JAVA — 3-BO‘LIM

ADVANCED ALGORITMLAR VA GRAF

2. Nazariy qism (to‘liq)

1. 3-bo‘lim nimani anglatadi?

1–2-bo‘limlar:

- Java sintaksisi
- DS va oddiy algoritmlar

3-bo‘lim:

- olimpiada va ICPC darajasi
- murakkab graf fikrlashi
- algoritmlar kombinatsiyasi

Bu bo‘limdan keyin:

Java — to‘liq olimpiada quroli

2. Graf asoslari (qisqa, lekin muhim)

Graf = tugunlar + qirralar

Turlari:

- yo‘naltirilmagan / yo‘naltirilgan
- og‘irliksiz / og‘irlikli

Java’da ifodalash:

```
ArrayList<int[]>[] g = new ArrayList[n];
```

Matritsa emas — ro‘yxat (adj list).

3. BFS va DFS (chuqur farq)

BFS

- qatlamma-qatlam
- eng qisqa yo‘l (og‘irliksiz)

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

DFS

- chuqurlashib boradi
- sikl, komponent, topologiya

Java’da DFS:

- rekursiya yoki
- stack bilan

4. Dijkstra algoritmi (eng muhim)

Qachon? og‘irliklar manfiy emas

Tuzilma: `PriorityQueue`

```
PriorityQueue<long[]> pq = new PriorityQueue<>(
    Comparator.comparingLong(a -> a[0])
);
```

Qoidasi:

```
if (d > dist[u]) continue;
```

5. Bellman–Ford (chegaraviy holat)

Qachon?

- manfiy og‘irlik bo‘lsa
- manfiy sikl tekshirish

Sekin: `O(n·m)`
Kam ishlatiladi, lekin bilish shart.

6. Union–Find (DSU)

Vazifasi:

- komponentlarni birlashtirish
- siklni aniqlash

```
find(x), union(x,y)
```

Path compression → majburiy.

7. Minimal Spanning Tree (MST)

Kruskal algoritmi:

Qirralarni saralash
DSU bilan birlashtirish

Murakkablik: `O(m log m)`

8. Topological Sort

Qachon? DAG (siklsiz yo‘naltirilgan graf)

Usullar:

- DFS
- Kahn (BFS + indegree)

Rejalashtirish masalalari.

9. Strongly Connected Components (SCC)

Algoritmlar:

- Kosaraju
- Tarjan

SCC → grafni siqish
Natija → DAG

10. Shortest Path in DAG

Graf DAG bo‘lsa:

- topological sort
- DP bilan eng qisqa yo‘l

Dijkstra’dan ham tez.

11. Segment Tree (asoslar)

Qachon?

- oraliq min/max/sum
- tez yangilash

Murakkablik: `O(log n)` / so‘rov

Java’da ehtiyot bilan yoziladi.

12. Fenwick Tree (BIT)

Segment Tree’ning yengil varianti:

- prefix sum
- `O(log n)`

Java’da juda qulay.

13. Advanced Dynamic Programming

Misollar:

- LIS (`O(n log n)`)
- knapsack (asos)
- bitmask DP

3-bo‘limda DP → chuqurlashadi.

14. Kombinatsiyalar

Olimpiada darajasi:

- graf + DP
- graf + DSU
- Dijkstra + state

3-bo‘limning yuragi.

15. Java optimalligi (3-bo‘limda)

Majburiy odatlar:

- `FastScanner` / `BufferedReader`
- `StringBuilder`
- `PriorityQueue`
- rekursiyada ehtiyotkorlik

Java sekin emas — to‘g‘ri yozilsa juda tez.

2. METODIK YORDAM (kengaytirilgan)

1. “Model → Algoritm → DS” ketma-ketligi

Metodik oltin qoida:
Masalani modelga keltir (graf/daraxt/holatlar).
Algoritm tanla (BFS/Dijkstra/DSU/DP).
Ma’lumot tuzilmasi bilan mustahkamla (`PriorityQueue`, `ArrayList`, DSU).

Algoritmni oldindan aytib bo‘lmaydi — model aytadi.

2. Grafni to‘g‘ri ifodalash madaniyati

- Har doim adjacency list.
- Og‘irlik bo‘lsa: `(to, w)` juftlik.
- Yo‘naltirilgan/yo‘naltirilmaganini aniq belgila.

```
ArrayList<int[]>[] g = new ArrayList[n];
```

Matritsa — faqat juda kichik `n` da.

3. BFS/DFS tanlash refleksi

- Og‘irliksiz eng qisqa yo‘l → BFS
- Sikl, komponent, tartib → DFS

DFS rekursiya bo‘lsa — chuqurlikni tekshir; aks holda stack bilan yoz.

4. Dijkstra’ni “to‘g‘ri” o‘rgatish

Metodik shablon:

- `PriorityQueue` (min-heap)
- Eskirgan holatni tashlab yuborish sharti

```
if (d > dist[u]) continue;
```

Bu qator yo‘q bo‘lsa — TLE/WA.

5. Bellman–Ford’ni chegaraviy holat sifatida ko‘rish

- Manfiy og‘irlik → Bellman–Ford
- Manfiy sikl → `n-1` relax + tekshiruv

Sekin, lekin to‘g‘ri.

6. DSU’ni refleks darajasiga chiqarish

Metodik mashqlar:

- Sikl bormi?
- Nechta komponent?
- MST qurish

`find()` → path compression majburiy.

7. MST: Kruskal’ni avtomatik ishlatish

Metodik ketma-ketlik:
Qirralarni saralash
DSU bilan birlashtirish
Og‘irlikni yig‘ish

Agar qirralar saralanmasa — Kruskal ishlamaydi.

