LABORATORIJSKE VJEŽBE IZ ALGORITAMA I STRUKTURA PODATAKA

Vježba 3

Rekurzija

Rekurzija je važan koncept u matematici i računarstvu. Mnogi algoritmi najbolje se opisuju pomoću rekurzije. Rekurzivni program najjednostavnije se definira kao program koji poziva sam sebe (rekurzivna funkcija kao funkcija koja poziva samu sebe). Da se program ne bi izvodio beskonačno dugo, u rekurziji je neophodno definirati tzv. **osnovni slučaj** za koji program daje direktno rješenje. U svakoj rekurziji trebamo imati barem jedan osnovni slučaj da bi rekurzija bila ispravno definirana.

Mnogi problemi rješavaju se korištenjem iteracije ili rekurzije. Općenito su rekurzivna rješenja manje učinkovita od iterativnih obzirom na trajanje računanja. Međutim, u mnogim slučajevima rekurzija daje prirodno i jednostavno rješenje za probleme velike složenosti.

Problemi koji se mogu rješavati rekurzivno trebaju imati slijedeće karakteristike :

- problem treba biti takav da se može redefinirati pomoću jednog ili više pod-problema koji su po prirodi identični ali na neki način manji po veličini
- jedno ili više osnovnih slučajeva problema ima direktno rješenje
- postupkom redefinicije na manje potprobleme, problem se u konačnici svodi na osnovne slučajeve

Algoritam koji slijedi ilustrira opći slučaj rekurzije.

```
if(to je osnovni slučaj) then
   riješi ga direktno
else if (to nije osnovni slučaj) then
   redefiniraj problem korištenjem rekurzije
end if
```

Prije konstrukcije rekurzivnog rješenje potrebno je odgovoriti na tri pitanja :

1) Kako se problem može redefinirati pomoću jednog ili više manjih problema istog tipa ?
2) Koji slučajevi problema mogu biti osnovni slučajevi ?
3) Da li će se doći do osnovnih slučajeva kako se veličina problema smanjuje ?

Primjer : Funkcija faktorijel

n-ti faktorijel računa se kao :

$$0! = 1$$

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-2)(n-1)n$$

a definiran je za sve pozitivne cijele brojeve. Ta definicija se može napisati i na slijedeći način :

$$n! = 1 \quad ; \text{za n=0}$$

$$n! = n \cdot (n-1)! \quad ; \text{za n>0}$$

a ta definicija je rekurzivna. Na temelje te definicije jednostavno se može izvesti funkcija za izračunavanje faktorijela u programskom jeziku C :

```
int fakt(int n) {
   if (n<=1) {
      return 1;
   } else {
      return n*fakt(n-1);
   }
}
```

PRIPREMA ZA VJEŽBU :

Neka su *n* i *m* nenegativni cijeli brojevi i pri tome je *n* veće ili jednako od *m*. "*n* povrh *m*" se može izračunati preko slijedeće matematičke relacije:

$$\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}; \quad \binom{k}{k} = 1, \quad \binom{1}{k} = 1, \quad \binom{k}{0} = 1$$

Ovdje vidimo i tri osnovna slučaja kada je rezultat jednak 1.

VJEŽBA:

Zadatak 1. Izračunati rezultat `povrh(n, m)` tako da riješimo problem kao rekurzivnu funkciju, osim toga, potrebno je riješiti isti problem i pomoću stoga kako bi usporedili dobivene rezultate. Dobivene rezultate napisati u predviđenu tablicu, a rješavati ćemo uvijek problem gdje je `m=n/2`.

Zadatak 2. (za veću ocjenu) Potrebno je stog realizirati pomoću povezanog popisa te također izvršiti računanje povrh. Usporediti brzinu rada ovakvog stoga s onime iz zadatka 1.

Prezime i ime: _____

Broj indeksa: _____

Zadatak 1.

A) Koja su ograničenja prilikom računanja operacije povrh koristeći faktorijele?

_____

_____

B) Popuniti tablicu tako da napravite i pokrenete program:

n	**Rezultat n povrh n/2**	**Vrijeme rekurzije [ms]**	**Vrijeme sa stogom [ms]**	**Vrijeme kad je stog izveden kao PP (za veću ocjenu) [ms]**

C) Skicirajte vrijeme izvršavanja u odnosu na broj n:

D) Napišite vlastiti komentar

_____

_____

_____