PYTHON — 5-BO‘LIM

ADVANCED DATA STRUCTURES VA GRAPH ALGORITMLARI

1. Nazariy qism

1. Nega bu bo‘lim eng muhim?

Agar 1–4-bo‘limlar: Python’da to‘g‘ri va tez kod yozishni o‘rgatsa,

5-bo‘lim: olimpiada darajasini ajratib beradigan bo‘lim.

Bu yerda:

- oddiy algoritmlar yetmaydi
- murakkab tuzilmalar + graf fikrlash talab qilinadi

2. Graf asoslari (eslatma)

Graf:

- tugunlar (vertex)
- qirralar (edge)

Turlari:

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

Python’da ifodalanishi:

```
g = [[] for _ in range(n)]
```

3. Dijkstra algoritmi (eng qisqa yo‘l)

Qachon ishlatiladi? og‘irliklar manfiy bo‘lmagan bo‘lsa

Murakkablik: $O((n + m) \log n)$ (`heapq` bilan)

G‘oya: eng yaqin tugunni tanlab kengaytirish

Python’da majburiy algoritm.

4. Floyd–Warshall (hamma juftlar)

Qachon?

- $n \leq 400$
- barcha juftlar orasida masofa

Murakkablik: $O(n^3)$

Python’da faqat kichik n.

5. Bellman–Ford (manfiy og‘irliklar)

Afzalligi:

- manfiy og‘irliklar bilan ishlaydi
- manfiy siklni topadi

Murakkablik: `O(n·m)`

Sekin, lekin kuchli.

6. Union–Find (DSU)

Vazifasi:

- komponentlarni birlashtirish
- siklni aniqlash

Asosiy amallar:

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

Kruskal algoritmining yuragi.

7. Minimal spanning tree (MST)

Kruskal algoritmi:

- qirralarni saralash
- DSU bilan birlashtirish

Murakkablik: `O(m log m)`

Eng ko‘p tushadigan mavzu.

8. Topological sort

Qachon? yo‘naltirilgan, siklsiz graf (DAG)

Ikki usul:

- DFS
- Kahn algoritmi (BFS + indegree)

Rejalashtirish masalalari.

9. Kuchli bog‘langan komponentlar (SCC)

Algoritmlar:

- Kosaraju
- Tarjan

Murakkab graf masalalarida asos.

10. Segment Tree (asoslar)

Qachon?

- oraliq so‘rovlar
- tez yangilash + so‘rov

Murakkablik:

- so‘rov: `O(log n)`
- yangilash: `O(log n)`

Python’da ehtiyotkorlik bilan.

11. Fenwick Tree (BIT)

Segment tree’ning soddaroq ko‘rinishi

- prefix sum
- `O(log n)`

Python’da ko‘proq ishlatiladi.

12. Graph + DS kombinatsiyasi

Olimpiada darajasida:

- graf + heap
- graf + DP
- graf + bitmask

5-bo‘lim — kombinatsiyalar bo‘limi.

13. Python’da advanced algoritmlar strategiyasi

- doim `heapq`
- rekursiyani cheklash
- global listlar
- `sys.stdin.readline`

Bu yerda har sekund qimmat.

5-BO‘LIM NATIJASI

Bu nazariyani puxta o‘zlashtirgan o‘quvchi:

- 🕮 xalqaro olimpiada masalalariga yaqinlashadi

- 🕮 Python’da murakkab graf va DS masalalarni yecha oladi
- 🕮 6-bo‘limsiz ham mustaqil kuchli dasturchi bo‘ladi

2. METODIK YORDAM

1. “Masala → Graf modeli → Algoritm” zanjiri

Metodik qoida:
Masalani grafga to‘g‘ri modellashtir (tugun/qirra/og‘irlik).
Graf turini aniqla (yo‘naltirilganmi? og‘irlik bormi?).
Shundan keyin algoritm tanla.

