08-11-1

* Aplicația se va rezolva folosind tehnici de programare generica si liste! Fără STL*

Se considera eșantioanele unui semnal stocate a valori întregi într-o lista simplu înlănțuita.

Se definește metoda **Prelucrare** care: (a) daca 2 eșantioane succesive au diferența absoluta mai mare decât pragul P1, atunci inserează intre ele un eșantion (nod in lista) cu valoarea egala cu media aritmetica a celor 2 eșantioane; (b) daca 2 eșantioane succesive au diferența absoluta mai mica decât pragul P2, atunci al doilea eșantion (nod in lista) este eliminat.

Se repeta procesul pana când nu mai este posibila nicio modificare a semnalului.

Solutia se completeaza in fisierul **List.h**
Se completeaza toate functiile (metode) cerute. Nu se modifica numele deja existente
Daca este necesar se pot adauga alte metode/date proprii.

NU trebuie implementată functia main, nici alte functii sepatate. Strict numai clasa List, Node si mai ales functia din cerință (membră a clasei List)

13-11-1

* Aplicația se va rezolva folosind tehnici de programare generica si liste! Fără STL*

Se considera eșantioanele unui semnal stocate ca valori întregi pe 16 biti într-o lista simplu înlănțuita.

Se definește metoda **Prelucrare(int dif)** care modifica lista după cum urmează:

(a) Se împarte semnalul (lista) in ferestre (subliste) disjuncte astfel încât pentru fiecare fereastra diferența in valoare absoluta dintre fiecare eșantion si primul eșantion al ferestrei sa fie mai mica decât argumentul **dif**

(b) Pentru fiecare fereastra se va păstra **un singur nod** al listei (celelalte se vor elimina) care conține **valoarea medie** a eșantioanelor din fereastra si **numărul de eșantioane** din fereastra.

28-11-1

La un magazin clienții pun pe banda de la casa produsele din cos. Pentru fiecare client sunt cunoscute

produsele, preturile lor si momentul de timp când s-au pus produsele pe banda.

Pentru fiecare produs care este scanat se pierd 3 unități de timp.

Pentru fiecare al P-lea produs facturat se oferă un bonus de 50% din preț (trunchiat la întreg), însă

fiecare client are dreptul să aleagă cel mai scump produs pe care-l cumpăra pentru a intra la aceasta

promoție.

Fiecare al C-lea client primește o reducere fixa de D lei, dar care este limitata la costul total al produselor

din coșul clientului.

Un client poate beneficia doar de una dintre cele 2 reduceri, dintre cele 2 se alege cea mai mare. Se

contorizează numai cea care s-a acordat. Procesul de numărare produse/clienți nu este afectat (continua

ca si când s-au acordat bonusurile)

De asemenea, daca un client ar trebui sa primească mai multe bonusuri pentru fiecare al P-lea produs, el

va primi numai unul pentru produsul cel mai scump. Cele care nu se acorda nu se socotesc pentru

cerința 1.

Clienții care au produsele pe banda sunt serviți la rând, însă clienții noi pot pune produsele pe banda

numai când le sosește momentul de timp (dată de intrare). Clientul după ce pune produsele pe banda

pune un delimitator semnalat in caracterul ’T’

Daca la același moment de timp t exista un eveniment (acordare de bonus produs, client care a terminat

de plătit si pleacă) si un client nou trebuie sa pună produsele pe banda, se considera ca acesta din urma

acționează la momentul t+epsilon (epsilon tinde la zero), adică cele doua evenimente nu se

intersectează.

Programul calculează:

1. Câți clienți (CB) așteptă la banda (sa facă plata) in momentul in care se oferă bonusul de 50%

pentru al B-lea produs

2. Care este suma totala acordata pentru bonusul de 50%/produs (SB50) și care este suma totala

pentru bonusul de client (SBC) după ce au plătiți toți clienții

3. Am câtelea client (CA) a așteptat cel mai mult de când a pus produsele pe banda pana a făcut

plata (după ce i-au fost scanata si produsele sale) si care este intervalul de timp așteptat (TA).

Date de intrare, citite de la tastatura:

P, întreg = din P in P produse se primește bonus

C, întreg = din C in C clienți se primește bonus fix

D, întreg = valoarea bonusului fix /client

B, întreg = nr de produse cu bonus la care se calculează numărul de clienți care așteaptă la banda

Pentru fiecare client

T, întreg = momentul de timp (exprimat in unități de timp) la care s-au pus produsele pe banda.

