo. Sledite lahko naslednjim navodilom:

- V razredu NodeSeznam lahko dodate recimo komponento rep, ki je tipa NodeSeznam, ter dodate konstruktor. Cela števila, ki jih vstavljamo v seznam, so shranjena kot ključi v objektih tipa NodeSeznam. Za primerjanje ključev (elementov) je potrebno uporabljati metodo `compare(NodeSeznam node)`, ker se le-ta uporablja pri testiranju pravilnosti naloge. Zato je koristno (ne pa nujno) v tem razredu dodati še metode `insert(NodeSeznam node)`, `delete((NodeSeznam node)` in `search((NodeSeznam node)`, ki jih nato uporabite v razredu Seznam.

- V razredu Seznam implementirajte metode:

  (i) `insert`, ki sprejme celo število in ga vstavi v seznam. Metoda vrne true, če je bil element uspešno vstavljen in false, če element že obstaja v podatkovni strukturi;

(ii) `search`, ki sprejme celo število in poišče element v seznamu. Metoda vrne true, če je bil element uspešno najden v podatkovni strukturi, in false sicer;

(iii) `delete`, ki sprejme celo število in izbriše element iz seznama. Metoda vrne true, če je bil element uspešno izbrisan iz podatkovne strukture, in false sicer.

- V razredu NodeBinarno lahko dodate recimo komponenti levi in desni otrok, ki sta tipa NodeBinarno, ter dodate tudi konstruktor. Cela števila, ki jih vstavljamo v drevo, so shranjena kot ključi v objektih tipa NodeBinarno. Za primerjanje ključev (elementov) je potrebno uporabljati metodo `compare(NodeBinarno node)`, ker se leta uporablja pri testiranju pravilnosti naloge. Zato je koristno (ne pa nujno) dodati še `insert(NodeBinarno node)`, `delete(NodeBinarno node)` in `search(NodeBinarno node)`, ki jih nato uporabite v razredu Binarno.

- V razredu Binarno implementirajte metode:

(i) `insert`, ki sprejme celo število in ga vstavi v drevo. Metoda vrne true, če je bil element uspešno vstavljen in false, če element že obstaja v podatkovni strukturi;

(ii) `search`, ki sprejme celo število in poišče element v drevesu. Metoda vrne true, če je bil element uspešno najden v podatkovni strukturi, in false če ga ni bilo v drevesu;

(iii) `delete`, ki sprejme celo število in izbriše element iz drevesa. Metoda vrne true, če je bil element uspešno izbrisan iz podatkovne strukture, in false, če ga ni bilo v drevesu. *Če ima vozlišče tako levo kot desno poddrevo, ga zamenjajte z minimalnim elementom v strukturi, ki je že večji od brisanega elementa; torej z minimalnim elementom v desnem podrevesu.* V ta namen se posebej splača dodati še metodo findMin() v razredu NodeBinarno.

Rešitev naloge oddajte preko https://marmoset.famnit.upr.si. Vse naloge je potrebno reševati **samostojno**. Prepisovanje se kaznuje z negativnimi točkami. Rok za oddajo je **nedelja, 10. november** 2024.