**Московский авиационный институт**

**(национальный исследовательский университет)**

**Факультет «Прикладная математика и физика»**

**Кафедра «Вычислительная математика и программирование»**

**Курсовой проект по курсу дискретной математики**

Выполнил: Забарин Н.И. 80-08Б

Руководитель: к. ф.-м.н.с. доцент Алексеев Н.С.

**Москва, 2016**

**Оглавление**

[Введение](#h.tqejpm7c0hj4)

[Задачи](#h.ey8to68vrt9q)

[Задача 1.](#h.d0s4epde55h1)

[Условие:](#h.b3ee4juifv1f)

[Метод решения:](#h.eaf6p8ymumag)

[Листинг:](#h.iy1nlircnpxq)

[Задача 2.](#h.e9n9ar5e9bad)

[Условие:](#h.j5rhgsxpc1j9)

[Метод решения:](#h.owijuv3s18h6)

[Листинг:](#h.3bhzwsliqomo)

[Задача 3.](#h.1sgwqhqn2frc)

[Условие:](#h.jnjamsezw2e)

[Метод решения:](#h.8wss11gq1sij)

[Листинг:](#h.95roahqhyuv0)

[Задача 4.](#h.27frmuxe5u67)

[Условие:](#h.xa6213pj0zvf)

[Метод решения:](#h.4k33osv2ukf3)

[Листинг:](#h.qb4i12e3gqzp)

[Задача 5.](#h.5voxg4f5dvgs)

[Условие:](#h.wfc0yrpwpota)

[Метод решения:](#h.k850xw8dn5o8)

[Листинг:](#h.kw42j5k95h9e)

[Заключение](#h.icujp28yot8l)

# Введение

# Задачи

## Задача 1.

### Условие:

Напишите программу, которая для заданного массива A находит количество пар (i, j) таких, что i < j и ai > aj.

### Метод решения:

Построим дерево Фенвика, в котором будем хранить кол-во элементов массива которые меньше текущего. Будем по очереди добавлять в него элементы массива и сразу считать сколько чисел, которые больше текущего мы уже встретили, просуммировав это мы и получим ответ.

На каждом шаге мы делаем порядка логарифм N операций, всего N шагов, получаем сложность в O(NlogN)

### Листинг:

program d02;

{$APPTYPE CONSOLE}

uses

SysUtils,

Math;

var

mas, d, used : array [0..1000001] of integer;

n, i : integer;

procedure up (var x : integer);

begin

x := x or (x + 1);

end;

procedure down (var x : integer);

begin

x := (x and (x + 1)) - 1;

end;

procedure UpDate(i, x : integer);

begin

while i < 1000000 do begin

inc(used[i], x);

up(i);

end;

end;

function Sum (r : integer) : int64;

begin

result := 0;

while r >= 0 do begin

inc(result, used[r]);

down(r);

end;

end;

begin

reset(input, 'inverse.in');

rewrite(output, 'inverse.out');

fillchar(used, sizeof(used), 0);

fillchar(d, sizeof(d), 0);

fillchar(mas, sizeof(mas), 0);

read(n);

for i := 0 to n - 1 do

read(mas[i]);

for i := 0 to n - 1 do begin

UpDate(mas[i], 1);

d[mas[i]] := sum(1000000) - sum(mas[i]);

end;

n := 0;

for i := 0 to 1000001 do inc(n, d[i]);

write(n);

end.

## 

## Задача 2.

### Условие:

Даны две строки длиной не больше 1000, необходимо найти наибольшую общую подпоследовательность этих строк.

### Метод решения:

Квадратная динамика по длинам строк. Если символы строк на текущих позициях совпадают, то берем ответ для позиций смещенных на 1 и прибавляем единицу, иначе берем максимум из смещений только одной из позиций. Ответ получаем обратным ходом динамики.

Сложность алгоритма O(NM), где N и M длины строк.

