Algoritmi i strukture podataka

ak. god. 2012./13.

Zadatci za 2. laboratorijsku vježbu

Za vrijeme laboratorijskih vježbi korištenje računala dopušteno je **isključivo** u svrhu rada na vježbi. Svako nenamjensko korištenje (surfanje, *chat* i slično) povlači za sobom **udaljavanje s vježbi** bez mogućnosti naknadnog stjecanja bodova.

Vježbe se ocjenjuju samo jednom, što znači da ponovno pristupanje vježbama ne može niti uvećati niti umanjiti prvu ocjenu, dakle naknadno stečeni bodovi nakon dobivanja prve ocjene ne ostaju u evidenciji. To ograničenje uvedeno je radi sprječavanja neodgovornih studenata da višestrukim pristupanjima vježbama nepotrebno opterećuju nastavno osoblje, ali nema prepreka da studenti željni dodatnih vježbi i rada s demonstratorima dođu i više puta, samo je poželjno da se pritom o svemu dogovore s odabranim demonstratorom.

Laboratorijske vježbe traju 2 školska sata. Počinju na puni sat (10:00, 12:00, 14:00...).

Skrećemo pozornost da uloga demonstratora nije rješavanje zadataka umjesto studenata, nego pomoć studentima u razvoju vlastitih rješenja i ocjenjivanje. Prema tome, ne tražite od demonstratora da Vam ponovno objašnjavaju gradivo ispočetka i uče Vas nečemu što ste trebali usvojiti u drugim oblicima nastave, ali slobodno im se obratite za pomoć s infrastrukturnim problemima ili nejasnoćama u zadatcima.

Na laboratorijske vježbe dođite **pripremljeni. Proučite do sada obrađeno gradivo i programe koji su Vam na raspolaganju u repozitoriju predmeta**. Dijelove tih programa **slobodno koristite** u svojim rješenjima zadataka za laboratorijsku vježbu.

Ova vježba sastoji se od nekoliko zadataka, pri čemu svaki donosi određeni broj bodova, a teme su: **sortiranje**, **rekurzija** i **složenost algoritama.** Primjetit ćete da je ukupni zbroj bodova veći od 2.5, koliko najviše donosi ova laboratorijska vježba. Time se studentima ostavlja mogućnost izbora zadataka prema vlastitim sklonostima i ambicijama.

Zadatci označeni zvjezdicom (*) su nešto zahtjevniji i namijenjeni ponajviše naprednijim studentima, ali pozivamo sve studente da ih makar pokušaju riješiti.

Programi u datotekama *Sortovi.c* i *Premetaljka.c*, koji se nalaze u repozitoriju predmeta, poslužit će Vam kao dobra osnova za ovu laboratorijsku vježbu.

1. a) zadatak(0.5 bodova)

Za pripremu napišite funkcije koje će sortirati zadani niz brojeva:

- bubble sortom
- algoritmom biranja (*selection sort*), rekurzivnu i nerekurzivnu inačicu
- quicksortom
- mergesortom

Sve funkcije moraju imati mogućnost uzlaznog ili silaznog sortiranja u ovisnosti o ulaznom parametru.

Također, napišite vlastitu ili preinačite funkciju s predavanja koja generira sve permutacije znakovnog niza.

Nijedna od ovih funkcija ne bi izvoditi ulazno/izlazne operacije (npr. pozivi printf, pisanje u datoteku i sl.).

Za sve funkcije iz ovog dijela zadatka napišite apriorne složenosti.

1. b) zadatak - mjerenja (1 bod)

U ovom zadatku mjeri se vrijeme izvođenja funkcija iz zadatka a) kako bi se odredila složenost a posteriori.
Prevedite program **gcc**-om koristeći i zastavicu *–O3*. Prije pokretanja ovog programa pogasite sve manje važne programe na Vašem računalu kako bi njihov utjecaj na mjerenje bio što manji.

Koristeći funkcije iz zadatka a) i parametre u Tablica 1, u glavnom dijelu programa:

- koristite generator slučajnih brojeva s Vašim JMBAG-om bez vodećih nula kao argumentom za poziv funkcije **srand**(umjesto vremena)
- za svaku funkciju **fja** iz a)
  - otvorite posebnu tekstualnu datoteku u koju ćete pisati podatke
  - generirajte **ulaz**:
    - ako je **fja** funkcija za sortiranje: neka **ulaz** bude niz slučajnih brojeva (cijeli brojevi iz intervala [1,RAND_MAX]) veličine ***kraj***
    - ako je **fja** funkcija za generiranje permutacija: neka **ulaz** bude proizvoljni string duljine ***kraj***
  - za **n**=***start***, ***start+korak***, ***start+2*korak***,…., ***kraj***
    - mjerite vrijeme izvođenja ***agreg*** poziva* funkcije **fja** nad istim nizom koji se sastoji od prvih **n** elemenata **ulaza**
      - kod sortova pazite da ne koristite ponovno već sortirani niz i sortirate silazno
      - kod generatora permutacija pazite da je podniz string
    - zapišite podatke mjerenja za svaki algoritam u njegovu tekstualnu datoteku u sljedećem formatu:

