JAVA — 2-BO‘LIM

MA’LUMOT TUZILMALARI VA ALGORITMLAR (JAVA)

1. Nazariy qism

1. Nega aynan 2-bo‘lim hal qiluvchi?

1-bo‘lim:

- sintaksis
- to‘g‘ri yozish

2-bo‘lim:

- to‘g‘ri fikrlash
- to‘g‘ri algoritm tanlash
- olimpiada darajasiga chiqish

Ko‘p WA/TLE sababi:

noto‘g‘ri ma’lumot tuzilmasi tanlash

2. Java Collection Framework — umumiy manzara

Asosiy interfeyslar:

- `List`
- `Set`
- `Map`
- `Queue`

Qoidasi:

Har vazifa → mos Collection

3. `ArrayList` vs `LinkedList`

`ArrayList`

- indeks orqali tez (`O(1)`)
- oxiriga qo‘shish tez
- o‘rtadan o‘chirish sekin

`LinkedList`

- ko‘rsatkichlar asosida
- o‘rtadan o‘chirish tez
- indeks bilan sekin

Olimpiadada:

`ArrayList` deyarli har doim yutadi

4. `Stack`, `Queue`, `Deque`

Stack (LIFO)

```
Stack<Integer> st = new Stack<>();
```

Amalda kam ishlatiladi.

Queue (FIFO)

```
Queue<Integer> q = new ArrayDeque<>();
```

Deque (ikki tomondan)

```
Deque<Integer> dq = new ArrayDeque<>();
```

BFS, sliding window → Deque

5. `HashSet` va `TreeSet`

`HashSet`

- tartibsiz
- `O(1)` o‘rtacha

`TreeSet`

saralangan

`O(log n)`

Qoidasi:

faqat mavjudlik → `HashSet`

tartib kerak → `TreeSet`

6. `HashMap` va `TreeMap`

`HashMap`

tez

tartib yo‘q

`TreeMap`

kalitlar saralangan

`O(log n)`

Eng ko‘p ishlatiladigan:

`HashMap`

7. Saralash (Sorting) Java’da

Oddiy saralash:

```
Arrays.sort(a);
```

Custom saralash:

```
Arrays.sort(a, (x, y) -> x - y);
```

Murakkablik: `O(n log n)` (TimSort)

8. Binary Search

Tayyor usul:

```
Arrays.binarySearch(a, x);
```

Shart: massiv saralangan bo‘lishi shart

9. Two Pointers texnikasi

Ikki ko‘rsatkich: `l`, `r`

Ishlatiladi:

- juftlik yig‘indisi
- noyob bo‘lak
- oraliq shartlar

Murakkablik: `O(n)`

10. Sliding Window

Two pointers’ning maxsus holati.

Masalalar:

- eng uzun/kichik bo‘lak
- chastota bilan ishlash

Afzalligi:

`O(n)` — ichma-ich sikl yo‘q

11. Prefix Sum

```
pref[i] = pref[i-1] + a[i];
```

Oraliq yig‘indi:

```
pref[r] - pref[l-1]
```

Ko‘p so‘rov → majburiy texnika

12. Greedy algoritmlar

G‘oya:

Har qadamda eng yaxshi tanlov

Misollar:

vazifalarni tanlash

minimal vaqt

Eslatma:

doim ishlamaydi

qarshi misol tekshiriladi

13. Rekursiya (asoslar)

```
static int f(int n) {
    if (n == 0) return 1;
```

```
    return n * f(n - 1);
}
```

Javada: chuqur rekursiya → `StackOverflowError`

14. Dynamic Programming (asoslar)

Asosiy bosqichlar:
Holat (`dp[i]`)
O‘tish formulasi
Boshlang‘ich holat

2-bo‘limda:

- 1D DP
- oddiy 2D DP

15. Algoritm tanlash jadvali

Vazifa	Yechim
Chastota	HashMap
Noyoblik	HashSet
Saralash	Arrays.sort
Qidirish	Binary Search
Uzluksiz bo‘lak	Sliding Window
Oraliq so‘rov	Prefix Sum
Optimal tanlov	Greedy

2-BO‘LIM NATIJASI

Bu nazariyani puxta bilgan o‘quvchi:

- Java’da to‘g‘ri DS tanlaydi
- $O(n^2)$ o‘rniga O(n) yozadi
- Olimpiada masalalarini tahlil qila boshlaydi

METODIK YORDAM

1. “Masala → DS → Algoritm” refleksi

Metodik qoida:

Har masalada birinchi savol:
Qaysi ma’lumot tuzilmasi eng mos?

