NAPOJ

UČNA PRIPRAVA

Slovar

OSNOVNI PODATKI

Šola:

Letnik: 1. Letnik

Datum:

Predmet: Informatika

Učna tema: Slovar

Učna enota: podatkovna struktura slovar, iskanje podatkov, implementacija slovarja, razpršilna funkcija

Učne oblike:

- frontalno, individualno, skupinsko

Učne metode:

- razlaga, pogovor, demonstracija, reševanje problemov

Predznanje:

- poznajo pojme algoritem, funkcija, tabela

Operativni učni cilji

Ob koncu učne ure učenec:

- Spoznati problem iskanja podatkov
- Spoznati podatkovno strukturo slovar
- Implementirati slovar z razpršeno tabelo
- Spoznati razpršilno funkcijo
- Spoznati kolizijo pri razprševanju in reševanje le-te

Učna sredstva:

- Učila: list, računalnik
- Učni pripomočki: računalnik, Python, zvezek, tabla

Didaktične etape učnega procesa:

1. Pripravljanje ali uvajanje
2. Obravnava nove učne snovi ali usvajanje
3. Urjenje
4. Ponavljanje
5. Preverjanje

Medpredmetne povezave: slovenščina, angleščina

Literatura:

- M. T. Goodrich, R. Tamassia, Data structures and algorithms in Java, 2001.
- M. A. Weiss, Data Structures and Algorithm Analysis (2nd Edition), Addison Wesley, 1994.
- B. Vocking, H. Alt, M. Dietzfelbinger, R. Reischuk, C. Scheideler, H. Vollmer, D. Wagner, Algorithms unplugged, Springer Science & Business Media, 2010.
- J. Hamer, Hashing revisited, in: ACM SIGCSE Bulletin, Vol. 34, ACM, 2002, pp. 80–83.
- M. Singh, D. Garg, Choosing best hashing strategies and hash functions, in: Advance Computing Conference, 2009. IACC 2009. IEEE International, IEEE, 2009, pp. 50–55.
- G. Anželj, J. Brank, A. Brodnik, P. Bulić, M. Ciglarič, M. Đukić, L. Furst, M. Kikelj, A. Krapež, H. Medvešek, N. Mori, M. Pančur, P. Sterle, Računalništvo in informatika 1, Založba Univerze na

Primorskem and Založba Fakultete za računalništvo in informatiko and Založba Fakultete za elektrotehniko and računalništvo in informatiko, 2015. URL https://lusy.fri.uni-lj.si/ucbenik/book/index.html.
Novi pojmi: - Slovar, razprševanje, razprševalna funkcija, razpršilna tabela, kolizija.
Priloga: - tabela za kolizije - rešitev Zaprto razprševanje - delovni list - delovni list - rešen

POTEK UČNE URE

UVODNI DEL: UVAJANJE

ČAS	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI
Uvod 5 min	*Pozdravi učence, zapiše manjkajoče učence in po potrebi prosi dežurnega učenca, da pobriše tablo.*	Odzdravijo, naštejejo manjkajoče učence, dežurni učenec pobriše tablo.	Frontalno.
Motivacija 5 min	Danes bomo spoznali problem iskanja oziroma hranjenja podatkov in hitrosti dostopa do njih. Do sedaj že vsi poznate tabelo, kjer je dostop do elementov zelo hiter. Problem je v tem, da do njih dostopamo preko indeksov – 0, 1, 2, itd.. Velikokrat v življenju pa imamo drug podatek, imenovan ključ, s katerim dostopamo do elementov. Primer: v zdravstvenem domu ima zdravnik kartoteko vsakega svojega pacienta. Ali je bolje, da do njih dostopal preko indeksov (prvi pacient, drugi pacient, …) ali preko ključa imena in priimka? Mi lahko naštejete še nekaj primerov, kjer potrebujemo za dostop to podatkov nek ključ? • Telefonski imenik. • Kazalo v knjigi. • EMŠO.	Poslušajo. Odgovorijo, naštejejo primere.	Frontalno, razlaga, pogovor.