Pornește de la 0 și este crescător pentru clienți succesivi

Înșiruire de perechi de valori: denumire_produs pret

denumire_produs conține un singur cuvânt care nu depășește 10 caractere

pretul este de tip întreg

'T' delimitator de produse

Date de ieșire:

CB SB50 SBC CA TA

Cu semnificația din enunț

Se vor implementa: funcția main in fișierul main.cpp si clasa Queue in fișierul Queue.h

Exemplu:

Date de intrare:

5

4

40

2

3 unt 12 carne 34 paine 8 T

5 branza 13 rosii 16 T

9 bere 12 oua 23 T

15 mere 120 pere 23 T

17 malai 8 paine 9 cas 23 T

21 suc 12 unt 16 cartofi 5 T

28 malai 9 varza 7 T

36 lapte 13 morcovi 8 unt 13 T

45 salam 20 T

Date de ieșire:

3 27 74 6 27

14-12-1

Folosind programare generică și POO, se va construi un arbore BST (fără autoechilibrare) având criteriu de ordonare a < b.

a) Se va implementa o metodă care calculează adâncimea arborelui (nodul rădăcină are adâncimea 1).
b) Se va implementa o metodă care verifică dacă arborele este echilibrat (pentru fiecare nod, diferența dintre adâncimile subarborelui stâng și cel drept în modul nu depășește 1)

16-12-1 & 16-12-2 & 16-12-3

Folosind programare generică și POO, se va construi un arbore BST (fără autoechilibrare) având criteriu de ordonare a < b cunoscând parcurgerea sa în Post Ordine.

Să se implementeze următoarele metode, fără a folosi memorie suplimentară:

1. Operatorul de indexare care returnează al k-lea cel mai mic element (k=0 pentru cel mai mic element). – **16-12-1**

2. Metodă care afișează frunzele arborelui în ordine descrescătoare – **16-12-2**

3. Metodă care primește o valoare V și returnează o pereche formată din vecinii lui V (cel mai mare element mai mic strict decât V respectiv cel mai mic element mai mare strict decât V).
Dacă unul dintre vecini nu există, valoarea returnată pentru acel vecin va fi V. Dacă valoarea V nu se găsește în arbore, atunci se întoarce o pereche de valori implicite. – **16-12-3**

14-01-1

Folosind noțiuni de POO și SDA, implementați următoarea variantă de listă simplu înlănțuită.

```
class Lista {

public:

    Lista();

    virtual ~Lista();

    void adaugaStudent(string nume, double medie);

    vector<int> returneazaIndecsiSortati();

};
```

Metoda adaugaStudent introduce un nou student în listă.

Metoda returneazaIndecsiSortati sortează toți studenții din listă descrescător după medie și în caz de egalitate, crescător după nume și apoi returnează un vector ce conține indecșii inițiali ai studenților din lista sortată.

Primul student adăugat în listă are indexul 0.

Folosirea vectorilor declarați static (T v[N];) sau dinamic (T *v = new T[N];) va conduce la acordarea punctajului 0.

Folosirea algoritmilor din STL va conduce la acordarea punctajului 0.

Cu excepția vectorului returnat, folosirea altor containere din STL va conduce la acordarea punctajului 0.

Rezolvarea cu ajutorul altor structuri (listă dublu înlănțuită, arbore dinamic, etc) va conduce la acordarea punctajului 0.

Clasa trebuie să poată construi, sorta și distruge 10 liste ce conțin sute de mii de elemente în 30 de secunde.

Se garantează faptul că numele studenților va fi unic.

Exemplu

Fie următorii studenți ce vor fi băgați în listă:
mihai 7.50
ana 9.66
marian 8.23
cristi 7.50

Metoda returneazaIndecsiSortati va returna:
[1, 2, 3, 0]

15-01-1

Fie un șir de N elemente întregi cuprinse între [0, 7]. Se construiește șirul de gradienți prin diferența dintre 2 elemente consecutive. Gradientul ultimului număr se obține prin diferența cu primul. Să se afișeze cea mai mică, din punct de vedere lexicografic, permutare circulară pentru șirul de gradienți.

Date de intrare:

Pe prima linie un număr natural N, numărul de elemente din șir. Pe a doua linie, separate prin spațiu, N numere naturale între 0 și 7 inclusiv.

Date de ieșire:

Pe prima linie, separate prin spațiu, N numere întregi, cea mai mică, din punct de vedere lexicografic, permutare circulară pentru șirul de gradienți.