Misollar:

- “necha marta uchradi?” → `HashMap`
- “tartiblangan holda kerak” → `TreeMap`
- “faqat bor-yo‘qligi” → `HashSet`

2. Collection tanlashda 3 ta asosiy mezon

Tezlik (O(1), O(log n))
Tartib kerakmi?
Indeks bilan ishlash kerakmi?

Shu 3 savolga javob → to‘g‘ri tanlov.

3. `ArrayList`ni standart sifatida qabul qilish

Metodik urg‘u:

- `LinkedList` — kamdan-kam
- `ArrayList` — 90% masala

O‘quvchiga:

“Alohida sabab bo‘lmasa — `ArrayList`”

4. `Deque` — BFS va Sliding Window yuragi

Metodik qoida:

- `Queue` emas
- `Deque` ishlat

```
Deque<Integer> dq = new ArrayDeque<>();
```

`LinkedList` → sekinroq.

5. `HashMap` bilan ishlash madaniyati

To‘g‘ri usul:

```
map.put(x, map.getOrDefault(x, 0) + 1);
```

`containsKey + get` — keraksiz ikki marta tekshiruv.

6. Saralashdan keyin — ikki pointer

Metodik formula:

`sort + two pointers` → kuchli kombinatsiya

Juftlik yig‘indisi, 3SUM va h.k.

7. Prefix sum’ni alohida massivda saqlash

Metodik qoida:

har so‘rovda qayta hisoblama

oldindan hisoblab qo‘y

Juda ko‘p TLE sababi shu.

8. Sliding window’da chastotani nazorat qilish

Metodik yondashuv:

- `HashMap` + `l, r`
- har element kiradi/chiqadi → `O(n)`

Ichma-ich sikl yo‘q.

9. Greedy’da qarshi misol qidirish

Metodik savol:

“Agar eng yaqin bo‘lmaganini tanlasam nima bo‘ladi?”

Greedy isbot talab qiladi.

10. DP’ni kichikdan boshlash

Metodik tavsiya:

- avval `dp[0]`, `dp[1]`
- keyin umumiy holat

2-bo‘lim DP → asosiy tushuncha.

3. TIPIK XATOLAR (eng ko‘p uchraydigan)

1. Noto‘g‘ri DS tanlash

Masala sekinlashadi yoki WA.

2. `LinkedList`ni default ishlatish

Java’da sekin.

3. `HashMap`ni noto‘g‘ri yangilash

```
map.put(x, map.get(x) + 1); // NullPointerException
```

4. Saralamasdan binary search

Natija noto‘g‘ri.

5. Sliding window’da `l` ni unutish

Cheksiz sikl xavfi.

6. Prefix sum indekslarini adashtirish

`pref[l-1]` unutib yuboriladi.

7. Greedy’ni isbotsiz qo‘llash

Qarshi misolda yiqiladi.

8. Rekursiyani haddan oshirish

`StackOverflowError`.

9. DP massivini noto‘g‘ri o‘lchash

`ArrayIndexOutOfBoundsException`.

10. Java sekin degan noto‘g‘ri xulosa

Asl sabab:

noto‘g‘ri algoritm + noto‘g‘ri DS

METODIK XULOSA (2-BO‘LIM)

Agar o‘quvchi:

- DS tanlash refleksini hosil qilsa
- $O(n^2)$ dan qochsa
- tayyor Java Collection’dan foydalansa

2-BO‘LIM olimpiada fikrlashini shakllantiradi.