Міністерство освіти і науки України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій

Кафедра компютерних наук

Методичні вказівки до виконання лабораторних робіт з дисципліни „Теорія алгоритмів та структур даних"

для студентів напрямку 0804 "Комп'ютерні науки" (заочної форми навчання)

Тернопіль-2007

О.В. Горішний //Методичні вказівки до виконання лабораторних робіт з дисципліни “Теорія алгоритмів та структур даних”, для студентів напрямку 0804 “Комп'ютерні науки”. - Тернопіль, 2007. - 13с.

Укладачі: Горішний Олег Володимирович, викладач кафедри комп'ютерних наук, ТНЕУ.

Відповідальний за випуск: Дивак Микола Петрович, д.т.н., проф., завідувач кафедри комп'ютерних наук, ТНЕУ.

Рецензенти: завідувач кафедри безпеки інформаційних технологій ТНЕУ, д.т.н., проф. Карпінський М.П.

доцент кафедри комп'ютерних технологій і систем управління Івано-Франківського національного технічного університету нафти і газу, к.т.н., доц. Малько О.Г.

Затвеоджено на засіданні кафедри комп'ютерних наук ТНЕУ.
Протокол №6 10 січня 2007 р.

Лабораторна робота №1

Тема: Дослідження алгоритмів пошуку найбільшого спільного дільника.

Мета: навчитися визначати часову складність алгоритму.

Основні теоретичні відомості.

Основними мірами обчислювальною складності алгоритмів є:

часова складність, яка характеризує час, необхідний для виконання алгоритму на даній машині; цей час, як правило, визначається кількістю операцій, які потрібно виконати для реалізації алгоритму;

ємнісна складність, яка характеризує пам'ять, необхідну для виконання алгоритму.Часова та ємнісна складність тісно пов'язані між собою. Обидві є функціями від розміру вхідних даних. Надалі обмежимося тільки аналізом часової складності.

Складність алгоритму описується функцією f(n), де n - розмір вхідних даних. Важливе теоретичне і практичне значення має класифікація алгоритмів, яка бере до увагу швидкість зростання цієї функції.

Визначення.
Кажуть, що f(n) = O (g(n)), якщо існує така константа c > 0, що для будь-якого n виконується нерівність: |f(n)| <= c |g(n)|.Оскільки і розмір вхідних даних, і кількість операцій є величинами невід'ємними (а фактично - додатніми), модулі можна опустити.

Виділяють такі основні класи алгоритмів:

логарифмічні: $f(n) = O (\log_2 n)$;

лінійні: f(n) = O (n);

поліноміальні: $f(n) = O (n^m)$; тут m - натуральне число, більше від одиниці; при m = 1 алгоритм є лінійним;

експоненційні: $f(n) = O (a^n)$; a - натуральне число, більше від одиниці.

Для однієї й тієї ж задачі можуть існувати алгоритми різної складності. Часто буває і так, що більш повільний алгоритм працює завжди, а більш швидкий - лише за певних умов.

Індивідуальні завдання.

Варіант 1.

```
program example(input, output);
var x, y: integer;
function gcd( u, v: integer) : integer;
var t: integer;
begin
if u<v then t:=u else t:=v;
while (u mod t<>O) or (vmod t<>O) do t:=t-1;
```

```
gcd:=t
end ;
begin
while not eof do
begin
readln (x, y ) ;
writeln(x, y, gcd(abs(x), abs(y)));
end
end.
```

Варіант 2.

```
program matrixadd(input, output);
const maxN=lO;
var p, q, r: array [O..maxN, O..maxN] of real;
N, i, j: integer;
begin
readln (N) ;
for i:=O to N-l do for j:=O to N-l do read(p[i, j]);
for i:=O to N-l do for j:=O to N-l do read(q[i, j]);
for i:=O to N-l do for j:=O to N-l do r[i, j]:=p[i, j]+q[i, j];
for i:=O to N-l do for j:=O to N do
if j=N then writeln else write(r[i, j]);
end.
```

