Oszd meg és uralkodj elv

(Divide and Conquer)

Szerző: Nikházy László
Előadó: Németh Zsolt

2025. október 23.

Oszd meg és uralkodj (Divide and Conquer)

- **A probléma felosztása:** bontsuk kisebb részekre, amelyek önmagukban is megoldhatók.
- **Rekurzív megoldás:** oldjuk meg az egyes részeket ugyanazzal a módszerrel.
- **Eredmények egyesítése:** kombináljuk a részproblémák megoldásait a teljes megoldáshoz.

Oszd meg és uralkodj (Divide and Conquer)

- **A probléma felosztása:** bontsuk kisebb részekre, amelyek önmagukban is megoldhatók.
- **Rekurzív megoldás:** oldjuk meg az egyes részeket ugyanazzal a módszerrel.
- **Eredmények egyesítése:** kombináljuk a részproblémák megoldásait a teljes megoldáshoz.

Lényeg

A „divide and conquer” elv lényege, hogy a probléma bonyolultságát felosztással csökkentjük, majd az így kapott kisebb feladatok eredményeit hatékonyan összekapcsoljuk.

Nevezetes alkalmazások

Milyen nevezetes oszd meg és uralkodj elven alapuló algoritmusokat ismertek?

Nevezetes alkalmazások

Milyen nevezetes oszd meg és uralkodj elven alapuló algoritmusokat ismertek?

Néhány példa (a teljesség igénye nélkül):

- Rendezési algoritmusok

Nevezetes alkalmazások

Milyen nevezetes oszd meg és uralkodj elven alapuló algoritmusokat ismertek?

Néhány példa (a teljesség igénye nélkül):

- Rendezési algoritmusok
  - **Összefésüléses rendezés (Merge Sort)**
  - **Gyorsrendezés (Quicksort)**

Nevezetes alkalmazások

Milyen nevezetes oszd meg és uralkodj elven alapuló algoritmusokat ismertek?

Néhány példa (a teljesség igénye nélkül):

- Rendezési algoritmusok
  - **Összefésüléses rendezés (Merge Sort)**
  - **Gyorsrendezés (Quicksort)**
- **Centroid dekompozíció fákra** – a fa „középponti” csúcs körüli felosztása, majd rekurzív feldolgozás.

Nevezetes alkalmazások

Milyen nevezetes oszd meg és uralkodj elven alapuló algoritmusokat ismertek?

Néhány példa (a teljesség igénye nélkül):

- Rendezési algoritmusok
  - **Összefésüléses rendezés (Merge Sort)**
  - **Gyorsrendezés (Quicksort)**
- **Centroid dekompozíció fákra** – a fa „középponti” csúcs körüli felosztása, majd rekurzív feldolgozás.
- **Divide-and-Conquer DP optimalizáció** - bizonyos tulajdonságú DP problémák hatékonyabb kiszámítása.

Nevezetes alkalmazások

Milyen nevezetes oszd meg és uralkodj elven alapuló algoritmusokat ismertek?

Néhány példa (a teljesség igénye nélkül):

- Rendezési algoritmusok
  - **Összefésüléses rendezés (Merge Sort)**
  - **Gyorsrendezés (Quicksort)**
- **Centroid dekompozíció fákra** – a fa „középponti” csúcs körüli felosztása, majd rekurzív feldolgozás.
- **Divide-and-Conquer DP optimalizáció** - bizonyos tulajdonságú DP problémák hatékonyabb kiszámítása.
- Geometriai algoritmusok térpartícionálással, pl. **Legközelebbi pontpár probléma** – a síkon elhelyezkedő pontok közül a minimális távolságú pár megtalálása.

