Vector:

- Szigorú sorrend az elemek között
- Majdnem mindig egymás mellett vannak az elemek a memóriában (tömbben mindig)
- Létezik benne a pozíció fogalma: [] operátor
- Lineáris keresés
- Nagyon rossz az új elem beszúrása vector közepébe

Set:

- Növekvő sorrend az elemek között (emiatt definiálva kell lennie a < operátornak)
- Keresőfába rendezett elemek
- Csak iterátorral iterálható
- Egy elem csak egyszer szerepelhet
- Logaritmikus keresés (.find() függvény, ami iterátorral tér vissza)
- Nincs pozíció jól definiálva, nincs [] operátor
- Új elem könnyen beilleszthető (.insert() függvény)

Map:

- set<pair<>> hasznos extra tulajdonságokkal
- a pair első eleme a kulcs
- a pair második eleme az érték
- a kulcsok találhatóak a set-ben (emiatt egyedinek kell lenniük)
- az értékeknek nem kell egyedinek lenniük
- [] operátorral rakható be új elem
- [] operátorral kershető egy kulcshoz tartozó érték is (óvatosan, mert ha nincs benne, létre fog jönni)
- A matematikai függvény fogalmához rendkívül jól illeszkedik (minden x-hez csak érték tartozik), gyakran használják is ilyen szerepkörben

---

Szám létezésének keresése vectorban és setben.

- generáljunk egy n (mondjuk 1000000) hosszú random számokból álló vektort és setet (0-tól n-1-ig random számok, a vektorba és a setbe ugyanazok a számok kerüljenek)
- vizsgáljuk meg mind a vektorra mind a setre hogy benne van egy adott szám
- mérjük le a keresés idejét millisec-ben:
  ```
  #include <chrono>
  auto kezdes = std::chrono::high_resolution_clock::now();
  auto vege = std::chrono::high_resolution_clock::now();
  cout << std::chrono::duration_cast<std::chrono::milliseconds>(vege-kezdes).count() << endl;
  ```

Recept tárolása map<string,float>-ban

- Hogyan oldottátok volna meg ezt a feladatot vector<> segítségével?
- A recept gyakorlatilag egy mennyiség (mondjuk tömeg) értékű alapanyag típusú függvény (miből mennyi kell?)
- Rakjunk bele pár elemet a mapbe
- Keressük meg az egyik alapanyaghoz, hogy mennyi kell belőle az [] operátor segítségével