2-MAVZU

MA’LUMOT TUZILMALARI VA STL

1. Nazariy qism

1. Ma’lumot tuzilmasi nima?

Ma’lumot tuzilmasi — bu:

- ma’lumotlarni qanday saqlash,
- qanday tartibda joylashtirish,
- va qanday tez ishlashni belgilovchi usul.

Olimpiadada savol ko‘pincha shunaqa bo‘ladi:

“Qanday tuzilma tanlasam, masala tez va optimal yechiladi?”

2. Oddiy massiv (array)

Ta’rif

Massiv — bir xil turdagi elementlarning ketma-ket xotirada saqlanishi.

```
int a[1000];
```

Xususiyatlari:

- indekslar `0` dan boshlanadi
- o‘lchami oldindan ma’lum
- juda tez (`O(1)` murojaat)

Kamchiligi:

- o‘lchamni keyin o‘zgartirib bo‘lmaydi
- qo‘shish/o‘chirish noqulay

Olimpiadada:

- kichik va o‘rtacha hajm → massiv
- dinamik hajm → `vector`

3. Ikki o‘lchamli massiv

```
int a[100][100];
```

Ishlatiladi:

- matritsa
- jadval
- grafning qo‘shnichilik matritsasi

Xato manbai:

- qator va ustunni adashtirish
- chegaradan chiqib ketish

4. `string` — satrlar bilan ishlash

```
string s;
cin >> s;
```

Afzalligi:

- uzunligi avtomatik
- indeks bilan ishlash mumkin
- `+`, `==`, `.size()` mavjud

`char[]` ga qaraganda:

- xavfsizroq
- qulayroq

5. `struct` — murakkab obyekt

Bir nechta bog‘liq ma’lumotni bitta tuzilmaga joylash.

```
struct Student {
    string name;
    int age;
    int score;
};
```

Olimpiadada: juftlik/trio ma’lumotlar, saralash (`sort`) bilan birga juda foydali

6. STL nima?

STL (Standard Template Library) — C++ ning tayyor:

- konteynerlari
- algoritmlari
- iteratorlari

Olimpiadada STL bilmaslik — katta minus.

7. `vector` — eng muhim konteyner

```
vector<int> v;
v.push_back(5);
```

Afzalliklari:

- o‘lchami dinamik
- massiv kabi tez
- oxiridan qo‘shish `O(1)`

Asosiy metodlar:

- `push_back()`
- `pop_back()`
- `size()`
- `clear()`

Olimpiada qoidasi:
Massiv o‘rniga `vector` ishlatish odatiy hol.

8. `pair` — ikki qiymatli tuzilma

```
pair<int, int> p = {3, 5};
```

Foydalaniladi:

- koordinata
- qiymat + indeks
- vaqt + hodisa

9. `map` — kalit–qiymat tuzilmasi

```
map<int, int> mp;
mp[5] = 10;
```

Xususiyatlari:

- kalitlar tartiblangan
- qidirish `O(log n)`

Qachon ishlatiladi?

- chastota hisoblash
- indekslash
- moslik (mapping)

10. `set` — noyob elementlar

```
set<int> s;
s.insert(5);
```

Xususiyatlari:

- takroriy element yo‘q
- avtomatik tartiblangan

Ko‘p ishlatiladigan joy:

takrorlarni yo‘qotish

eng kichik/katta elementni tez olish

11. `queue` — navbat (FIFO)

```
queue<int> q;
q.push(1);
q.pop();
```

BFS algoritmining asosi.

12. `stack` — stek (LIFO)

```
stack<int> st;
```

Qavslar, rekursiya, orqaga qaytish.

13. `deque` — ikki tomondan ishlov

```
deque<int> d;
```

- oldidan ham
- oxiridan ham
  qo‘shish/o‘chirish mumkin.

14. `priority_queue` — ustuvor navbat

```
priority_queue<int> pq;
```

- doim eng katta element ustida ishlaydi

- `max-heap`

Agar eng kichik kerak bo‘lsa:

```
priority_queue<int, vector<int>, greater<int>> pq;
```

15. `algorithm` kutubxonasi — YADRO

```
sort(v.begin(), v.end());
```

Eng muhimlari:

- `sort`
- `reverse`
- `min`, `max`
- `count`
- `binary_search`

Ko‘p masalalar faqat `sort` bilan yechiladi.

16. Iterator tushunchasi (asosiy)

Iterator — konteyner bo‘ylab yurish vositasi.

```
for (auto it = v.begin(); it != v.end(); it++)
```

`auto` — kodni soddalashtiradi.

2-MAVZU

2. METODIK YORDAM (kengaytirilgan)

1. “Avval muammo → keyin tuzilma” yondashuvi

Eng katta metodik xato — STL ni alohida o‘rgatish.

To‘g‘ri yondashuv:

- Masalani berish
- “Bu yerda nima sekin bo‘ladi?” degan savol
- Shundan keyin to‘g‘ri konteyner tanlash

Masalan:

- Takrorlarni yo‘qotish → `set`
- Tartiblash kerak → `vector + sort`
- Eng katta qiymatni tez topish → `priority_queue`

2. Massiv → `vector` o‘tishini ongli qilish

O‘quvchilar ko‘pincha:

```
int a[100000];
```

ni har joyda ishlatib ketadi.