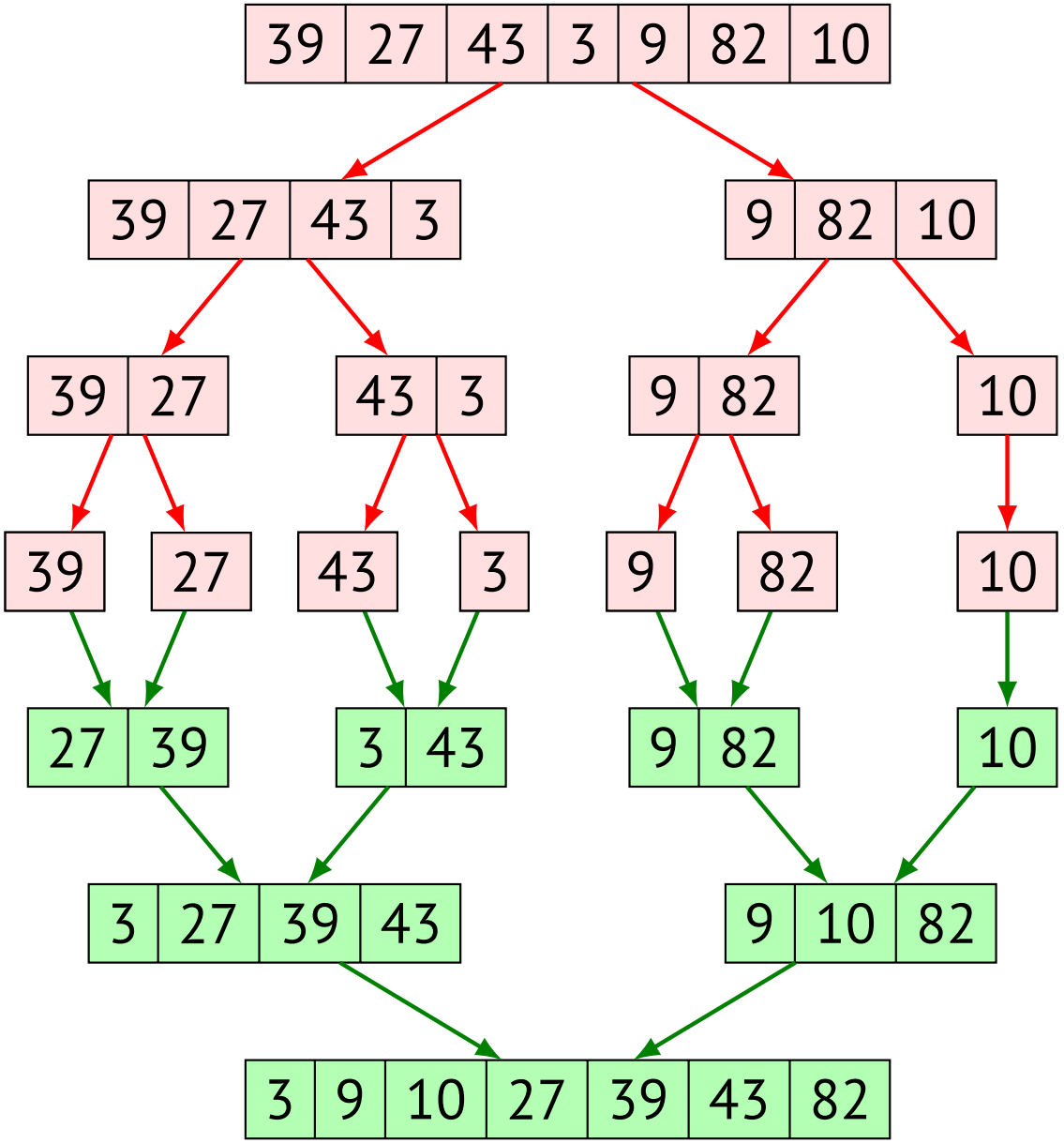
Nevezetes alkalmazások

Milyen nevezetes oszd meg és uralkodj elven alapuló algoritmusokat ismertek?

Néhány példa (a teljesség igénye nélkül):

- Rendezési algoritmusok
  - **Összefésüléses rendezés (Merge Sort)**
  - **Gyorsrendezés (Quicksort)**
- **Centroid dekompozíció fákra** – a fa „középponti” csúcs körüli felosztása, majd rekurzív feldolgozás.
- **Divide-and-Conquer DP optimalizáció** - bizonyos tulajdonságú DP problémák hatékonyabb kiszámítása.
- Geometriai algoritmusok térpartícionálással, pl. **Legközelebbi pontpár probléma** – a síkon elhelyezkedő pontok közül a minimális távolságú pár megtalálása.
- **Karatsuba-algoritmus** nagy számok vagy polinomok szorzására

Összefésüléses rendezés (Merge Sort)



Merge Sort – Tipikus Divide-and-Conquer

Tipikus példa a „Oszd meg és uralkodj” elvre

- **Felosztás:** A probléma méretét (n) két közel egyenlő részre bontjuk ($n/2$).
- **Rekurzív megoldás:** A két részproblémát külön-külön oldjuk meg.
- **Összefésülés:** A két rész megoldását **lineáris időben** kombináljuk.
  - *Kérdés:* Így mennyi lesz a teljes műveletigény? Miért?

Érdekesség

Neumann János találta fel 1945-ben.

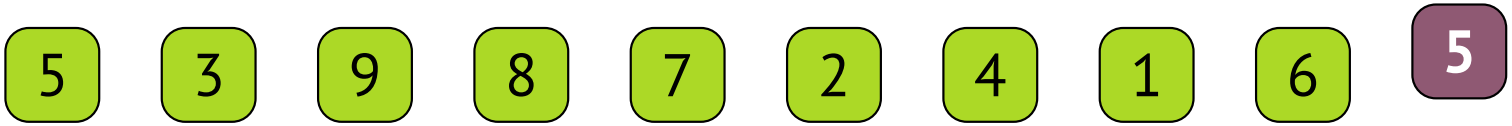
Merge Sort – Műveletigény

Elemzés

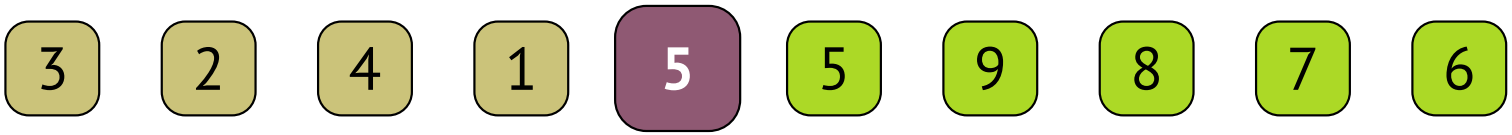
- A rekurzió $\log n$ mélységű, mivel minden szinten felezzük a méretet.
- Minden szinten az összefésülendő részek elemszáma **összesen** n.
- Az összefésülés lineáris (az elemek számával arányos)
- Tehát a teljes komplexitás $O(n \log n)$.