GLAVNI DEL: OBRAVNAVANJE UČNE SNOVI

VSEBINSKI POUDARKI	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI
Slovar 10 min	Imamo službo v banki kot programer. Dobili smo programersko nalogo, da v bazo shranimo številke bančnih računov ter ime in priimek vseh strank. Pri tem moramo imeti v mislih, da bomo lahko preko številke bančnega računa čim hitreje dostopali do osebnih podatkov (primer: na bančnem računu XXXXX zmanjšaj stanje za 20€). Katero podatkovno strukturo bomo uporabili? • Tabela: je hitra za iskanje (O(1)), a lahko dostopamo zgolj preko indeksov (0, 1, 2,...do velikost tabele). • Povezan seznam: iskanje je O(n), kar ni dovolj hitro. • Drevo: časovna zahtevnost iskanja je O(log(n)), kar ni slabo, a bi lahko bilo bolje. Katero podatkovno strukturo bi vi uporabili? Vidimo, da je za iskanje res najbolj primerna uporaba tabele, le njeno slabost – to, da dostopamo do elementov le preko indeksov – moramo nekako odstraniti. Zato spoznajmo podatkovno strukturo, ki se imenuje slovar. Slovar je kot tabela, le da vsak element vsebuje ključ in vrednost. Tako kot je pri tabeli ključ za iskanje elementov indeks elementa, imamo sedaj za ključ poljuben podatek. Slovar torej vzdržuje množico parov oblike ⟨ključ, vrednost⟩, pri čemer so vsi ključi različni. Ključi so lahko števila, datumi, nizi, ... karkoli. Rešimo torej našo nalogo iz službe. Naši vnosi v slovarju bodo seveda bili (bančni račun, ime + priimek). Poglejmo si rešitev v Pythonu, ki že vsebuje podatkovno strukturo slovar. Nov slovar ustvarimo z: `racuni = {}`	Spremljajo razlago in si zapisujejo v zvezek. Odgovorijo: uporabili bi tabelo.	Frontalno, razlaga, pogovor.

	Nov podatek dodamo z:
	
	`racuni["4987082554356391"] = "Ana Akonto"`
	`racuni["1936534552807894"] = "Betka Betic"`
	
	Če sedaj izpišemo vsebino slovarja, dobimo:
	
	`print(racuni)`
	
	Vrednost ključa dobimo z:
	
	`print(racuni.get("1936534552807894"))`
	
	Če ključ že obstaja, uporabimo:
	
	`if "5534250889479163" in racuni:`
	`  del racuni["5534250889479163"]`
	
	V zgornjem primeru smo tudi izbrisali vnos, z uporabo ukaza del.
	
	Preko ključev se lahko tudi sprehodimo:
	
	`for kljuc in racuni:`
	`  print("kljuc: " + str(kljuc) + ", vrednost: " + str(racuni[kljuc]))`		

razprševanje
 	
15 min
	Slovar bomo sedajo sami implementirali, in sicer z uporabo tabele.
	Učitelj napiše spodnjo kodo na svoj računalnik.
	
	`tabRacunov = []`
	`tabRacunov[4987082554356391] = "Ana Akonto"`
	`tabRacunov[1936534552807894] = "Betka Betic"`
	
	Kaj je problem s to rešitvijo?
	
	Tako, saj mora vsebovati vsaj 4987082554356391 celic.
	Ali bi lahko ta ključ nekako pretvorili v manjšega?
	
	Seveda, so rekle bistre glave, in si izmislile podatkovno strukturo imenovano razpršena tabela. To je tabela, kjer uporabimo razprševalno funkcijo, ki vsak ključ pretvori v indeks omejene velikosti. Pripravimo tabelo velikosti n in si določimo razpršilno funkcijo h(), ki bo ključe preslikala v indeks med 0 in n. Ko bomo hoteli dodati nov element, bomo ključ s pomočjo funkcije preslikali v indeks in ga shranili v njega. Pa kar napišimo v
		Sledijo razlagi in si zapisujejo.
		
