Lab6: Complexitate algoritmica

Formatori:
Tutor: Stiegelbauer Paul
Tutor: Iovanovici Alexandru
+1

Data de începere a cursului:
19.02.2024
Utilizatori înscriși
Calendar
Note

Lab6: Complexitate algoritmica

Scopul acestei lucrari de laborator este determinarea experimentala a claselor de complexitate algoritmica (prin masurarea timpilor de executie) cat si proiectarea si implementarea unor algoritmi simpli in timp subliniar.

1. Masurarea timpului de executie

1.1 Utilizarea utilitarului time

Pe sisteme Linux/UNIX aveti "by default" utilitarul time care permite executarea unui program (primit ca si argument in linie de comanda), masurarea si afisarea timpilor: total de executie, utilizat de sistemul de operare, si timpul efectiv de rulare al programului analizat.

Spre exemplu un posibil output pe MacOS este

```
./a.out  0.02s user 0.00s system 79% cpu 0.031 total
```

1.2 Utilizarea functiilor din time.h

Pentru o analiza mai amanuntita (spre exemplu a unei singure functii sau chiar a unei portiuni dintr-o functie) se poate folosi functia clock, declarata in time.h.
In mod uzual se apeleaza clock() la inceputul si sfarsitul portiunii de analizat, se scad valorile si converteste in timp-real, prin impartire la CLOCKS_PER_SEC (numarul de "cloci" ai procesorului), astfel:

```
/* preluat din referinta [2] */
#include <time.h>

clock_t start, end;
double cpu_time_used;

start = clock();
… /* Do the work. */
end = clock();
cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
```

Rezolutia tipica oferita cf. [2] este oriunde intre 1e-2 si 1e-6 dintr-o secunda. C++ ofera in std::chrono o serie de functie cu precizie sporita. Deoarece pentru aplicatii practice de multe ori timpii de executie, pentru seturi reduse de date, procesoare performante si algoritmi efiecienti, sunt in zona pragului de zgomot/eroare se recomanda masurarea unui set mai mare de executii ale aceluiasi algoritm (spre exemplu 100 de apeluri ale functiei de sortare analizate) si impartirea timpului gasit la numarul de apeluri.

Aspecte ce tin de hardware (mecanisme de memorie cache, lucrul cu discul) si/sau sistem de operare (multithreading) pot influenta semnificativ rezultatele, deci se recomanda reluarea analizei in anii superiori dupa parcurgerea disciplinelor cu accent pe arhitectura sistemelor de calcul si/sau sisteme de operare.

2. Aplicatii

1. Implementati un algoritm care primeste ca parametru un numar k si returneaza al k-lea termen al sirului 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5 ... (solutie in timp constant)
2. Implementati un algoritm care primeste ca parametru un numar k si returneaza al k-lea termen al sirului 1 2 1 3 2 1 4 3 2 1 5 4 3 2 1 ... (solutie in timp constant)
3. Implementati o functie care citeste de la intrarea standard valori naturale (citirea se opreste la intalnirea primei valori de 0) si afiseaza cele mai mari trei valori de trei cifre care nu se gasesc printre numerele citite.
4. **a)** Implementati o functie cu prototipul **int findElemLin(int v[], unsigned n, int x)** care returneaza pozitia primei aparitii a elementului **x** in vectorul **v** avand **n** elemente sau **-1** daca acel numar nu apare in vector. Se va folosi un algoritm cu complexitate liniara (spre exemplu https://en.wikipedia.org/wiki/Linear_search#Basic_algorithm).
   b) Implementati o functie cu prototipul **int findElemBin(int v[], unsigned n, int x)** cauta elementul, similar cu cerinta de la punctul **a)**, dar opereaza asupra unui vector sortat. Pentru rezolvare se va folosi un algoritm cu complexitate logartimica, mai precis "cautarea binara", prin apelul adecvat al functiei **bsearch** din **stdlib.h**.
   c) Masurati timpul de executie pentru un numar semnificativ de rulari ale celor doua functii (spre exemplu 100 de rulari) si repetati acest proces pentru un set de date din ce in ce mai mare (spre exemplu de la 100 de elemente la 50000 de elemente din 100 de rulări).
   Tipariti datele pe iesirea standard si redirectati iesirea spre un fisier .csv, iar mai apoi reprezentati grafic rezultatele facand observatii asupra timpilor de executie raportat la marimea datelor de intrare.
   Obs:
   Pentru generarea vectorilor cu date se va folosi una dintre functiile create la prima sesiune de Proiect (pentru cerinta (b) este necesar sa folositi functia de generare care primeste si pointer la functie, deoarce trebuie generate numere cu ajutorul unei "politici"; in caz extrem puteti genera un vector pseudoaleator uniform distribuit pe care mai apoi sa il sortati);

4. Resurse

1. https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/
2. https://www.gnu.org/software/libc/manual/html_node/CPU-Time.html

◄ Test1 Lab restant

Sari la...

Model evaluare1 lab. Pentru sesiunea de proiect ►