3-MAVZU BLOKI

ALGORITMLAR

1. Nazariy qism

1. Algoritm tushunchasi

Algoritm — bu:

- masalani yechish uchun
- aniq, cheklangan,
- bosqichma-bosqich bajariladigan ko‘rsatmalar ketma-ketligi.

Olimpiadada savol:

“Kod yozish emas, qanday algoritm tanlash muhim.”

2. Algoritmning asosiy xossalari

1. Aniqlik — har qadam tushunarli
2 Cheklanganlik — algoritm tugashi shart
3 Natijaviylik — yechim beradi
4 Ommaviylik — bir turdagi masalalar uchun ishlaydi

3. Hisoblash murakkabligi (Big-O)

Nima uchun kerak?

Bir xil masalani:

- 1 soniyada yechish
- yoki 10 soniyada yechish

farqi olimpiadada hal qiluvchi.

Asosiy murakkabliklar

Murakkablik	Izoh
O(1)	Juda tez
O(log n)	Binary search
O(n)	Bitta sikl
O(n log n)	Saralash

Murakkablik	Izoh
$O(n^2)$	Ichma-ich sikl
$O(2^n)$	Juda sekin

10^5 gacha → O(n log n) dan oshmasligi kerak.

4. Saralash algoritmlari

4.1. Oddiy saralashlar (nazariy)

- Bubble sort → $O(n^2)$
- Selection sort → $O(n^2)$
- Insertion sort → $O(n^2)$

Olimpiadada ishlatilmaydi (faqat tushunish uchun).

4.2. Tez saralash (amaliy)

C++ da:

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

Murakkablik: O(n log n)
Eng ko‘p ishlatiladigan algoritm

5. Binary search (ikkilik qidiruv)

Shart: Ma’lumot saralangan bo‘lishi kerak

Murakkablik:

O(log n)

```
binary_search(v.begin(), v.end(), x);
```

Juda tez va kuchli.

6. Two pointers (ikki ko‘rsatkich)

G‘oya: chap va o‘ng ko‘rsatkich, massiv bo‘ylab bir marta yurish

Ishlatiladi: minimal/maximal bo‘lak, juftliklar, oynacha masalalari

Murakkablik: O(n)

7. Prefix sum (oldindan yig‘indi)

Maqsad:

Oraliq yig‘indilarni tez hisoblash.

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

Oraliq yig‘indi:

```
sum(l, r) = pref[r] - pref[l-1];
```

Ko‘p testli masalalarda juda muhim.

8. Greedy algoritmlar

G‘oya:

Har qadamda eng yaxshi lokal tanlov qilish.

Misollar:

- eng arzonni tanlash
- eng qisqasini tanlash
- eng tez tugaydiganini tanlash

Doim ishlamaydi, lekin ishlaganda juda tez.

9. Rekursiya

Ta’rif:

Funksiya o‘zini o‘zi chaqiradi.

```
int f(int n) {
    if (n == 0) return 1;
    return n * f(n - 1);
}
```

Har doim:

- asosiy holat bo‘lishi shart
- aks holda cheksiz chaqiruv

10. Backtracking

G‘oya:

- barcha variantlarni sinab ko‘rish

- noto‘g‘ri yo‘ldan orqaga qaytish

Kichik n uchun ishlatiladi.

11. Dynamic programming (DP)

Asosiy tushuncha:

- masalani kichik qismlarga bo‘lish
- natijalarni saqlab borish

Takroriy hisoblashni yo‘q qiladi.

Misol:

- Fibonacci
- yo‘llar soni
- maksimal foyda

12. Graf algoritmlariga kirish

Graf:

1. tugunlar (vertex)
2. qirralar (edge)

Asosiy algoritmlar:

1. BFS — kenglik bo‘yicha
2. DFS — chuqurlik bo‘yicha

📌 Yo‘l topish, bog‘lanish tekshirish.

13. BFS (Breadth-First Search)

1. `queue` ishlatiladi
2. eng qisqa yo‘l (bir xil og‘irlikda)

Murakkablik: O(n + m)

14. DFS (Depth-First Search)

1. `stack` yoki rekursiya

2. komponentlar
3. tsikl aniqlash

Murakkablik: O(n + m)

15. Bit amallar (algoritmik)

Amal	Ma’nosi
&	VA
`	`
^	XOR
<<	chapga siljitish
>>	o‘ngga siljitish

Tez ishlaydi, maxsus masalalarda juda foydali.

3-MAVZU

2. METODIK YORDAM (kengaytirilgan)

1. “Koddan oldin algoritm” qoidasini singdirish

Eng katta xato — darrov kod yozishga o‘tish.

To‘g‘ri ketma-ketlik:
1 Masalani tahlil qilish
2 Kirish–chiqarishni aniqlash
3 Algoritmni tanlash
4 Murakkablikni baholash
5 Shundan keyin kod

Olimpiadada algoritm tanlash — 70% g‘alaba.

