Fakultet elektrotehnike i računarstva
Zavod za primjenjeno računarstvo

Napredni algoritmi i strukture podataka

2. laboratorijska vježba

Beli_cigan XD

Zagreb, 12.12.2012.

1. Zadatak

Napisati program koji će dinamičkom strategijom (dinamičkim programiranjem) riješiti problem u kojem svaka kategorija ima konačni broj podkategorija, a može se uzeti samo po jedan član tih podskupova. U primjeru s predavanja, to bi značilo da npr. postoje iste stvari od različitih materijala pa su im različite i vrijednosti i težine, a lopov može (želi) uzeti najviše jednu stvar „svake vrste“, pri čemu je ograničenje ukupna težina, a ne volumen.

- za kolokviranje vježbe važno je pregledno i jasno opisati problem i njegove karakteristike koje ga čine prikladnim za primjenu dinamičkog programiranja
- naglasak rada treba biti na kvalitetnom programu pa u dokumentaciji treba jasno (ali sažeto!) prikazati ideju rješenja i organizaciju programa (strukture podataka, pomoćne funkcije, itd.). Programski kod ne upisivati u dokumentaciju, osim kratkih izvadaka ako se radi o posebno važnim blokovima.

2. Rješenje zadatka

2.1. Teorijski uvod

Zbog općenitosti zadatka i lakšeg razumijevanja samog problema odabran je konkretan problem iz modernog svijeta čije se programsko rješenje može primijeniti na ostale probleme koji se rješavaju dinamičkim programiranjem.

Recimo da gospodin Zdravko Mamić zaposli novog trenera u GNK Dinamo i daje mu maksimalni budžet od 20 milijuna kuna. Gospodin Mamić od tog trenera traži da na raspolaganju za dovesti ima 4 vrste igrača (golman, obrambeni, vezni i napadač). Također mu je zadano da od svake vrste igrača može dovesti samo jednoga. Za svaku od vrsta igrača poznata su po 3 igrača koja se nalaze na popisu želja (njihova imena sad nisu bitna, ona su navedena u kodu programa). Svaki od tih igrača ima svoju cijenu izraženu u milijunima kuna (zaokruženo na cijele milijune da se olakša zadatak), a također ima i pridruženu ocjenu iz intervala od 1 do 20 koja prikazuje igračevu sposobnost na njegovoj pozicji. Problem je kako olakšati treneru da izabere najbolju kombinaciju igrača uz maksimalni zbroj ocjena, a da im je cijena <= 20 milijuna kuna.

2.2. Implementacija

Program je pisan u jeziku Python v2.7.3 i može se razlučiti na 4 velike cjeline. Ukratko problem je riješen uporabom dinamičkog programiranja, odnosno pomoću dvije uparene tablice. Obje tablice su dimenzija 5 x 20, a popunjuju se zajedno paralelno. Prva tablica ima 5 redova (0...4) koji označavaju igrača, a stupci (1...20) označuju cijene u milijunima. Detaljnije o tablicama i kako se popunjavaju u poglavlju 2.2.3. Kada se tablice popune iz njih se iščitava rješenje, odnosno maksimalni zbroj ocjena koji može biti postignut, a da je zbroj cijena igrača <= 20. Također se ispisuju imena igrača koji daju taj maksimalni zbroj ocjena. Ako postoji više kombinacija rješenja koja daju isti maksimalni zbroj ocjena, sva takva se ispisuju, neovisno o tome kolki je broj igrača uvršten u jednu kombinaciju rješenja.

2.2.1. Main

U glavnom dijelu programa prvo je potrebno instancirati globalnu varijablu **maxOcjene** koja se koristi za usporedbu između pojedinih zbrojeva ocjena dobivenih različitim kombinacijama igrača. Nakon toga u program se unose podaci o igračima pomoću klase Igrac (detaljno u poglavlju 2.2.2). Potom je potrebno igrače razvrstati po vrsti igrača (golmani, obrambeni, vezni i napadaci) tako da se generiraju **4 liste** s referencama na igrače. Kada se liste stvore kroz svaku od njih se iterira pomoću četverostruke for petlje:

```
for golman in golmani:
        for obrana in obrambeni:
                for veza in vezni:
                        for napad in napadaci:
```