Gyorsrendezés - Quicksort

1. Pivot elem választása



2. Partícionálás: kisebb elemek balra, nagyobb-egyenlők jobbra



3. A két résztömbre ismételjük rekurzívan az 1. lépéstől.

Quicksort – Tulajdonságok

„Oszd meg és uralkodj” véletlen választás alapján

- **Felosztás:** A két rész mérete a pivot elem választásától függ.
- **Partícionálás:** A rekurzív hívás előtt már lényegében megcsináljuk a kombinálást, hiszen az elemek csak a résztömbökön belül fognak mozogni. A pivot elem a helyén van.
- **Műveletigény:** A partícionálás **lineáris** idejű.
  - *Kérdés:* Így mennyi lesz a teljes műveletigény? Miért?

Érdekesség

Tony Hoare találta fel 1959-ben.

Gyorsrendezés (Quicksort) – Futási idők

Futási idők

- **Legjobb eset:** $O(n \log n)$ Ha a pivot minden lépésben nagyjából felezi a tömböt.
- **Legrosszabb eset:** $O(n^2)$ Ha a pivot mindig a legkisebb vagy legnagyobb elem – a felosztás aránytalan.
- **Átlagos eset:** $O(n \log n)$ Véletlenszerű pivotválasztás mellett.

Mit jelent az „átlagos eset”?

Nem a legjobb és legrosszabb eset **átlaga**, hanem a **várható futási idő**, ha azt feltételezzük, hogy az adatok minden lehetséges permutációja azonos valószínűséggel lehet a bemeneten.

Feladat: Legközelebbi pontpár

Minimum Euclidean Distance
`https://cses.fi/problemset/task/2194`

Adott n darab pont a kétdimenziós síkon. Határozd meg bármely két **különböző** pont közötti **legkisebb euklideszi távolság négyzetét**.

Definíció

Két pont, (x_1, y_1) és (x_2, y_2) **euklideszi távolsága**:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Feladat: Legközelebbi pontpár

Bemenet / Kimenet

A standard bemenet első sora tartalmazza a pontok n számát. Ezt követően n sorban a pontok koordinátái (x, y) egész számokként. Feltételezhető, hogy minden pont különböző.
A standard kimenetre egyetlen egész számot kell írni: d^2, ahol d a legkisebb euklideszi távolság a pontok között.