### Листинг:

program a03;

{$APPTYPE CONSOLE}

uses

SysUtils, Math;

var

s1, s2, ans : string;

i, j, x, y : integer;

d : array [0..2000, 0..2000] of integer;

begin

reset(input, 'lcs2.in');

rewrite(output, 'lcs2.out');

readln(s1);

readln(s2);

fillchar(d, sizeof(d), 0);

for i := 1 to length(s1) do

for j := 1 to length(s2) do

if s1[i] = s2[j] then

d[i, j] := d[i - 1, j - 1] + 1

else

d[i, j] := max(d[i, j - 1], d[i - 1, j]);

x := length(s1);

y := length(s2);

ans := '';

while (x > 0) and (y > 0) do

if s1[x] = s2[y] then begin

ans := s1[x] + ans;

dec(x);

dec(y);

end

else

if d[x - 1, y] > d[x, y - 1] then

dec(x)

else

dec(y);

writeln(ans);

end.

## 

## Задача 3.

### Условие:

Назовем мега-инверсией в перестановке P такую тройку (i, j, k), что Pi > Pj > Pk и i < j < k. Написать алгоритм для быстрого подсчета количества мега-инверсий в данной перестановке.

### Метод решения:

Построим три RMQ: первое для указания какие числа уже есть в перестановке, второе для подсчета инверсий, третье для подсчета мега инверсий. Будем последовательно обновлять эти три дерева. Для получаения ответа просуммируем все последнее дерево.

Сложность алгоритма O(NlogN)

### Листинг:

program a01;

{$APPTYPE CONSOLE}

uses

SysUtils,

Math;

type

TElem = record

value : int64;

l, r : integer;

end;

TTree = array [1..300000] of TElem;

var

used, inv, minv : TTree;

mas : array [1..200000] of integer;

n, i : integer;

function func (a, b : int64) : int64;

begin

result := a + b;

end;

procedure Build(var tree : TTree; l, r, id : integer);

begin

tree[id].l := l;

tree[id].r := r;

if r - l = 1 then

tree[id].value := mas[l]

else begin

Build(tree, l, (r + l) div 2, id \* 2);

Build(tree, (r + l) div 2, r, id \* 2 + 1);

tree[id].value := func(tree[id \* 2].value, tree[id \* 2].value);

end;

end;

procedure UpDate(var tree : TTree; id, n : integer; v : int64);

begin

if (n >= tree[id].r) or (n < tree[id].l) then exit;

if tree[id].r - tree[id].l = 1 then

tree[id].value := v

else begin

Update(tree, id \* 2, n, v);

Update(tree, id \* 2 + 1, n, v);

tree[id].value := func(tree[id \* 2].value, tree[id \* 2 + 1].value);

end;

end;

function Proc(var tree : TTree; l, r, id : integer) : int64;

begin

if (tree[id].l >= r) or (tree[id].r <= l) then begin

result := 0;

exit;

end;

if (l <= tree[id].l) and (tree[id].r <= r) then begin

result := tree[id].value;

exit;

end;

result := func(Proc(tree, l, r, id \* 2), Proc(tree, l, r, id \* 2 + 1));

end;

begin

reset(input, 'mega.in');

rewrite(output, 'mega.out');

fillchar(used, sizeof(used), 0);

fillchar(inv, sizeof(inv), 0);

fillchar(minv, sizeof(minv), 0);

fillchar(mas, sizeof(mas), 0);

readln(n);

Build(used, 0, n, 1);

Build(inv, 0, n, 1);

Build(minv, 0, n, 1);

for i := 0 to n - 1 do

readln(mas[i]);

for i := 0 to n - 1 do begin

UpDate(used, 1, mas[i] - 1, 1);

UpDate(inv, 1, mas[i] - 1, Proc(used, mas[i], n, 1));

UpDate(minv, 1, mas[i] - 1, Proc(inv, mas[i], n, 1));

end;

write(Proc(minv, 0, n, 1));

end.