Metodik tavsiya:

- Avval massiv bilan masala yechiladi
- Keyin xuddi shu masala `vector` bilan qayta yoziladi
- Farqi muhokama qilinadi

Natija: o‘quvchi qachon qaysi biri kerakligini sezadi.

3. `vector` — “standart konteyner” sifatida mustahkamlash

Deyarli barcha boshlang‘ich va o‘rta olimpiada masalalarida:

`vector` = default tanlov

Tavsiya:

- 2-mavzu davomida massivni faqat majbur bo‘lsa ishlatish
- Boshqa hollarda doim `vector`

4. `pair` va `struct` ni taqqoslab berish

Ko‘p o‘quvchi bu ikkalasini chalkashtiradi.

Metodik izoh:

- 2 ta qiymat → `pair`
- 3+ qiymat yoki nomli maydon → `struct`

Mashq:

- Avval `pair<int,int>` bilan yechim
- Keyin `struct` bilan qayta yozish

5. `map` va `set` ni “sekin, lekin qulay” deb tushuntirish

O‘quvchilar `map` ni:

“har joyda ishlatsa bo‘ladi”
deb o‘ylaydi — xato.

To‘g‘ri tushuncha:

- `map` → qulay, lekin `O(log n)`
- `vector` → tez, lekin qo‘lda boshqariladi

Metodik qoida:

Agar indeks ketma-ket bo‘lsa → `vector`
Agar kalit ixtiyoriy bo‘lsa → `map`

6. Navbat va stekni hayotiy misol bilan berish

- `queue` → navbat (bank, kassir)
- `stack` → tarelka ustma-ust

Bu tushuncha:

- BFS
- rekursiya
- qavslar masalasida juda muhim

7. `priority_queue` ni kechroq, lekin kuchli berish

Bu konteyner:

- yangi boshlovchi uchun qiyin
- lekin olimpiadada juda kuchli qurol

Metodik tavsiya:

- Avval `max` ni qo‘lda topish
- Keyin `priority_queue` bilan 2 qator kodda yechish

8. `algorithm` — “kodni qisqartirish san’ati”

Ko‘p o‘quvchi `sort` ni biladi, lekin:

- `count`
- `reverse`
- `binary_search`
  ni ishlatmaydi.

Tavsiya:

- Har bir algoritmni siklli yechim bilan solishtirish
- Qaysi biri:
  - qisqaroq
  - tushunarliroq
  - kam xatoli

9. Iteratorni majbur qilmaslik

Boshlanishda:

```
for (int i = 0; i < v.size(); i++)
```

yetarli.

Keyin asta-sekin:

```
for (auto x : v)
```

Bu psixologik yengillik beradi.

10. 2-mavzuda maqsadni aniq aytish

Bu mavzudan kutiladigan natija:

“Hamma STL ni bilaman” emas
“Masalaga qarab to‘g‘ri konteyner tanlay olaman”

3. TIPIK XATOLAR (kengaytirilgan va tahlilli)

1. `vector` o‘lchamidan tashqariga chiqish

```
vector<int> v(5);
v[5] = 10; // XATO
```

Indekslar: `0` dan `size()-1` gacha.

2. `map` ni massiv o‘rnida ishlatish

```
map<int,int> mp;
for (int i = 0; i < n; i++) mp[i] = 0;
```

Juda sekin ishlaydi.

3. `set` da indeks bilan murojaat qilishga urinish

```
s[0]; // XATO
```

`set` — indeksli emas.

4. `priority_queue` ning yo‘nalishini tushunmaslik

```
priority_queue<int> pq; // max-heap
```

O‘quvchi min chiqadi deb o‘ylaydi — xato.

5. `sort`dan oldin `#include <algorithm>` ni unutish

Eng ko‘p uchraydigan kompilyatsiya xatosi.

6. `pair` ichidagi `.first`, `.second` ni adashtirish

```
p.second = x; // aslida first kerak bo‘lgan
```

Nom yo‘qligi sababli tez-tez xato qilinadi.

7. `map` elementini tekshirmasdan o‘qish

```
if (mp[x] == 0)
```

Bu kod yangi element qo‘shib yuboradi.

To‘g‘risi:

```
if (mp.count(x))
```

8. `vector` ni uzatishda nusxa ko‘paytirish

```
void f(vector<int> v) // sekin
```

To‘g‘risi:

```
void f(const vector<int>& v)
```

9. STL ni haddan ortiq ishlatish

Oddiy masala → murakkab STL.

Olimpiada qoidasi:

Eng oddiy va eng tez yechim — eng yaxshi yechim.

10. Konteynerni tozalashni unutish

Bir nechta test bo‘lsa:

```
v.clear();
mp.clear();
```

unutiladi — noto‘g‘ri javob.

METODIK XULOSA

Agar o‘quvchi:

- 🕮 muammo → tuzilma → algoritm
- 🕮 sekin yechim → tez yechim
- 🕮 qo‘lda → STL

ketma-ketligini tushunsa, STL haqiqiy qurolga aylanadi, shunchaki “bilim” bo‘lib qolmaydi.