Korlátok

$2 \leq n \leq 2 \cdot 10^5$
$-10^9 \leq x, y \leq 10^9$

Példa

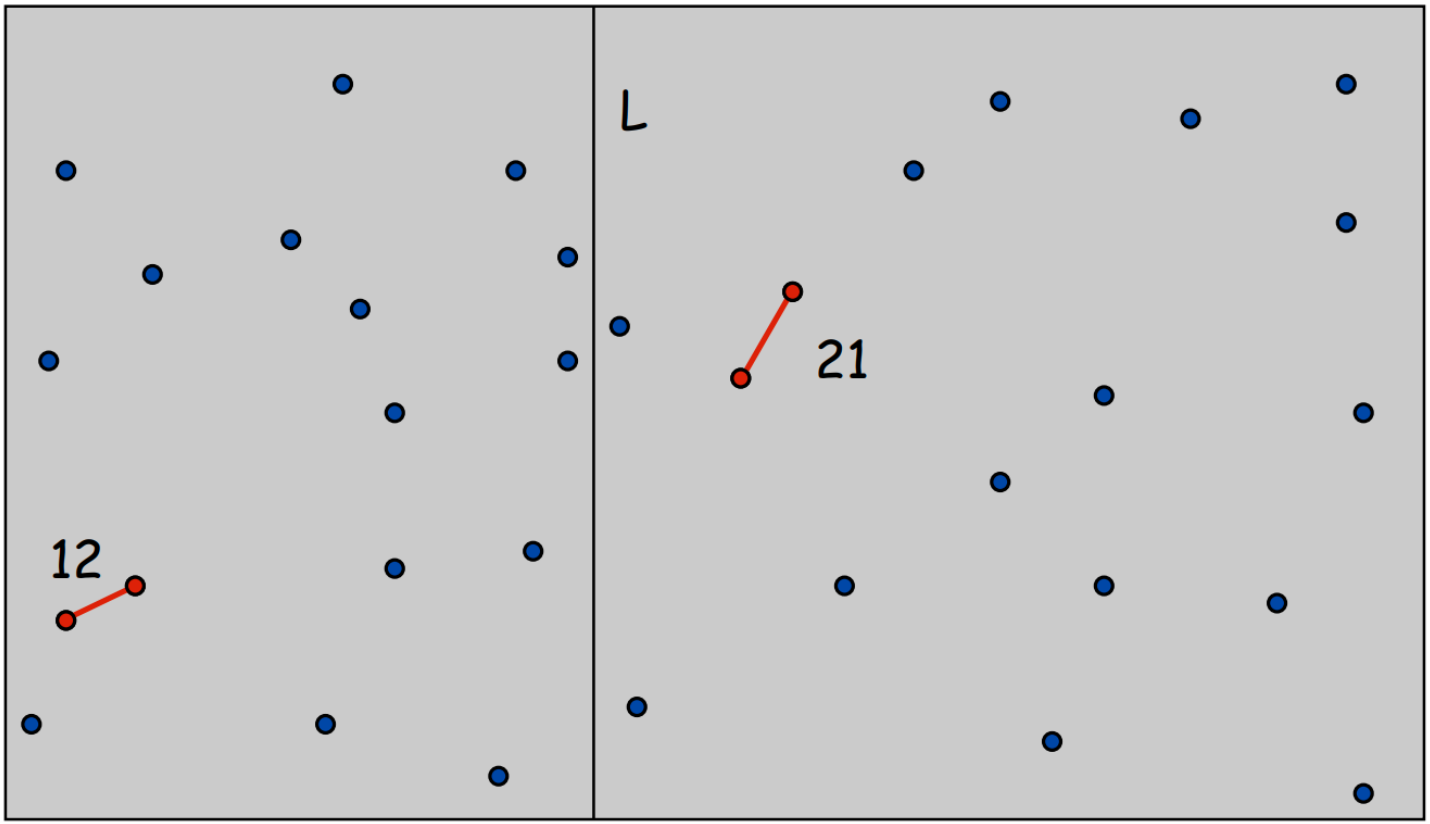
Bemenet	Kimenet
4 2 1 4 4 1 2 6 3	2

Legközelebbi pontpár

Megoldási ötletek?

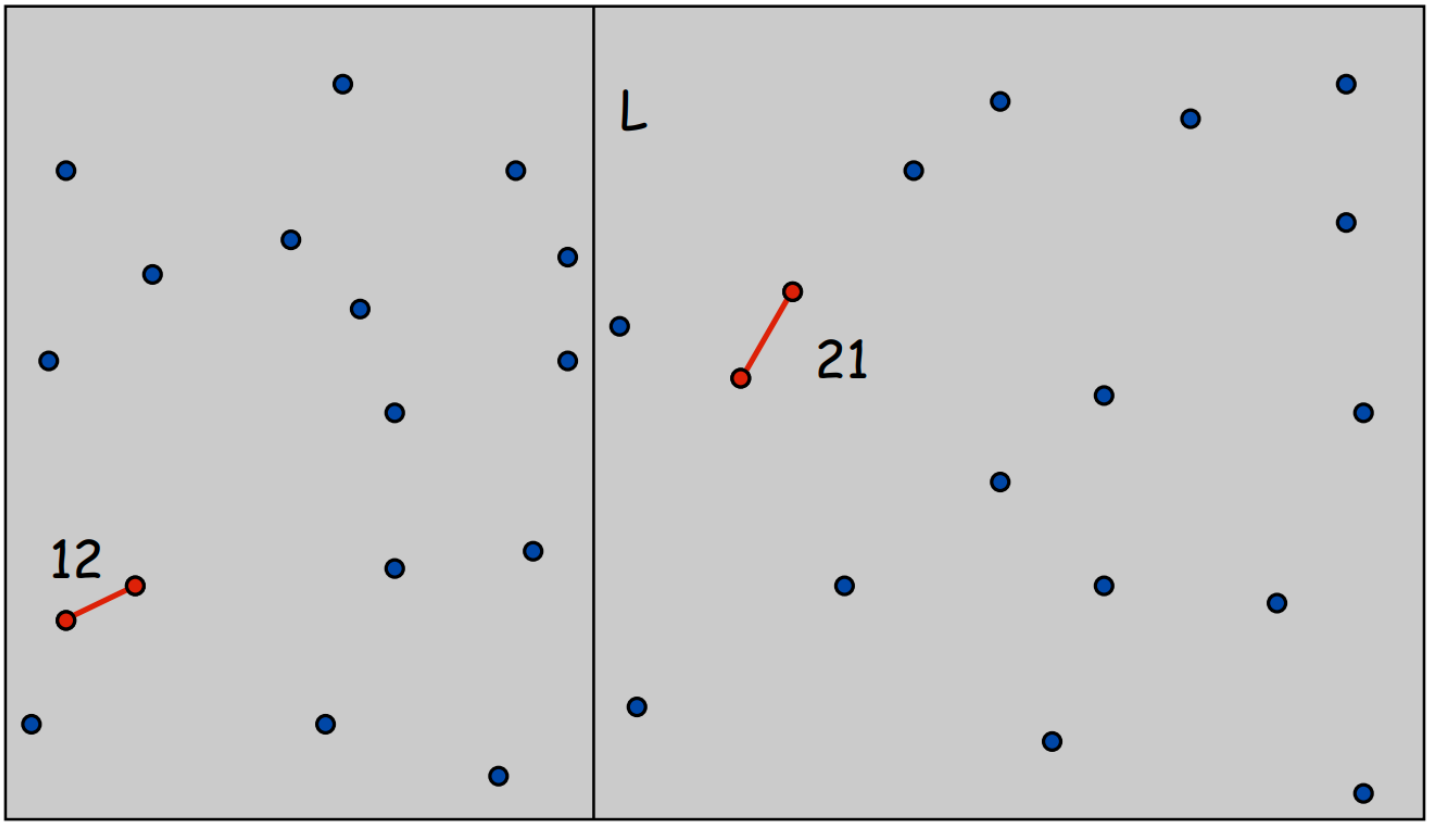
Legközelebbi pontpár

- **Felosztás**: húzzunk be egy függőleges vonalat úgy, hogy nagyjából $\frac{n}{2}$ pont van mindkét oldalon.



