Căutări/inserări în structuri de date – complexități

1. **Căutare secvențială într-un vector memorat ca**
   - masiv/tablou
   - vector stl
2. **Căutare binară într-un vector ordonat memorat ca**
   - masiv/tablou
   - vector stl
3. **Căutare în alte structuri de date stl**
   - **set (mulțime ordonată – arbore de căutare)**
   - **unordered_set (hash-table, cheie=valoare)**
   - **unordered_map (hash)**

A. PROBLEME 2-SUM și 3-SUM

Se dă un vector $v = (v_1, ..., v_n)$ cu n elemente.
Numim **pereche** de elemente din v o pereche (v_i,v_j) cu $i \neq j$.
Numim **triplet** de elemente din v un triplet (v_i, v_j, v_k) cu i, j, k –distincte două câte două.
Două **perechi** de elemente din v: (v_i,v_j) și (v_r,v_s) se numesc **distincte** dacă $\{v_i, v_j\} \neq \{v_r, v_s\}$.
Două **triplete** de elemente din v: (v_i,v_j, v_k) și (v_r,v_s,v_t) se numesc **distincte** dacă $\{v_i, v_j, v_k\} \neq \{v_r, v_s, v_t\}$

Problema 1. 2-SUM

a) Știind că elementele lui v sunt **ordonate crescător**, propuneți un algoritm eficient (timp+memorie) care să afișeze perechile **distincte** de elemente din v care au suma 0. Ce complexitate are algoritmul propus?

date.in	date.out
11 -4 -4 -2 0 0 0 0 1 2 4 6	-4 4 -2 2 0 0 nu neapărat în această ordine.

b) Propuneți un algoritm eficient (timp) care să determine dacă există în v cel puțin o pereche de elemente cu suma 0 și, în caz afirmativ, afișează o astfel de pereche (elementele lui v nu sunt neapărat ordonate crescător). Ce complexitate are algoritmul propus?

Problema 2. 3-SUM

Aceleași cerințe ca la 2-SUM, dar pentru triplete în loc de perechi

date.in	date.out
12 -4 -4 0 0 0 0 1 2 2 2 3 4	-4 0 4 -4 2 2 -4 1 3 0 0 0 nu neapărat în această ordine.

B. TABELE HASH (DE DISPERSIE); unordered_map

http://www.cplusplus.com/reference/unordered_map/unordered_map/ - studiați metodele principale (insert, find, erase, at, operatorul [], parcurgerea cu iterator)

Exemplu: Dat un fișier „fisier_numere.txt” cu numere reale separate prin spații, să se afișeze fiecare numerele distincte din fișier și frecvența (numărul de apariții) fiecărui astfel de număr.

```
#include<iostream> #include<fstream>
#include<unordered_map>
using namespace std;
int main(){
  unordered_map<float,int> um;

  float n;
  ifstream f("fisier_numere.txt");

  while(f>>n){
      unordered_map<float, int>::iterator iter= um.find(n);
      //auto iter = um.find(s);

      if(iter == um.end())
           um.insert({n,1});//um.insert(make_pair(n,1));
       else
           iter->second++;
  }
  f.close();

  //afisare cu iterator
  for (unordered_map<float, int>::iterator it = um.begin();
                                        it != um.end(); ++it)
  //for (auto it = um.begin(); it != um.end(); ++it)
         cout << it->first <<" : "<< it->second << endl;
  cout<<endl;

  //afisare cu for each
  for(pair<float,int> elem:um) //for (auto elem:ums)
       cout << elem.first <<" : "<< elem.second << endl;
}
```

```
while(f>>n){
       um[n]++;
}
```

Pentru a afișa frecvența unui număr x citit de la tastatură în fișierul „fisier_numere.txt”:

```
unordered_map<float, int>::iterator iter
= um.find(x);

if(iter!=um.end())
       cout<<iter->second<<endl;
else
       cout<<"nu apare in fisier"<<endl;
```

```
try{
    cout<<um.at(x)<<endl;
}
catch(out_of_range  oor) {
  cout<<"nu apare in fisier"<<endl;
}
```

```
cout<<um[x];
//cum functioneaza?
```

Exercițiu - Distanța (similitudinea) între două documente

Fie două documente text, F_1 și F_2, și $\{c_1, c_2, \dots, c_n\}$ mulțimea cuvintelor care apar în cel puțin unul din cele două documente. Pentru $1 \leq i \leq n$, fie v_{i1}, v_{i2} numărul de apariții al cuvântului i în primul, respectiv în al doilea document. Distanța cosinus dintre cele două documente, notată $dcos(F_1, F_2)$, dintre F_1 și F_2 se calculează după formula:

$$dcos(F_1, F_2) = \frac{\sum_{i=1}^{n} v_{i1} v_{i2}}{\sqrt{\sum_{i=1}^{n} v_{i1}^2} \sqrt{\sum_{i=1}^{n} v_{i2}^2}}$$

Fiind date două fișiere text, *F1* și *F2*, să se calculeze distanța cosinus dintre cele două fișiere. Fiecare dintre cele două fișiere va fi parcurs o singură dată. Cuvintele dintr-un fișier sunt separate prin spații, virgulă, punct.

Exemplu: dacă F_1 conține textul: *primul laborator primul exercitiu* și F_2 conține textul: *primul exercitiu usor*

$\{c_1 =' primul', c_2 = 'laborator', c_3 = 'exercitiu'\ c_4 = 'usor'\}$

$\{v_{11} = 2,\ v_{21} = 1,\ v_{31} = 1,\ v_{41} = 0\}$

$\{v_{12} = 1,\ v_{22} = 0,\ v_{32} = 1,\ v_{42} = 1\}$

$$dcos(F_1, F_2) = \frac{2 * 1 + 1 * 0 + 1 * 1 + 0 * 1}{\sqrt{6}\sqrt{3}} = 0{,}7071$$

Bibliografie:

- https://en.wikipedia.org/wiki/Cosine_similarity