		Odgovorijo: problem je velikost tabele.
			Frontalno, razlaga, pogovor.

	Pythonu.
	
	`def ustvari(velikost):`
	`  return [None]*velikost`
	
	`def dodaj(slovar, kljuc, vrednost):`
	`  indeks = h(slovar, kljuc)`
	`  slovar[indeks] = vrednost`
	
	`def h(slovar, kljuc):`
	`  indeks = kljuc % len(slovar)`
	`  return indeks`
	
	`racuni = ustvari(107)`
	`dodaj(racuni, 4987082554356391, "Ana Akonto")`
	
	Za pretvorbo v indeks smo uporabili kar matematično operacijo modul, to je, koliko je ostanek pri deljenju vrednosti ključa in dolžine slovarja. S tem smo zagotovili, da bo indeks v rangu 0 in velikosti slovarja. Mi lahko poveste kolikšna je časovna zahtevnost dodajanja in iskanja elementa?
	
	Če želimo poiskati ime in priimek bančnega računa, moramo le pretvoriti številko bančnega računa v indeks ter preveriti, ali ga tabela vsebuje.
	
	`def pridobi(slovar, kljuc):`
	`  indeks = h(slovar, kljuc)`
	`  return slovar[indeks]`
	
	Tako nam:
	
	`ime = pridobi(racuni, 4987082554356391)`
	`print(ime)`
	
	izpiše »Ana Akonto«.
	
	Če imamo za ključe niz, in ne številke, lahko vsako črko niza pretvorimo v število iz tega pa potem izračunamo indeks. En primer razpršilne funkcije je lahko:
	
	`def h(slovar, kljuc):`
	`  k = 0`
	`  for crka in kljuc:`
	`  k += ord(crka)`
	`  indeks = k % len(slovar)`
	`  return indeks`
		Odgovorijo: časovna zahtevnost je O(1), saj za element zgolj izračunamo indeks ter ga dodamo v tabelo..	

Kolizije 10 minut	Rešimo sedaj naslednjo nalogo: Imamo razpršilno tabelo s 6 celicami in razpršilno funkcijo h(x) = x mod 10. V njo postopoma dodajmo števila 4371, 11323, 4199, 4344, 1989 ter posodabljajmo razpršilno tabelo. *Učitelj na tablo nariše tabelo (v prilogi 1) in z učenci reši nalogo.* Kaj se zgodi, ko želimo dodati število 1989? Moramo jo dodati na 3. indeks, ki pa že vsebuje število 4371. Temu pravimo kolizija. V praksi se nam to zgodi, če razpršilna funkcija ni dovolj dobra in želi 2 različna elementa zapisati v isto celico v tabeli. Na voljo imamo več rešitev, pri čemer bomo danes obravnavali dve.	Rešijo nalogo.	Frontalno, razlaga, pogovor, skupinsko delo, razlaga, reševanje problemov.

Zaprto razprševanje
 	
15 minut
	Prva rešitev je zaprto razprševanje. To pomeni, da, ko naletimo na kolizijo, element, ki ga želimo dodati, preprosto dodamo na naslednjo prazno mesto v tabeli. Na kateri indeks se bo sedaj zapisalo število 1989?
	
	Psevdokoda za dodajanje sedaj izgleda takole:
	
	`def dodaj(slovar, kljuc, vrednost):`
	`  indeks = h(slovar, kljuc)`
	`  če je slovar na indeksu prazen:`
	`  dodaj vrednost na ta indeks`
	`  drugače:`
	`  indeks += 1`
	`  dokler slovar[indeks mod len(slovar)] ni prazen:`
	`  indeks += 1`
	`  dodaj vrednost na ta indeks`
	
	Naloga: za zgornjo psevdokodo napišite Python kodo ter preverite, če je rešitev pravilna.
	
	Rešitev je v prilogi 2.
		Si zapisujejo.
		
		Odgovorijo: na indeks 4.
		
		Rešijo nalogo.
			Frontalno, razlaga, pogovor, individualno delo, razlaga, reševanje problemov.