Time dobivamo sve moguće kombinacije igrača koje možemo izabrati prema zadatku. Sad dolazimo do srca programa i instanciranja objekta klase Dinamicko (poglavlje 2.2.3) koji u sebi sadrženi dva globalno važna podataka (`maxOcjene` i `rezultatIgraci`). Kod prve kombinacije igrača, globalna varijabla **maxOcjene** postavlja se na povratnu vrijednost `dinamicko.maxOcjene`, a u globalnu listu **rezultat** spremiti imena igrača. Kod svake iduće kombinacije potrebno je provjeriti je li nova povratna vrijednost veća od trenutne. Ako je, onda trenutnu vrijednost **maxOcjene** zamijeniti s novom, a u **rezultat**-u izbrisati sve stare zapise i zapisati nove. Ako se pojavi nova kombinacija koja daje jednak zbroj ocjena kao trenutni, jednostavno u **rezultat** dodati novu kombinaciju igrača. Na kraju je samo potrebno uljepšati spremljene podatke i obaviti pregledan ispis.

2.2.2. Klasa Igrac

Klasa Igrac služi za lakše i bolje upravljanje unesenim podacima o igračima. Kad se instancira objekt klase Igrac poziva se __init__ metoda (zapravo konstruktor) koja prima ulazne parametre (imePrezime, cijena i ocjena) i sprema ih kao jedan objekt. Tu je također i metoda __str__ koja služi za ispis podataka o igraču.

2.2.3. Klasa Dinamicko

Klasa Dinamicko je zapravo glavni dio programa i u njoj je prikazana uporaba korištenja dinamičkog programiranja. Sastoji se samo od metode __init__ (konstruktora) koja se automatski poziva kod instanciranja objekta, a kao argumente prima 5 referenci na objekte klase Igrac. Prvi objekt je uvijek objekt BB koji služi za lakšu implementaciju i nema nikakvo funkcionalno značenje. Ostale reference primaju se pomoću 4 FOR petlje (2.2.1.), gdje prva uvijek daje golmana, druga obrambenog igrača, treća veznog i zadnja napadača. Kod svakog poziva potrebno je incijalizirati početne podatke na zadane vrijednosti. Vrijednost **zbrojOcjena** predstavlja maksimalni zbroj ocjena za odabranu kombinaciju, a da je zbroj cijena<=20). Vrijednosti **igrac** i **budzet** predstavljaju broj igrača izmedu kojih se bira i maksimalnu ukupnu cijenu koja se može platiti za igrače. U vrijednosti **rezultat** i **rezultatIgraci** spremaju se kombinacije igrača koje daju rješenje za zadane ulazne parametre. Prije početka izgradnje tablica potrebno je napraviti listu s igračima radi lakše gradnje tablica. **poljeOcjena** i **polje01** su dvodimenzionalna polja dimenzija 5 x 20 i inicijalno su sve vrijednosti postavljene na 0.

2.2.3.1 poljeOcjena (prva tablica, tablica ocjena)

Prvi redak tablice(index [0,y]) puni se 0-ma jer to označuje kakve su ocjene ako uzmemo 0 igrača u izboru igrača. Drugi redak (index [1,y]) puni se tako da se uzme golman i gleda se njegova cijena. Svi stupci čiji je index manji od zadane cijene igrača pune se s 0, a ona gdje je cijena veća ili jednaka indexu stupca popuni se s ocjenom igrača. Treći red popunjava se isto kao i drugi, ali s dva dodatna uvijeta. Prvi od njih je da ukoliko je zbroj cijena dva igrača koja su se dosad gledala manji ili jedank od indexa stupca (odnosno y), onda se na poziciju [2,y] sprema zbroj ocjena tih igrača. Drugi uvijet je da ako vrijedi poljeOcjena[2,y] < poljeOcjena[1,y], onda se u poljeOcjena ne zapisuje cijena novog igrača, nego vrijednost iz poljeOcjena[1,y]. Četvrti i peti red popunjavaju se na isti način kao i drugi, samo što ovdje ima više kombinacija zbrojeva jer u četvrtom redu ima 3 igrača(zbrojevi su: 3+2+1, 3+2, 3+1

3+1),a zapisuje se najveći ako je za njega zadovoljeno da je zbroj cijena manji od y. Zbroj 2+1 ne treba gledati jer će biti pokriven uvijetom poljeOcjena[3,y] < poljeOcjena[2,y], a tu se vidi prednost dinamičkog programiranja. U zadnjem redu (petom) ima 7 zbrojeva koje treba provjeriti (4+3+2+1, 4+3+2, 4+3+1, 4+2+1, 4+3, 4+2, 4+1).