Варіант 3.

```
program gauss(input, output);
const maxN=50;
var a: array[I..maxN, l..maxN] of real;
i, J', k, N: integer;
begin
readln (N) ;
for j:==l to N do
begin for k:=l to N+1 do read(ab, k]); readln end;
for i:==l to N do
for j:=i+l to N do
for k:=N+l downto i do
ab,k]:=aIj,k]-a[i,k]*ab,i]/a[i,i];
for j:==l to N do
begin for k:=l to N+1 do write(ab, k]); writeln end;
end.
```

Варіант 4.

```
procedure eliminate;
var i, j, k, max: integer;
t: real;
```

```
begin
for i:=l to Ndo
begin
max:=i;
for j:=i+l to N do
if abs(aIj, i])>abs(a[max, i]) then max:=j;
for k:=i to N+l do
begin t:=a[i, k]; a[i, k] :=a[max, k]; a[max, k] :=t end;
for j:=i+l to N do
for k:=N+l downto i do
ab,k]:=ab,k]-a[i,k]*ab,i]/a[i,i];
end
end ;
```

Варант 5.

```
procedure insertion;
var i, j, v: integer;
begin
for i:=2 to N do
begin
v:=a[i]; j:=i;
while ab-1]>v do
begin ab] :=ab-11; j:=j-1 end;
ab] :=v
end ;
end ;
```

Варіант 6.

```
procedure shellsort;
label 0;
var i, j, h, v: integer;
begin
h:=l; repeat h:=3*h+l until h>N;
repeat
h:=h div 3;
for i:=h+l to N do
begin
v:=a[i]; j:=i;
while ab-h]>v do
begin
a[j]:=ab-h]; j : = j - h ;
if j<=h then goto 0
end ;
0: ab]:=v
end ;
until h= 1;
end ;
```

Варіант 7.

```
procedure bubblesort;
var j, t: integer;
begin
repeat
t:=a[l];
for j:=2 to N do
if ab--l]>alj] then
begin t:=:a[j-I]; ab-1]:=ab]; a[j]:=t end
until t=a[l];
end ;
```

Варіант 8.

```
procedure straig&radix( b: integer) ;
var i, j, pass: integer;
begin
for pass:=0 to (b div m)-1 do
begin
for j:=O to M-l do countlj] :=O;
for i:=l to N do
count[bits(a[i],pass*m, m)] :=count[bits(a[i],pass*m, m)]+l;
for j:=l to M-l do
countIj]:=countIj-l]+countb];
for i:=N downto 1 do
begin
t[count[bits(a[i],pass*m,m)]]:=a[i];
count[bits(a[i],pass*m,m)]:=count[bits(a[i],pass*m,m)]-l;
end ;
for i:=l to N do a[i]:=t[i];
end ;
end ;
```

Варіант 9.

```
procedure select(k: integer) ;
var v t i j 1 r: integer; , 7 9 t )
begin
l:=l; r:=N;
while r>l do
begin
v:=a[r]; i:=l-1; j : = r ;
repeat
repeat i:=i+l until a[i]>=v;
repeat j:=j-1 until ab]<=v;
t:=a[i]; a[i]:=alj]; ab]:=t;
```

```
until j<=i;
alj]:=a[i]; a[i]:=a[r]; a[r]:=t;
if i>=k then r:=i-1;
if i<=k then l:=i+l;
end ;
end ;
```

Варіант 10.

```
function wrap: integer;
var i, min, M: integer;
minangle, v: real;
t: point;
begin
min:=l;
for i:=2 to Ndo
if p[i].y<p[min].y then min:=i;
M:=O; p[N+1]:=p[min]; minangle:=O.O;
repeat
M:=M+1; t:=p[M]; p[Mj:=p[min]; p[min]:=t;
min:=N+l; v:=minangle; minangle:=360.0;
for i:=M+1 to N+I do
if theta(p[M],p[i])>v then
if theta(p[M], p[i])<minangle then
begin min:=i; minangle:=theta(p[M], p[min]) end;
until min= N+1;
wrap:=M;
end ;
```