## 

## Задача 4.

### Условие:

### Реализовать сбалансированное дерево поиска, поддерживающее следующие команды: добавление элемента, удаление элемента, проверка на существование элемента, следующий/предыдущий по возрастанию элемент после данного, k-тый по возрастанию элемент.

### Метод решения:

Основным алгоритмом является декартово дерево со случайным вторым ключом. Все операции перечисленные в условии являются стандартными для данного алгоритма.

Время выполнения каждого запроса O(N), где N размер дерева.

### Листинг:

program b01;

{$APPTYPE CONSOLE}

uses

SysUtils;

const

num = 100000;

type

TElem = record

x, y : integer;

l, r : integer;

end;

TPair = record

a, b : integer;

end;

var

stack, sz : array [0..num] of integer;

r, t, i, n, q : integer;

a : array [1..num] of TElem;

s : string;

tmp : TPair;

function cnt(t : integer): integer;

begin

result := 0;

if t = -1 then exit;

result := sz[t];

end;

procedure update(t : integer);

begin

if t = -1 then exit;

sz[t] := cnt(a[t].l) + cnt(a[t].r) + 1;

end;

procedure push(a : integer);

begin

inc(r);

stack[r] := a;

end;

function new : integer;

begin

result := stack[r];

dec(r);

end;

function merge (t1, t2 : integer) : integer;

begin

result := t2;

if t1 = -1 then exit;

result := t1;

if t2 = -1 then exit;

if (a[t1].y > a[t2].y) then begin

a[t1].r := merge(a[t1].r, t2);

update(t1);

result := t1;

end

else begin

a[t2].l := merge(t1, a[t2].l);

update(t2);

result := t2;

end

end;

function split (t, k : integer): TPair;

var

tmp : TPair;

begin

result.a := -1;

result.b := -1;

if t = -1 then exit;

if k > a[t].x then begin

tmp := split(a[t].r, k);

a[t].r := tmp.a;

update(t);

update(tmp.b);

result.a := t;

result.b := tmp.b;

end

else begin

tmp := split(a[t].l, k);

a[t].l := tmp.b;

update(tmp.a);

update(t);

result.a := tmp.a;

result.b := t;

end;

end;

function exist (t, k: integer) : boolean;

begin

result := false;

if t = -1 then exit;

result := true;

if a[t].x = k then exit;

if a[t].x > k then begin

result := exist(a[t].l, k);

exit;

end;

if a[t].x < k then begin

result := exist(a[t].r, k);

exit;

end;

end;

function insert (t, k : integer) : integer;

var

p : integer;

tmp : TPair;

begin

result := t;

if exist(t, k) then exit;

p := new;

a[p].x := k;

a[p].y := random(100000000);

a[p].l := -1;

a[p].r := -1;

tmp := split(t, k);

tmp.a := merge(tmp.a, p);

result := merge(tmp.a, tmp.b);

end;

function delete (t, k : integer): integer;

begin

result := -1;

if t = -1 then exit;

if a[t].x < k then begin

a[t].r := Delete(a[t].r, k);

result := t;

exit;

end;

if a[t].x > k then begin

a[t].l := delete(a[t].l, k);

result := t;

exit;

end;

result := merge(a[t].l, a[t].r);

end;

procedure exists (t, k: integer);

begin

if exist(t, k) then

writeln('true')

else

writeln('false');

end;

function next (t, key : integer): TPair;

var

tmp : TPair;

v : integer;

begin

tmp := split(t, key + 1);

v := tmp.b;

result.b := v;

result.a := tmp.a;

if v = -1 then exit;

while a[v].l <> -1 do

v := a[v].l;

tmp.a := merge(tmp.a, tmp.b);

tmp.b := v;

result := tmp;

end;

function prev (t, key : integer) : TPair;

var

tmp : TPair;

v : integer;

begin

tmp := split(t, key);

v := tmp.a;

result.a := t;

result.b := v;

if v = -1 then exit;