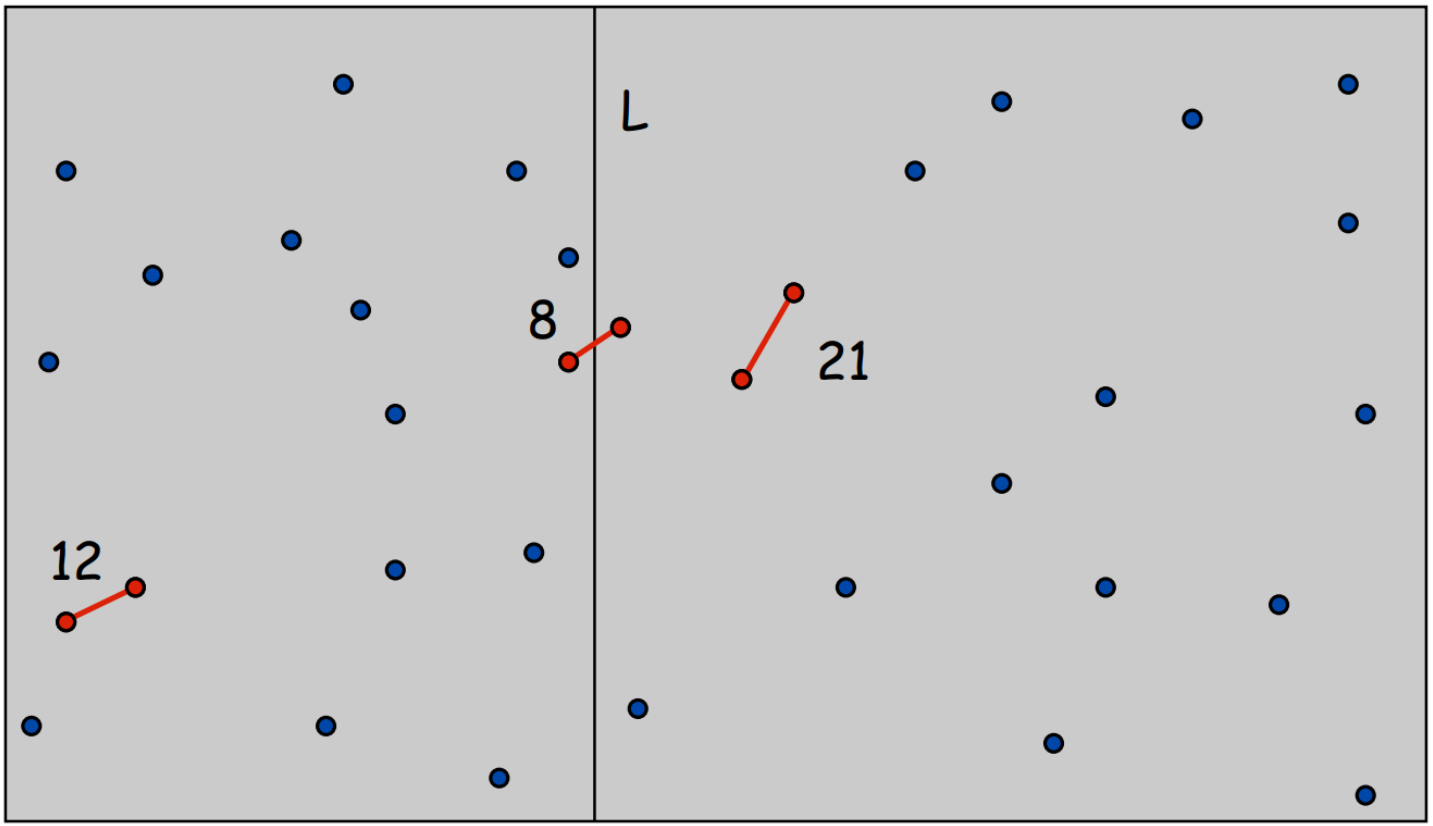
Legközelebbi pontpár

- **Felosztás**: húzzunk be egy függőleges vonalat úgy, hogy nagyjából $\frac{n}{2}$ pont van mindkét oldalon.
- **Rekurzív megoldás**: a két oldalon belül keressük meg a legközelebbi pontpárt.