2. Murakkablikni doim ovoz chiqarib aytish

Har bir yechimdan keyin:

“Bu algoritm O(n) yoki O(n log n)?”

deb ayttirish kerak.

O‘quvchi: sekin yechimni o‘zi rad etishni o‘rganadi.

3. Bir masalani 2 xil algoritm bilan yechish

Metodik jihatdan juda kuchli usul:

1. 1-yechim → oddiy (sekin)
2. 2-yechim → optimal

Masalan: juftliklarni qidirish

- ichma-ich sikl → $O(n^2)$
- two pointers → $O(n)$

Farqni ko‘rgan o‘quvchi algoritmni his qiladi.

4. Saralashni “maqsad” emas, “vosita” sifatida berish

Ko‘p o‘quvchi:

“sort bo‘lsa, masala yechiladi”
deb o‘ylaydi — xato.

To‘g‘ri yondashuv:

1. nima uchun saralayapmiz?
2. saralashdan keyin nima qilamiz?

5. Two pointers va sliding window ni sxema bilan tushuntirish

Bu algoritmlar: chizmasiz tushunilishi qiyin

Metodik tavsiya:

1. chap–o‘ng ko‘rsatkichni chizib ko‘rsatish
2. har qadamda qaysi biri siljishini izohlash

Natija: ko‘p masalalar O(n) ga tushadi.

6. Prefix sum ni “tezlashtirish vositasi” sifatida berish

Tavsiya:

1. Avval har so‘rovni sikl bilan yechish
2. Keyin prefix sum bilan bir qatorda taqqoslash

“1 so‘rov → 1 amal” g‘oyasi mustahkamlanadi.

7. Greedy algoritmlarni ehtiyotkorlik bilan berish

Muhim ogohlantirish:

Greedy doim to‘g‘ri ishlamaydi.

Metodik yondashuv:

- Avval qarshi misol ko‘rsatish
- Qachon ishlashi va qachon ishlamasligini ajratish

8. Rekursiyada “chaqiruv daraxti” ni chizish

Ko‘p o‘quvchi rekursiyani:

sehrli narsa
deb qabul qiladi.

To‘g‘ri usul:

- rekursiv chaqiruvlarni daraxt ko‘rinishida chizish
- qachon to‘xtashini ko‘rsatish

9. DP ni formuladan boshlash

Dynamic programming:

- koddan emas,
- **formula (rekurrens)**dan boshlanishi kerak.

Masalan:

dp[i]=max⁡(dp[i−1],dp[i−2]+a[i])dp[i] = \max(dp[i-1], dp[i-2] + a[i])dp[i]=max(dp[i−1],dp[i−2]+a[i])

Avval formula → keyin kod.

10. BFS/DFS ni real hayot bilan bog‘lash

- BFS → eng yaqin joy
- DFS → labirint, yo‘l qidirish

Metodik foyda: grafik masalalar qo‘rqinchli bo‘lib ko‘rinmaydi.

TIPIK XATOLAR (chuqur tahlil bilan)

1. Murakkablikni hisoblamaslik

```
for (i)
  for (j)
```

n = 10^5 bo‘lsa → TLE

Eng ko‘p yiqitadigan xato.

2. Saralanmagan massivda binary search ishlatish

```
binary_search(v.begin(), v.end(), x);
```

lekin `sort` yo‘q.

Natija tasodifiy bo‘ladi.

3. Two pointers da noto‘g‘ri ko‘rsatkich siljitish

1. shart noto‘g‘ri
2. cheksiz sikl

Har qadamda qaysi pointer yuradi? savolini berish shart.

4. Prefix sum indekslarini adashtirish

```
sum(l, r) = pref[r] - pref[l];
```

To‘g‘risi:

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

5. Greedy ni tekshirmasdan ishlatish

“Har doim eng kichigini tanlaymiz” — noto‘g‘ri bo‘lishi mumkin.

Qarshi misol yo‘q → greedy yo‘q.

6. Rekursiyada asosiy holatni unutish

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

Stack overflow.

7. DP jadvalini to‘ldirish tartibini buzish: `dp[i]` uchun kerakli qiymatlar hali hisoblanmagan bo‘ladi.

TARTIB juda muhim.

8. BFS o‘rniga DFS ishlatish (eng qisqa yo‘l)

DFS eng qisqa yo‘l bermaydi.

9. Grafda belgilash (`visited`)ni unutish

Cheksiz aylanish.

10. Bit amallarni noto‘g‘ri tushunish

```
if (x & 1 == 0) // XATO
```

To‘g‘risi:

```
if ((x & 1) == 0)
```

METODIK XULOSA (3-MAVZU)

Agar o‘quvchi:

- masala → algoritm → murakkablik → kod
- sekin → tez
- oddiy → optimal

ketma-ketligini ongli o‘zlashtirsa, u haqiqiy olimpiadachi darajasiga chiqadi.