while a[v].r <> -1 do

v := a[v].r;

result.a := merge(tmp.a, tmp.b);

result.b := v;

end;

function kth (t, key : integer): integer;

begin

result := -1;

if t = -1 then exit;

result := -1;

if cnt(t) < key then exit;

if key <= cnt(a[t].l) then begin

result := kth(a[t].l, key);

exit;

end;

if key = cnt(a[t].l) + 1 then begin

result := a[t].x;

exit;

end;

result := kth(a[t].r, key - cnt(a[t].l) - 1);

end;

begin

reset(input, 'bst2.in');

rewrite(output, 'bst2.out');

Randomize;

fillchar(stack, sizeof(stack), 0);

fillchar(a, sizeof(a), -1);

r := 0;

for i:= num downto 1 do push(i);

t := -1;

while not(eof(input)) do begin

readln(s);

if s[1] = 'i' then begin

n := strtoint(copy(s, 8, 10));

t := insert(t, n);

end;

if s[1] = 'e' then begin

n := strtoint(copy(s, 8, 10));

exists(t, n);

end;

if s[1] = 'd' then begin

n := strtoint(copy(s, 8, 10));

t := delete(t, n);

end;

if s[1] = 'n' then begin

n := strtoint(copy(s, 6, 10));

tmp := next(t, n);

t := tmp.a;

if tmp.b = -1 then

writeln('none')

else

writeln(a[tmp.b].x);

end;

if s[1] = 'p' then begin

n := strtoint(copy(s, 6, 10));

tmp := prev(t, n);

t := tmp.a;

if tmp.b = -1 then

writeln('none')

else

writeln(a[tmp.b].x);

end;

if s[1] = 'k' then begin

n := strtoint(copy(s, 5, 10));

q := kth(t, n);

if q = -1 then

writeln('none')

else

writeln(q);

end;

end;

close(input);

close(output);

end.

## 

## Задача 5.

### Условие:

Дано N солдат, пронумерованных от 1 до N. Капрал любит давать приказы вида «Вперед с i по j», солтады стоящие на данном промежутке должны сохраняя порадок переместиться в начало шеренги. Дан список команд, необходимо вычислить конечное положение солдат.

### Метод решения:

Построим декартово дерево. Для каждой команды будет разрезать его на три дерева и сливать их в одно в нужном порядке.

Сложность выполнения одной команды O(logN).

### Листинг:

program c01;

{$APPTYPE CONSOLE}

uses

SysUtils;

const

num = 100000;

type

TElem = record

x, y : integer;

l, r : integer;

end;

TPair = record

a, b : integer;

end;

var

used : array [0..num] of boolean;

stack, sz : array [0..num] of integer;

r, t, i, n, q, m, aa, b : integer;

a : array [1..num] of TElem;

s : string;

tmp : TPair;

function cnt(t : integer): integer;

begin

result := 0;

if t = -1 then exit;

result := sz[t];

end;

procedure update(t : integer);

begin

if t = -1 then exit;

sz[t] := cnt(a[t].l) + cnt(a[t].r) + 1;

end;

procedure push(a : integer);

begin

inc(r);

stack[r] := a;

end;

function new : integer;

begin

result := stack[r];

dec(r);

end;

function merge (t1, t2 : integer) : integer;

begin

result := t2;

if t1 = -1 then exit;

result := t1;

if t2 = -1 then exit;

if (a[t1].y > a[t2].y) then begin

a[t1].r := merge(a[t1].r, t2);

update(t1);

result := t1;

end

else begin

a[t2].l := merge(t1, a[t2].l);

update(t2);

result := t2;

end

end;

function split (t, k : integer): TPair;

var

tmp : TPair;

begin

result.a := -1;

result.b := -1;

if t = -1 then exit;

if k > cnt(a[t].l) then begin

tmp := split(a[t].r, k - cnt(a[t].l) - 1);

a[t].r := tmp.a;

update(t);