Варіант 11.

```
function grahamscan : integer;
var i, j, min, M: integer;
1: line; t: point;
begin
min:=l;
for i:=2 to N do
if p [i] .y<p [min] .y then min:=i;
t:=p[l]; p[l]:=p[min]; p[min]:=t;
shellsort ;
M:=2;
for i:=4 to Ndo
begin
M:=M+2;
repeat
M:=M-1;
I.pl:=p[M]; l.p2:=p[M-I];
until same(l,p[l],p[i])>=O;
t:=p[M+I]; p[M+I]:=p[i]; p[i]:=t;
end ;
```

```
grahamscan : =M;
end ;
```

Варіант 12.

```
procedure gridrange(rect : rectangle) ;
var t: link;
i, j: integer;
begin
for i:=(rect.xl div size) to (rect.x2 div size) do
for j:=(rect.yl div size) to (rect.y2 div size) do
begin
t:=grid[i, j];
while t<>z do
begin
if insiderect( tt.p, rect) then write(name( tt.p.info));
t:=tf.next
end
end
end ;
```

Варіант 13.

```
procedure densepfs;
var k, min, t: integer;
begin
for k:=l to Vdo
begin vaJ[k]:=-unseen; dad[k]:=O end;
vaJ[O]:=-(unseen+l);
min:=l;
repeat
k:=min; vaJ[k]:=-vaJ[k]; min:=O;
if vaJ[k]=unseen then vaJ[k] :=O;
for t:=l to Vdo
if vaJ[t]<O then
begin
if (a[k, t]<>O) and (vaJ[t]<-priority) then
begin vaJ[t] :=-p riority; dad [ t] :=k end;
if vaJ[t]>vaJ[min] then min:=t;
end
until min=O;
end ;
```

Лабораторна робота №2

Тема: Способи конструювання ефективних алгоритмів.

Мета: навчитися будувати алгоритми з мінімальною часовою складністю для вирішення поставлених задач.

Індивідуальні завдання.

1. Побудувати алгоритм множення двох натуральних чисел без використання операції множення.
2. Побудувати алгоритм знаходження n-го простого числа.
3. Побудувати алгоритм знаходження біноміального коефіцієнта C_n^k для цілих n та k, де $0 \le k \le n$. Використайте наступні співвідношення: $C_n^0 = C_n^n = 1, C_{(n+1)}^{(k+1)} = C_n^{(k+1)} + C_n^k$.
4. Побудувати алгоритм знаходження всіх натуральних розв'язків нерівності $x^2 + y^2 < n$ для заданого натурального числа n.
5. Побудувати алгоритм, який для введеного цілого числа R знайде кількість точок з цілочисельними координатами, які лежать всередині кулі з радіусом R і центром в початку координат.
6. Побудувати алгоритм, який для заданого цілого числа *x* шукатиме найбільш близикий до $\sqrt{x}$ дріб n/m , де $m < 100$.
7. Побудувати алгоритм для знаходження кількості щасливих білетів із шестизначними номерами. Білет вважається щасливим, якщо сума перших трьох цифр дорівнює сумі трьох останніх.
8. Побудувати алгоритм знаходження суми $\frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + ... + \frac{1}{n!}$, складність якого була б лінійною.
9. Є 25 золотих монет. Всі вони мають однакову вагу, за винятком однієї монети з дефектом, яка важить менше інших. Розробіть алгоритм, що визначає дефект за три зважування. Яка максимальна кількість монет, для яких можна визначити монету з дефектом не більш ніж за три зважування на вагах з чашками?
10. Три місіонери і три канібали знаходяться на одному березі річки; човен може перевезти двох чоловік. Всі вони хочуть перебратися через річку. Не можна допустити, щоб на одному березі знаходилася група місіонерів з численнішою групою канібалів. Розробіть процедуру, що дозволяє всім шістьом перебратися через річку.
11. Побудувати алгоритм, який здійснює обхід шахівниці конем. При чому кінь ніколи не буває двічі на одній клітці.
12. Побудувати алгоритм знаходження найбільшої спільної підпослідовності двох послідовностей.
13. Побудувати алгоритм, який набиратиме суму N копійок з набору монет 1, 2, 5,10, 25 та 50 копійок.

Лабораторна робота №3

Тема: Дослідження алгоритмів сортування.

Мета: навчитися досліджувати алгоритми сортування.

Основні теоретичні відомості.

Задача сортування полягає у впорядкуванні деякої неврегульованої множини ключів за збільшенням чи зменшенням в чисельному чи лексикографічному порядку.

