UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

Uvod u računarske algoritme

Dinamičko programiranje i memoizacija

Zadaća 4

Tuzla, maj/svibanj 2018.

Franjevačka 2, 75000 Tuzla, Bosna i Hercegovina
Web: http://fet.ba/

Sadržaj

Napomena

U svim problemima koji slijede nije dozvoljena upotreba komandi i funkcija koje dosad nisu korištene na predavanjima ili vježbama. Dozvoljena je upotreba kontejnera iz standardne biblioteke (`std::vector` i `std::list`), kao i C nizova.

Zadatak 1:

Potrebno je napisati funkciju koja će za proslijeđeni cijeli broj `n` vratiti broj različitih načina na koje možemo dobiti broj `n` kao zbir 1, 3 i 4. Rješenje implementirati rekurzivno bez memoizacije, sa memoizacijom, te nerekurzivno primjenom principa dinamičkog programiranja.

Naprimjer, za n = 5 imamo 6 različitih kombinacija brojeva 1, 3 i 4 čijim sabiranjem možemo dobiti 5:

```
5  = 1 + 1 + 1 + 1 + 1
   = 1 + 1 + 3
   = 1 + 3 + 1
   = 3 + 1 + 1
   = 1 + 4
   = 4 + 1
```

Zadatak 2:

Neka je sekvenca cijelih brojeva definisana na sljedeći način:

```
a(0) = 2
a(1) = 4
a(2) = 12
a(n) = 3*a(n-3)+2*a(n-2)+a(n-3)
```

Potrebno je napisati funkciju koja će za proslijeđeni cijeli broj n vratit `n`-ti broj u prethodno definisanoj sekvenci. Rješenje implementirati rekurzivno bez memoizacije, sa memoizacijom, te nerekurzivno primjenom principa dinamičkog programiranja.

Zadatak 3:

Implementirati algoritam koji će za zadanu sekvencu cijelih brojeva pronaći dužinu njene najduže rastuće podsekvence. Na primjer, za sekvencu {9, 21, 8, 32, 20, 40, 59} treba da vrati broj 5 (jer je najduža rastuća podsekvenca date sekvence {9, 21, 32, 49, 59}). Implementirati rekurzivno rješenje bez memoizacije, sa memoizacijom, te nerekurzivno rješenje primjenom principa dinamičkog programiranja. Dodatno, napisati funkciju koja će na osnovu orginalne sekvence i *lookup* tabele rekonstruisati i ispisati najdužu rastuću podsekvencu.

Zadatak 4:

Dječak je došao u zabavni park sa ukupno 10KM i želi potrošiti što više vremena na različitim vožnjama. Pošto svaka vožnja ima različitu cijenu i dužinu trajanja, dječak je odlučio napraviti efikasan algoritam (polinomske vremenske složenosti) koji će mu u što kraćem roku dati optimalno rješenje sa stanovišta utrošenog vremena u skladu sa njegovim budžetom. Zadatak je implementirati takav algoritam uzimajući u obzir da dječak nije htio ni na jednu vožnju ići više od jednom.

tip vožnje	**cijena**	**trajanje (min)**
1	10	40
2	5	18
3	4	35
4	2	2
5	3	4
6	1	10

Algoritam treba da vrati ukupno trajanje svih odabranih vožnji. Analizirati implementirani algoritam u smislu složenosti i memorijskih zahtjeva i uporediti takvo rješenje sa drugim mogućim rješenjima. Da li postoji više rješenja koja daju isti rezultat, ali uz manju efikasnost? Ako da, opisati jedno od takvih rješenja (bez implementacije). Također, potrebno je osmisliti i implementirati funkcionalnost za ispis tipova vožnji koje je najbolje odabrati da bi ukupno trajanje bilo maksimalno.