8. Topological sort’ni real hayot bilan bog‘lash

Metodik misollar:

- fanlar prerequisiti
- ishlar ketma-ketligi
- build/deploy tartibi

DAG sharti alohida urg‘u bilan beriladi.

9. SCC’ni “siqish” g‘oyasi bilan o‘rgatish

Metodik yondashuv:

- SCC → super-tugun
- SCC’lar orasida → DAG

Murakkab graf masalalarining kaliti.

10. DAG’da eng qisqa/uzun yo‘lni DP bilan yechish

Metodik formula:
Topological sort
`dp[u]` dan `dp[v]` ga o‘tish

Dijkstra shart emas — tezroq va aniqroq.

11. Segment Tree vs Fenwick Tree

Metodik tanlov jadvali:

Vazifa	Tuzilma
Prefix sum	Fenwick
Range min/max + update	Segment Tree

Java’da: Fenwick — ustun.

12. Advanced DP’ni bosqichlab berish

Metodik yo‘l:
Oddiy DP (1D)
O‘tish optimallashtirish
`O(n log n)` DP (masalan, LIS)

Birdan murakkab DP bermaslik.

13. Algoritmlar kombinatsiyasini ajratib o‘rgatish

Metodik tartib:

- Graf + BFS
- Graf + heap (Dijkstra)
- Graf + DSU (MST)
- Graf + DP (DAG)

Bir masalada hammasini aralashtirmaslik.

14. Java optimalligi (3-bo‘limda majburiy)

Metodik talablar:

- `BufferedReader` / `FastScanner`
- `StringBuilder`
- `PriorityQueue`
- Katta massivlarni global e’lon qilish

Bu bo‘limda har millisekund qimmat.

15. Isbot va qarshi misol madaniyati

Metodik savollar:

- Nega bu algoritm ishlaydi?
- Qachon ishlamaydi?

Ayniqsa:

- Greedy
- Topological sort
- Dijkstra

METODIK XULOSA (3-BO‘LIM)

Agar o‘quvchi:

- masalani to‘g‘ri modellashtirsa
- algoritmni joyida tanlasa
- Java optimalligiga amal qilsa

u holda JAVA — to‘liq olimpiada quroliga aylanadi.

3. TIPIK XATOLAR (ADVANCED GRAF + ALGORITMLAR)

1. Grafni noto‘g‘ri modellashtirish

Xato:
Masala graf emas, deb o‘ylash (yoki aksincha).

Misol: vaqtlar, bog‘lanishlar, o‘tishlar → graf

Har masalada savol: “Bu tugunlar va o‘tishlarmi?”

2. Adjacency matrix ishlatish

Xato:`int[][] g = new int[n][n];` $n \geq 10^5$ bo‘lsa — xotira portlaydi.

Har doim: `ArrayList<int[]>[] g`

3. BFS o‘rniga DFS ishlatish

Xato:
Og‘irliksiz eng qisqa yo‘lda DFS.

DFS eng qisqa yo‘l bermaydi.

BFS majburiy.

4. Dijkstra’da eskirgan holatni tekshirmaslik

Xato kod:

```
// if (d > dist[u]) continue; // yo‘q
```

Natija:

- keraksiz holatlar
- TLE yoki WA

5. Dijkstra’ni manfiy og‘irlikda ishlatish

Qat’iyan taqiqlanadi.

Agar manfiy og‘irlik bo‘lsa: Bellman–Ford

6. DSU’da path compression yo‘q

Xato: Faqat oddiy `find`.

Katta `n` da: vaqt portlaydi → TLE

Har doim: `parent[x] = find(parent[x]);`

7. Kruskal’da qirralarni saralamaslik

MST noto‘g‘ri chiqadi.

Kruskal: saralash → birlashtirish → yig‘indi

8. Topological sort’da siklni tekshirmaslik

Sikl bo‘lsa:

- to‘liq tartib chiqmaydi
- natija xato

Kahn algoritmida: chiqgan tugunlar soni < n → sikl bor

9. SCC’ni oddiy komponent deb o‘ylash

Yo‘naltirilgan grafda: SCC ≠ oddiy komponent

Faqat:

Kosaraju

Tarjan

10. DAG’da Dijkstra ishlatish

Keraksiz murakkablik.

DAG:

- topological sort + DP
- tezroq va aniqroq

11. Segment Tree’da indeks xatolari

Off-by-one: eng ko‘p WA sababi

Boshlanish/oxiri aniq belgilanadi.

12. Fenwick Tree’da yangilashni unutish

Faqat so‘rov ishlaydi, lekin qiymat o‘zgarmaydi.

`add()` va `sum()`ni ajratib yozish shart.

13. DP holatini noto‘g‘ri aniqlash

DP noto‘g‘ri ishlaydi.

Avval: `dp[i]` nimani bildiradi?

14. Greedy’ni isbotsiz qo‘llash

Qarshi misolda yiqiladi.

Har doim savol: “Agar boshqacha tanlasam nima bo‘ladi?”

15. Java optimalligini e’tiborsiz qoldirish

Xatolar:

- `Scanner`
- `System.out.println` ko‘p marta
- keraksiz obyektlar

Natija: Java sekin ko‘rinadi

To‘g‘ri Java: tez Java

XULOSA (3-BO‘LIM · TIPIK XATOLAR)

Agar o‘quvchi:

- yuqoridagi 15 xatoni qilmasa
- algoritm tanlashda shoshilmasa
- Java optimalligiga amal qilsa

Respublika / ICPC darajasida barqaror natija qiladi.