update(tmp.b);

result.a := t;

result.b := tmp.b;

end

else begin

tmp := split(a[t].l, k);

a[t].l := tmp.b;

update(tmp.a);

update(t);

result.a := tmp.a;

result.b := t;

end;

end;

function exist (t, k: integer) : boolean;

begin

result := false;

if t = -1 then exit;

result := true;

if a[t].x = k then exit;

if a[t].x > k then begin

result := exist(a[t].l, k);

exit;

end;

if a[t].x < k then begin

result := exist(a[t].r, k);

exit;

end;

end;

function insert (t, k : integer) : integer;

var

p : integer;

tmp : TPair;

begin

result := t;

if exist(t, k) then exit;

p := new;

a[p].x := k;

a[p].y := random(100000000);

a[p].l := -1;

a[p].r := -1;

tmp := split(t, k);

tmp.a := merge(tmp.a, p);

result := merge(tmp.a, tmp.b);

end;

function delete (t, k : integer): integer;

begin

result := -1;

if t = -1 then exit;

if a[t].x < k then begin

a[t].r := Delete(a[t].r, k);

result := t;

exit;

end;

if a[t].x > k then begin

a[t].l := delete(a[t].l, k);

result := t;

exit;

end;

result := merge(a[t].l, a[t].r);

end;

procedure exists (t, k: integer);

begin

if exist(t, k) then

writeln('true')

else

writeln('false');

end;

function next (t, key : integer): TPair;

var

tmp : TPair;

v : integer;

begin

tmp := split(t, key + 1);

v := tmp.b;

result.b := v;

result.a := tmp.a;

if v = -1 then exit;

while a[v].l <> -1 do

v := a[v].l;

tmp.a := merge(tmp.a, tmp.b);

tmp.b := v;

result := tmp;

end;

function prev (t, key : integer) : TPair;

var

tmp : TPair;

v : integer;

begin

tmp := split(t, key);

v := tmp.a;

result.a := t;

result.b := v;

if v = -1 then exit;

while a[v].r <> -1 do

v := a[v].r;

result.a := merge(tmp.a, tmp.b);

result.b := v;

end;

function kth (t, key : integer): integer;

begin

result := -1;

if t = -1 then exit;

result := -1;

if cnt(t) < key then exit;

if key <= cnt(a[t].l) then begin

result := kth(a[t].l, key);

exit;

end;

if key = cnt(a[t].l) + 1 then begin

result := a[t].x;

exit;

end;

result := kth(a[t].r, key - cnt(a[t].l) - 1);

end;

procedure outputt(t, v: integer);

begin

if v = -1 then exit;

if used[v] then begin

used[v] := false;

outputt(t, a[v].l);

end;

if not(used[v]) then begin

write(v, ' ');

outputt(t, a[v].r);

end;

end;

function move(var t, l, r : integer): integer;

var

a, b : TPair;

begin

a := split(t, r);

b := split(a.a, l - 1);

t := merge(b.a, a.b);

t := merge(b.b, t);

result := t;

end;

begin

reset(input, 'movetofront.in');

rewrite(output, 'movetofront.out');

Randomize;

fillchar(stack, sizeof(stack), 0);

fillchar(a, sizeof(a), -1);

fillchar(used, sizeof(used), True);

read(n, m);

r := 0;

for i:= num downto 1 do push(i);

t := -1;

for i := 1 to n do

t := insert(t, i);

for i:=1 to m do begin

read(aa, b);

t := move(t, aa, b);

end;

outputt(t, t);

close(input);

close(output);

end.

## Задача 5.

### Условие:

### Метод решения:

### Листинг:

## Задача 5.

### Условие:

### Метод решения:

### Листинг:

## Задача 5.

### Условие:

### Метод решения:

### Листинг:

## Задача 5.

### Условие:

### Метод решения:

### Листинг:

## Задача 5.

### Условие:

### Метод решения:

### Листинг:

# Заключение