      „%d\t%f\n“,n, prosj_trajanje

      gdje su: *n*- veličina obrađenog niza

      prosj_trajanje - mjerenje izvođenja ***agreg*** poziva funkcije **fja**(nad nizom veličine **n**) podijeljeno s ***agreg***

algoritam	*start*	*korak*	*kraj*	*agreg*
Bubble sort	0	250	12500	30
Selection sort (rekurzivni i nerekurzivni)	0	250	12500	50
quicksort	0	1500	112500	150
mergesort	0	1500	112500	120
permutacije	2	1	11	40

Tablica 1 Parametri za b) dio zadatka†

* Koristimo radi nepreciznosti mehanizma za mjerenje vremena izvođenja na računalu kao i radi smanjenja utjecaja drugih procesa na mjerenje.

† † Empirijski postavljeni podatci za koje izvođenje za svaku funkciju traje do 4min na računalu starom ~4 godine. Za novija, brža, računala možete mijenjati parametre ***kraj*** i ***agreg*** na više koristeći npr. Moorev zakon kao heuristiku za odabir vrijednosti.

1. c) zadatak – a posteriori analiza (1 bod)

Dobivene datoteke s rezultatima mjerenja iz zadatka b) iskoristite za određivanje funkcije koja u intervalu u kojem ste mjerili trajanje izvođenja, tj. [**start, kraj**][‡], dovoljno dobro opisuje zavisnost vremena izvođenja o veličini ulaza. Koristiti ćete program ***Eureqa*** [§].

Slijed postupaka:

1. Preuzeti program u obliku zip datoteke s ove poveznice, raspakirajte na željeno mjesto i pokrenite.
2. Na kartici(tab) **„Enter Data“**
    a. Kopirajte podatke iz jedne od svojih datoteka koje ste stvorili u zadatku b) u program *Eureqa* poštujući format iz primjera koji je tamo već otvoren (zalijepite podatke preko postojećih u prva dva stupca)
    b. Kako imate samo dva stupca podataka, pobrišite sve podatke iz trećeg stupca
    c. Varijable nazovite **N** i **trajanje**
    d. Uvedite oznaku **H** za omjer **N/trajanje**, i izračunajte taj omjer za zadnji redak u vašim podatcima. Zaokružite na najbližu potenciju broja 10.
3. Prijeđite na karticu **„Prepare Data“**
    a. Odaberite varijablu **N** te s desne strane u opcijama označite „Normalize and scale offset of (N)“
        i. Ako je riječ o podatcima za algoritam za sortiranje, u „Divide by“ upišite[**] **H**
4. Prijeđite na karticu **„Set target“**
    a. U „Formula building-blocks“ označite sve opcije u kategorijama „Basic“ i „Exponential“, sve ostale opcije moraju biti neodabrane
    b. Pod „Error metric“ izaberite „Squared error“
    c. Pod „Data Splitting“ izaberite „Split data points for extrapolating future values“
5. Prijeđite na karticu „Start search“
    a. Kliknite „Run“
    b. Ako Vam se pojavi prozor za registraciju, registrirajte se kao beta-tester(kako se podatci ne provjeravaju, možete unijeti bilo koje podatke)
    c. Po potrebi ponovo kliknite na „Run“
6. Algoritam traži analitički oblik funkciju koja najbolje opisuje zavisnost između veličine ulaza i trajanja izvođenja (traži regresijsku funkciju).
7. Nakon nekog vremena (nekoliko minuta) možete zaustaviti program pritiskom na „Stop“[††]
8. Prijeđite na karticu „View results“ i na popisu „Best solutions of different sizes “ izaberite jedno od rješenja s najmanjim vrijednostima podatka „Fit“ te zapišite rješenje.[‡‡] Klikom na svako rješenje možete u grafu na desnoj strani vidjeti kako se funkcija prilagodila zadanim točkama(Vašim mjernim rezultatima).[§§]
9. Ponovite za sve preostale datoteke s mjerenjima

[‡] Izraz možete, s oprezom, koristiti i za procjenu(ekstrapolaciju) vremena izvođenja izvan intervala mjerenja.
[§] Za više informacija o programu, pogledajte na http://creativemachines.cornell.edu/eureqa.
[**] Svrha dijeljenja s veličinom H je izbjegavanje numeričkih problema prilikom računanja s brojevima bitno različitih redova veličina.
[††] Što dulje ostavite program da traži, možete dobiti bolji izraz.
[‡‡] U slučaju izjednačenih izraza izaberite onaj s najmanjom vrijednosti podatka„Size“
[§§] Vjerojatno za quicksort i mergesort rezultat nećete dobiti C*n*log(n), ali biste trebali dobiti superlinearnu funkciju. Razlog je što za veličine ulaza pri kojima mjerimo trajanje izvođenja dobiveni izrazi dobro opisuju podatke.

1. d) zadatak – dodatak (0.5 bodova)

Napišite rekurzivnu funkciju za *bubble sort* algoritam.

1. e) zadatak * (1 bod)

Napišite nerekurzivne funkcije za *quick sort* i *mergesort* algoritme (npr. moguće korištenjem vlastitih stogova).