Всі алгоритми сортування можна розбити на групи:

- алгоритми вибірки. З вхідної множини вибирається наступний за критерієм елемент і включається в вихідну множинуна місце, наступне за номером.
- алгоритми включення. З вхідної множини вибирається наступний за номером елемент і включається в вихідну множину на те місце, яке він повинен займати відповідно дор критерію впорядкованості.
- алгоритми розподілу. Вхідна множина розбивається на ряд підмножин і сортування ведеться всередині кожної такої підмножини.
- алгоритми злиття. Вихідна множина отримується шляхом злиття маленьких впорядкованих підмножин.

Індивідуальні завдання.

1. Написати програму, яка реалізує алгоритм сортування методом бульбашки у порядку зростання та дослідити його складність.
2. Написати програму, яка реалізує алгоритм сортування методом бульбашки у порядку спадання та дослідити його складність.
3. Написати програму, яка реалізує алгоритм сортування методом перестановки у порядку зростання та дослідити його складність.
4. Написати програму, яка реалізує алгоритм сортування методом перестановки у порядку спадання та дослідити його складність.
5. Написати програму, яка реалізує алгоритм сортування методом підрахунку у порядку зростання та дослідити його складність.
6. Написати програму, яка реалізує алгоритм сортування методом підрахунку у порядку спадання та дослідити його складність.
7. Написати програму, яка реалізує алгоритм сортування методом Шелла у порядку зростання та дослідити його складність.
8. Написати програму, яка реалізує алгоритм сортування методом Шелла у порядку спадання та дослідити його складність.
9. Написати програму, яка реалізує алгоритм сортування методом швидкого сортування у порядку зростання та дослідити його складність.

10. Написати програму, яка реалізує алгоритм сортування методом швидкого сортування у порядку спадання та дослідити його складність.
11. Написати програму, яка реалізує алгоритм сортування методом швидкого сортування з розподілом за алгоритмом Ломуто у порядку зростання та дослідити його складність.
12. Написати програму, яка реалізує алгоритм сортування методом швидкого сортування з розподілом за алгоритмом Ломуто у порядку спадання та дослідити його складність.
13. Написати програму, яка реалізує алгоритм сортування методом вибірки у порядку зростання та дослідити його складність.
14. Написати програму, яка реалізує алгоритм сортування методом вибірки у порядку спадання та дослідити його складність.
15. Написати програму, яка реалізує алгоритм сортування методом порозрядного цифрового сортування у порядку зростання та дослідити його складність.
16. Написати програму, яка реалізує алгоритм сортування методом порозрядного цифрового сортування у порядку спадання та дослідити його складність.

Зміст звіту.

Звіт повинен містити:

- титульну сторінку;
- тему, мету роботи;
- завдання, робочий варіант тексту програми, результати обчислень;
- висновки.

Лабораторна робота №4

Тема: Дослідження структур даних.

Мета: навчитися досліджувати структури даних.

Основні теоретичні відомості.

Без розуміння структур даних і алгоритмів неможливо створити програмний продукт. Структури даних покликані полегшити організацію та зберігання даних, а також полегшити роботу багатьох алгоритмів, зробивши їх більш ефективними і менше прив'язаними до ресурсів комп'ютера. Структури даних можна розбити на статичні (масиви, структури, ...), напівстатичні (стеки, черги, деки, ...), динамічні (списки). Нелінійні (графи, дерева). Правильний вибір структури дозволяє суттєво пришвидшити виконання програм, що реалізують потрібний алгоритм.

Індивідуальні завдання.

Написати програму, яка реалізує вказану структуру даних:

1. стек;
2. черга;
3. дек;
4. лінійний однозв'язний список;
5. лінійний двозв'язний список;
6. бінарне дерево;
7. червоно-чорне дерево;
8. графи;
9. нелінійний розгалужений список;
10. розріджені масиви;
11. збалансовані дерева
12. Б-дерева;
13. ХЕШ-таблиці;
14. тернатрні дерева

Зміст звіту.

Звіт повинен містити:

- титульну сторінку№;
- тему, мету роботи;
- завдання, робочий варіант тексту програми, результати обчислень;
- висновки.