Misollar:

- Eng qisqa yo‘l (og‘irlik ≥ 0) → Dijkstra
- Minimal xarajat bilan bog‘lash → MST (Kruskal)
- Siklsiz yo‘naltirilgan graf → Topological sort

2. Dijkstra’ni Python’da “to‘g‘ri” o‘rgatish

Metodik urg‘u:

- `heapq` majburiy
- “visited” emas, eng kichik masofa tekshiruvi

Standart shablon:

```
if d > dist[u]:
    continue
```

Bu qator yo‘q bo‘lsa — TLE yoki WA.

3. DSU (Union–Find) ni refleks darajasiga chiqarish

O‘quvchi avtomatik bilishi kerak:

- `find()` → yo‘lni siqish (path compression)
- `union()` → rank/size bilan

Metodik mashq:

- sikl bormi-yo‘qmi
- nechta komponent bor

4. MST masalalarini ikki qadamda o‘rgatish

Qirralarni saralash
DSU bilan birlashtirish

Agar qirralar saralanmasa — Kruskal ishlamaydi.

5. Topological sort’ni real masalalar bilan bog‘lash

Metodik misollar:

- fanlar prerequisiti
- ishlar ketma-ketligi
- loyihalash jadvali

DAG sharti alohida urg‘ulanadi.

6. SCC’ni “graf ichidagi graf” sifatida tushuntirish

Metodik yondashuv:

- SCC → super-tugun
- SCC’lar orasida DAG hosil bo‘ladi

Murakkab masalalar aynan shu yerda ochiladi.

7. Segment Tree vs Fenwick Tree

Metodik jadval:

Tuzilma	Qachon
Fenwick	prefix sum
Segment Tree	min/max/sum + update

Python’da:

- Fenwick → ko‘proq
- Segment Tree → ehtiyot bilan

8. Graf + DS kombinatsiyasini bosqichma-bosqich berish

Avval: graf + BFS/DFS

Keyin: graf + heap (Dijkstra)

So‘ng: graf + DSU (MST)

Bir vaqtda bermaslik — metodik shart.

9. Python optimalligi (5-bo‘limda)

Majburiy odatlar:

- `sys.stdin.readline`
- global listlar
- rekursiyada `setrecursionlimit`
- keraksiz obyekt yaratmaslik

5-bo‘limda har qator ahamiyatli.

10. Isbot va qarshi misol madaniyati

Metodik qoida:

“Nega bu algoritm ishlaydi?”
“Qachon ishlamaydi?”

Ayniqsa:

- greedy
- topological sort
- SCC

3. TIPIK XATOLAR

1. Dijkstra’ni manfiy og‘irlikda ishlatish

Qat’iyan man etiladi — noto‘g‘ri natija.

2. `heapq`dan noto‘g‘ri foydalanish

```
heappop()  # lekin eski masofani tekshirmaydi
```

`if d > dist[u]: continue` sharti yo‘q.

3. DSU’da path compression yo‘q

Katta `n` da TLE.

4. Kruskal’da qirralarni saralamaslik

MST noto‘g‘ri chiqadi.

5. Topological sort’da siklni tekshirmaslik

Natija to‘liq chiqmaydi — xato.

6. DFS’da rekursiya limitini oshirmaslik

`RecursionError`.

7. Segment tree’da indekslarni adashtirish

off-by-one xatolar — eng ko‘p uchraydi.

8. Fenwick’da yangilash va so‘rovni chalkashtirish

`add` va `sum` joyi almashib ketadi.

9. SCC’ni oddiy komponent deb o‘ylash

Yo‘naltirilgan grafda butunlay boshqa tushuncha.

10. Python sekin degan xulosa

Asl sabab:

murakkab algoritmni noto‘g‘ri qo‘llash
yoki optimallik qoidalariga amal qilmaslik

METODIK XULOSA (5-BO‘LIM)

Agar o‘quvchi:

- grafni to‘g‘ri modellashtirsa
- DS va algoritmni joyida ishlatsa
- Python optimalligiga amal qilsa

u holda 5-bo‘lim — olimpiada darajasiga chiqish nuqtasi bo‘ladi.