Odprto razprševanje
 	
10 minut
	Druga rešitev kolizij je odprto razprševanje. Tu v celicah tabele ne hranimo zgolj enega para (ključ, vrednost), ampak zaporedje parov, ki se zapišejo na ta indeks. Vsaka celica v tabeli torej vsebuje zaporedje. Psevdokoda za dodajanje izgleda takole:
	
	`def dodaj(slovar, kljuc, vrednost):`
	`  indeks = h(slovar, kljuc)`
	`  pripni vrednost na konec seznama na tem indeksu tabele`
	
	Če sedaj želimo poiskati vrednost nekega ključa, moramo najprej najti indeks, nato pa se sprehodimo po seznamu na tem indeksu v tabeli.
		Poslušajo, si zapisujejo.	Frontalno, razlaga.

ZAKLJUČNI DEL: ZAKLJUČNO PONAVLJANJE / PREVERJANJE

ČAS	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI
15 min	Za konec vas prosim, da rešite delovni list (*priloga 3*), na koncu pa bomo pregledali vaše odgovore.	Poslušajo, rešijo delovni list, pregledajo rešitve.	Frontalno, individualno, razlaga, reševanje problemov, pogovor.
5 min	*Se zahvali učencem za sodelovanje, jim da napotke za domačo nalogo. Dežurni učenec naj po potrebi pobriše tablo. Učenci naj ugasnejo računalnike.*	Poslušajo, ugasnejo računalnike, dežurni učenec pobriše tablo.	Frontalno.

Priloge

Priloga 1: tabela za kolizije

Dodaj:	tabela				
4371					
11323					
4199					
4344					
1989					

Priloga 2: zaprto razprševanje

```
def dodaj(slovar, kljuc, vrednost):
    indeks = h(slovar, kljuc)
    if slovar[indeks] is None:
        slovar[indeks] = vrednost
    else:
        indeks += 1
        while slovar[indeks % len(slovar)] != None:
            indeks += 1
        slovar[indeks] = vrednost
```

Priloga 3: delovni list

Delovni list – slovar

1. V čem sta si podobna slovar in tabela?

2. Kaj je razpršilna funkcija? Kako se dodajajo elementi v razpršilno tabelo?

3. Kaj je kolizija? Opišite oba načina reševanja kolizije.

4. V eni od nalog smo opazili, da je razpršilna funkcija hotela zapisati 2 vrednosti v isti indeks. Zaradi tega smo rekli, da je ta funkcija slaba. Kako bi torej definirali, kaj je to dobra razpršilna funkcija?

5. Sprogramirajte funkcijo v Pythonu, ki prejme slovar in izpiše vsebino v obliki: »[ključ] : [vrednost]«.

Priloga 3: delovni list - rešen

Delovni list – slovar

1. V čem sta si podobna slovar in tabela?

Oba delujeta po sistemu ključ:vrednost, le da lahko imamo v slovarju za ključ kakršenkoli podatek, v tabeli pa imamo le indekse – števila.

2. Kaj je razpršilna funkcija? Kako se dodajajo elementi v razpršilno tabelo?

Razpršilna funkcija preslika ključe v vrednost med 0 in velikostjo slovarja.

3. Kaj je kolizija? Opišite oba načina reševanja kolizije.

Kolizija se zgodi, ko skušamo dodati dva vnosa na isto mesto v slovarju. Rešujemo ga z odprtim (zapišemo v naslednjo prosto celico) in zaprto razprševanje (v celicah hranimo povezane sezname).

4. V eni od nalog smo opazili, da je razpršilna funkcija hotela zapisati 2 vrednosti v isti indeks. Zaradi tega smo rekli, da je ta funkcija slaba. Kako bi torej definirali, kaj je to dobra razpršilna funkcija?

Dobra razpršilna funkcija ne zapiše 2 različnih vnosov na isto mesto v slovarju.

5. Sprogramirajte funkcijo v Pythonu, ki prejme slovar in izpiše vsebino v obliki: »[ključ] : [vrednost]«.

```
def izpisi(slovar):
    for i in slovar:
        print(i, ":", slovar[i])
```