2.2.3.2 polje01 (druga tablica, tablica 0 i 1)

Ova tablica iste je dimenzije kao i prva tablica, a puni se na temelju vrijednosti zapisanih u prvoj. Svako polje može poprimiti vrijednost ili 0 ili 1. Prvi red popuni se s 0-ma po uzoru na prvu tablicu. Drugi red puni se s 1-icima tamo gdje je zadovoljen uvijet (poljeOcjena[x-1][y] >= poljeOcjena[x][y]). To znači da ako je vrijednost iz prethodnog reda za isti stupac bila veća ili jednaka, ta ista vrijednost zapisuje se i u trenutni red.

2.2.3.3 while petlja (čitanje rješenja)

Nakon što su obje tablice popunjene vrijednostima treba pročitati rješenje za danu kombinaciju igrača. Prvo se za maksimalnu vrijednost broja ocjena uzima vrijednost iz zadnjeg reda i stupca tablica jer je on zasigurno najveći. To se može provjeriti uspoređivanjme svih članova tablice, no nema potrebe jer je to i matematički zadovoljeno iz dva uvjeta. Prvi je da je desni član reda uvijek veći ili jednak od prethodnog zato jer ako se igrač može kupiti pomoću **Y** milijuna kuna, onda se može kupiti i za **Y+1** milijuna kuna, što zbroj ocjena ostavlja istim. A ako se **Y** toliko poveća da se može kupiti i dodatni igrač, to rezultira sigurno većim zbrojem ocjena. Drugi uvijet je onaj koji se koristi kod samog popunjavanja tablice: ako vrijedi poljeOcjena[2,y] < poljeOcjena[1,y], onda se u poljeOcjena ne zapisuje cijena novog igrača, nego vrijednost iz poljeOcjena[1,y].

Sada treba saznati koji to igrači daju maksimalni zbroj ocjena za ukupnu cijenu manju od zadanog budžeta. To se radi pomoću 2 varijable: igrac i budzet. Igrac se postavi na 4, a budzet na 20 kao što je zadano u zadatku. Krenemo od kraja tablice 0 i 1-ica. Gledamo zadnji stupac i tražimo prvu 1-icu krenuvši odozdo.Kad je pronađemo gledamo do kojeg smo reda došli i uzimamo igrača s istim rednim brojem iz ulazne liste igrača i stavljamo ga u listu s rješenjima. Nakon toga smanjujemo budzet za cijenu igrača kojeg smo uzeli i gledamo stupac koji je jednak dobivenoj razlici. Ponavljamo opet isti postupka sve dok vrijedi (igrac > 0 && budzet > 0).

2.2.4. Klase Zbroj

Tri klase Zbroj (Zbroj2, Zbroj3 i Zbroj4) napravljne su kao 3 klase jer Python ne podržava da jedna klasa ima više implementiranih konstruktora. Broj u nazivu klase označava broj pribrojnika koje __init__ prima i to je jedina razlika između klasa. Ove klase su potrebne da možemo pamtiti par vrijednosti (zbrojOcjena i zbrojCijena) korištenih u klasi Dinamicko.

3. Zaključak

Dinamičko programiranje uvelike štedi vrijeme i resurse potrebene za izračun nekih specifičnih zadataka. Glavna prednost je obrađivanje problema bottom-up načinom pristupa, gdje se jednom izračunata jednostavnija podrješenja koriste kao alat za daljnje rješavanje kompletnog problema. Ovaj program lako je proširljiv na veći broj zadanih igrača, a dodavanje novih vrsta igrača također ne predstavlja probleme. No dodavanjem nove vrste igrača, raste i složenost. Stoga trebalo bi potražiti bolji i brži način kako rješiti problem četverostruke FOR petlje, te smanjiti složenost. No pošto ovaj konkretan problem ne bi realno trebao prelaziti preko 100 igrača, ovaj program brzo računa i zadovoljava potrebe korisnika.
Također želim napomenuti da je očito da nisam radio po primjeru s predavanja niti po postupku kojim je tamo obrađeno dinamičko programiranje, nego meni razumljivijim načinom navedenim u literaturi.

4. Literatura

1. YouTube: http://www.youtube.com/watch?v=EH6h7WA7sDw