Legközelebbi pontpár

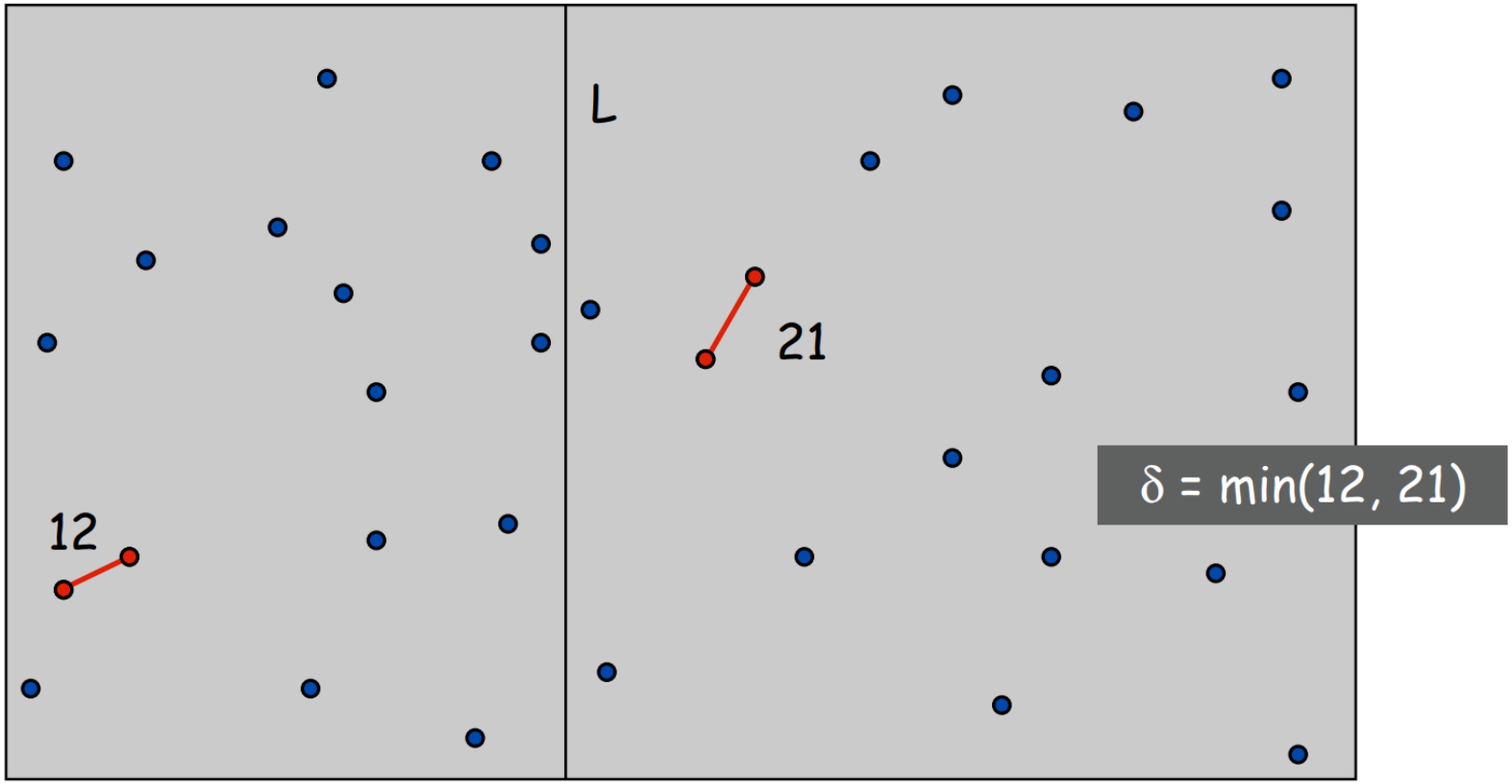
- **Felosztás**: húzzunk be egy függőleges vonalat úgy, hogy nagyjából $\frac{n}{2}$ pont van mindkét oldalon.
- **Rekurzív megoldás**: a két oldalon belül keressük meg a legközelebbi pontpárt.
- **Egyesítés**: keressük meg a határ két különböző oldalán lévő pontok közül a legközelebbi párt, és
- a három lehetséges pontpár közül a legjobbat tartsuk meg.