Exemplu:

Intrare:

4

3 2 6 2

Ieșire:

-4 4 -1 1

Explicație:

Șirul de gradienți obținut: 1 -4 4 -1. Permutarea circulară care începe cu -4 este cea mai mică lexicografic.

Restricții și precizări:

Este permisă folosirea de structuri de date din STL. Se va afișa spațiu și după ultimul număr.

15-01-2

Fie un șir de N numere întregi cu ajutorul cărora se construiește un arbore binar de căutare simplu, fără autobalansare. Fie 2 valori întregi A, B cu A <= B. Să se afișeze dimensiunea celui mai mare subarbore care are toate elementele în intervalul [A, B]. Dacă nu există nici un subarbore care să respecte condiția, se va afișa 0.

Date de intrare:

Pe prima linie un număr natural N, numărul de elemente din șir.

Pe a doua linie, separate prin spațiu, N numere întregi.

Pe a treia linie, separate prin spațiu, 2 numere întregi A B.

Date de ieșire:

Pe prima linie dimensiunea celui mai mare subarbore care are toate elementele în intervalul [A, B].

Exemplu:

Intrare:

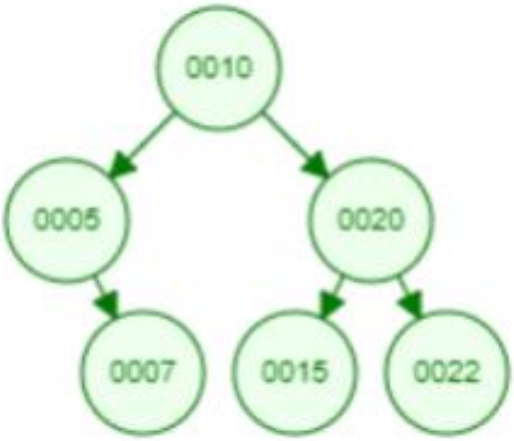
6

10 5 20 7 15 22 6 30

Ieșire:

3

Explicație: Arborele este reprezentat în următoarea figură:



Subarborele nodului 20 are toate elementele în intervalul [6, 30] și are dimensiunea 3.

Subarborele nodului 10 nu are toate elementele în intervalul [6, 30] din cauza lui 5.

28-01-1

Descriere

Folosind structuri de date înlănțuite cu alocare dinamică (fără a utiliza STL), să se rezolve următoarea problema:

Containerul stochează coordonate (X, Y) ale unor puncte din plan ce reprezintă un contur închis (ultimul punct este conectat cu primul punct). X si Y sunt valori întregi pozitive nenule pe 32 biti, citite de la tastatura până se întâlnește valoarea 0.

Se citesc de la tastatura două valori reale pozitive (de tip double) D1 si D2. Se garantează D2 > 2 * D1.

Dacă două puncte consecutive se află la distanța mai mica strict decât D1, atunci al doilea punct se elimina. Dacă două puncte consecutive se afla la distanță mai mare strict decât D2, atunci între cele 2 puncte se inserează un punct nou care se afla la mijlocul segmentul determinat de cele 2 puncte inițiale. Se repetă aceasta procedură până când cele 2 condiții (D1 și D2) sunt respectate. În final se va afișa perimetrul conturului obținut / numărul de puncte.

Ajustarea coordonatelor se face pe tip întreg de 32 biti (prin trunchiere) . Calculul distanțelor și a perimetrului se face pe tip double. Prelucrarea începe de la prima pereche (X, Y) citită și se termină atunci când toate punctele respectă cele 2 condiții.

Date de intrare

Un număr necunoscut de perechi de numere naturale X Y. Citirea se termină la primul 0. Se garantează că doar X va fi 0.

Două numere reale pozitive D1 D2.

Date de ieșire

Un număr real, scris cu 3 zecimale, perimetrul conturului obținut / numărul de puncte.

Exemplu

Date de intrare:

4 4

4 3

4 2

2 2

2 4

0

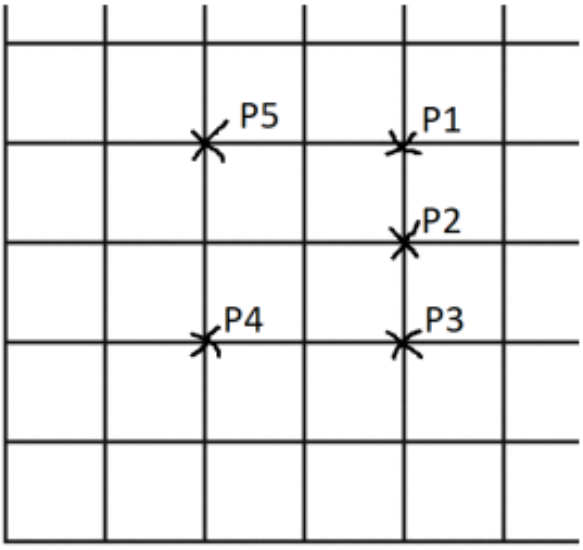
1.5 3.2

Date de ieșire:

2.000

Explicație:

Desenul următor arată punctele inițiale:



Distanța dintre P1 și P2 este de 1 < 1.5 deci P2 este eliminat. Distanța dintre P1 și P3 este 2 deci nu se întâmplă nimic. În mod similar pentru P3 P4, P4 P5, P5 P1. Perimetrul final este de 8 și sunt 4 puncte, deci răspunsul este 2.000

28-01-2

Descriere

Fie un vector ce contine N numere naturale nenule. Orice numar din vector poate fi inlocuit cu un numar mai mic decat el sau poate ramane neschimbat.

Care este valoarea maxima posibila pentru mex-ul vectorului daca se fac schimbarile in mod optimal (astfel incat sa se obtina aceasta valoare maxima posibila).

Se numeste mex-ul unui vector cea mai mica valoare care nu face parte din acel vector.

Date de intrare

Pe prima linie un numar natural nenul N

Pe urmatoarea linie, separate prin spatiu, N numere naturale nenule, valorile din vector.

Date de iesire

Pe prima linie un numar natural nenul, mex-ul vectorului atunci cand se fac schimbarile in mod optimal.

Exemplu

Date de intrare:

5

1 3 3 3 6

Date de iesire:

5

Explicatie:

Se inlocuieste primul 3 cu 2 care este mai mic decat el. Celelalte valori de 3 raman neschimbate. Se inlocuieste 6 cu 4 care este mai mic decat el. 5 devine prima valoare ce nu face parte din vector.

Orice alte variante nu vor conduce la un mex mai mare.

28-01-3

Să se implementeze un arbore multicăi, în care nu se cunoști numărul maxim de descendenți ai fiecărui nod. Implementarea este cu alocare dinamica, optimală (nu se aloca spațiu suplimentar față de numărul curent de noduri), folosind una dintre abordările descrise în curs.

Datele stocate în fiecare nod al arborelui sunt valori întregi pozitive, oarecare. Prima valoarea citită este a rădăcinii, iar apoi se vor citi perechi tata fiu (tatăl exista deja inserat in arbore). Fiii unui nod se inserează (adaugă) în ordinea în care apar în datele de intrare, de la stânga la dreapta.

Valorile nodurilor nu se repetă. Construcția arborelui se face strict pe baza relației tata-fiu (fără vreo relație de ordonare).

Construcția arboretului se termina atunci când valoare pentru tată este 0.

Se citesc, de pe rândul următor, două valori, se va determina și afișa strămoșul comun cel mai apropiat (cel mai tânăr) al nodurilor care conțin cele doua valorii. Se garantează că cele 2 valori exista in arbore.

Exemplu:

Date de intrare:

100

100 50

100 20

50 30

100 70

20 10

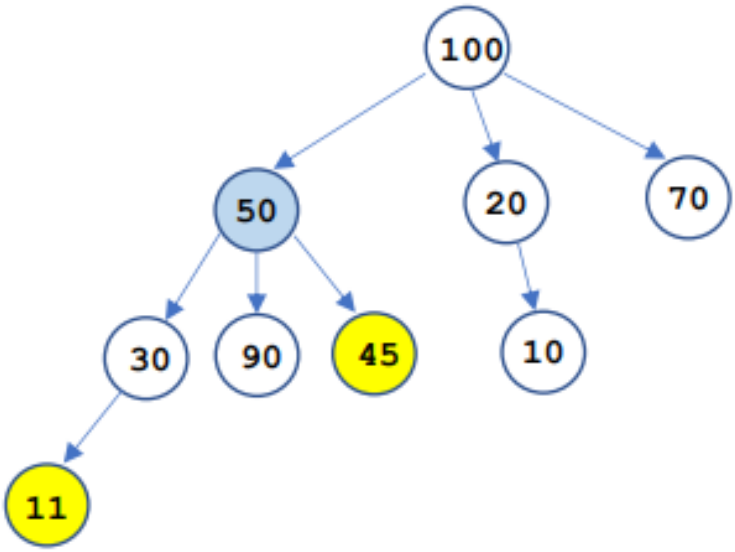
50 45

30 11

50 90

0

11 45



Date de ieșire:

50