Legközelebbi pontpár

Keressük meg a határ két különböző oldalán lévő pontok közül a legközelebbi párt

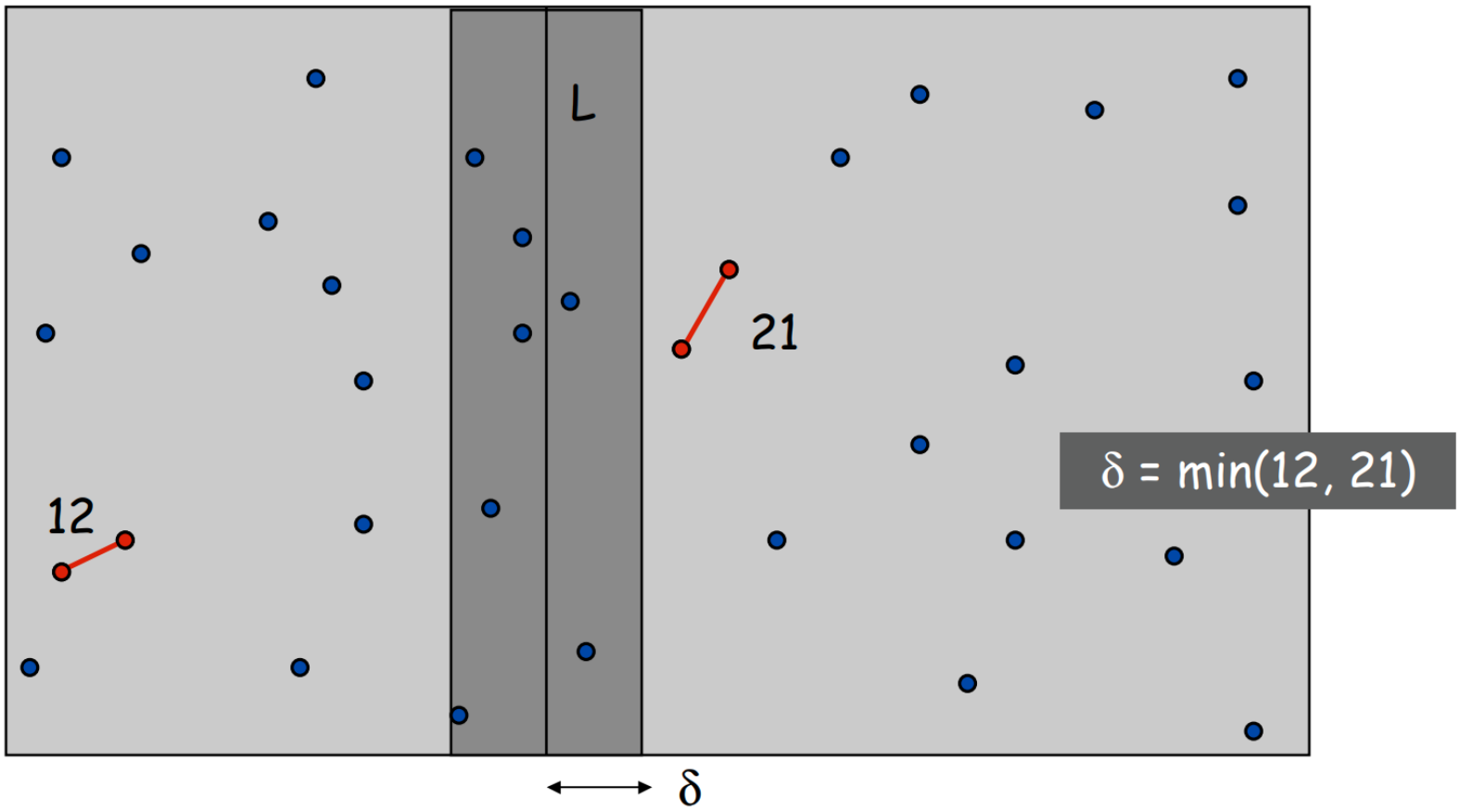
- Ez $O(n^2)$ műveletigényűnek tűnik elsőre
- De kihasználhatjuk, hogy csak az olyan pontpárok érdekesek, amelyek távolsága kisebb, mint a két oldalon talált pontpárok távolsága (δ)



Legközelebbi pontpár

Keressük meg a határ két különböző oldalán lévő pontok közül a legközelebbi párt ***feltéve, hogy ez a távolság*** $< \delta$.

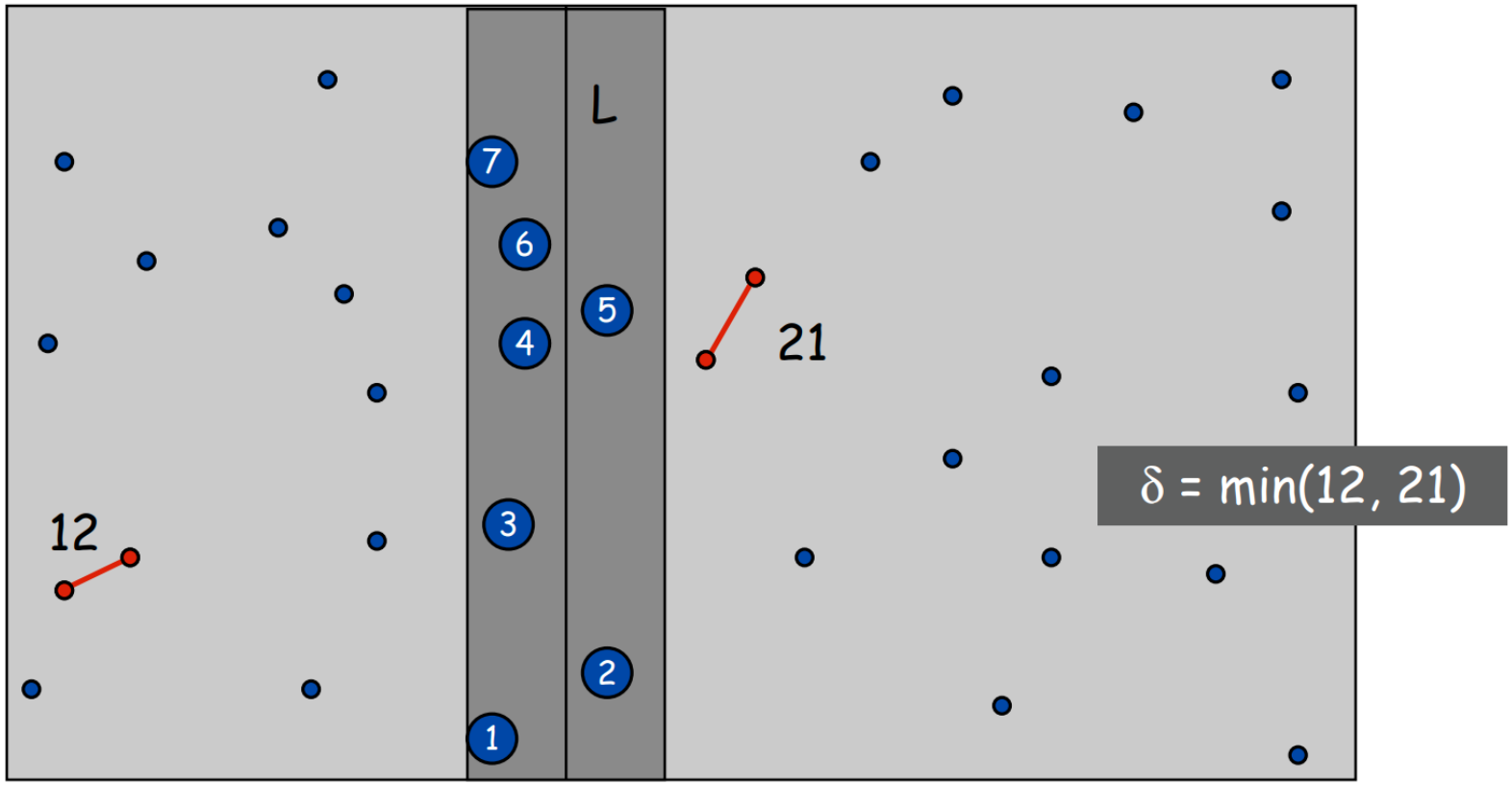
- Csak az L határvonaltól $< \delta$ távolságra lévő pontokat kell vizsgálni



Legközelebbi pontpár

Keressük meg a határ két különböző oldalán lévő pontok közül a legközelebbi párt ***feltéve, hogy ez a távolság*** $< \delta$.

- Csak az L határvonaltól $< \delta$ távolságra lévő pontokat kell vizsgálni
- Rendezzük ebben a 2δ szélességű sávban a pontokat az y koordinátájuk alapján.
- A sorrendben legfeljebb 11 távolságra lévő pontok távolságát kell csak vizsgálni.



Legközelebbi pontpár

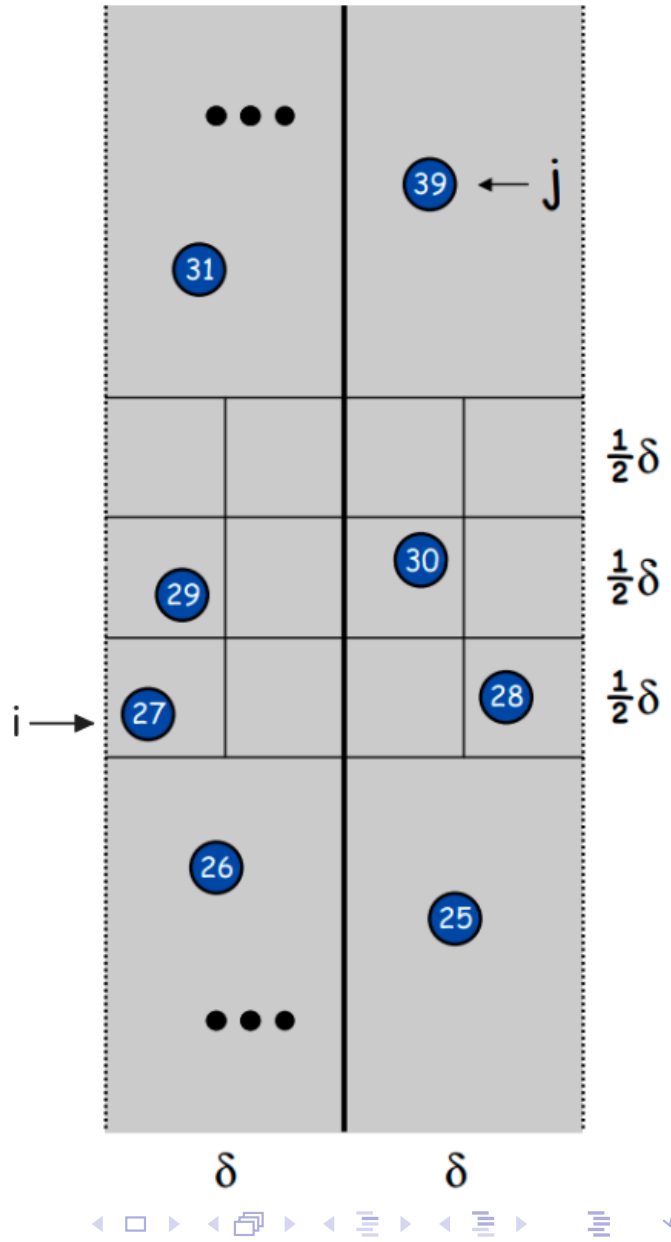
Jelölje P_i a 2δ széles sávon belüli i-edik legkisebb y-koordinátájú pontot.

Állítás: ha $j - i \geq 12$, akkor P_i és P_j távolsága legalább δ.

Bizonyítás:

- Nem lehet egynél több pont egy $\frac{\delta}{2}$ oldalú négyzeten belül
- Így legfeljebb 11 pont lehet a P_i után, ami még nincs messzebb, mint δ (csupán az y koordinátát tekintve).

Megjegyzés: valójában 12 helyett 7-re is igaz az állítás.



Legközelebbi pontpár – Összefoglalás és futásidő

Az algoritmus lépései:

1. **Felosztás:** válasszunk egy függőleges vonalat, ami nagyjából felezi a pontokat.
2. **Rekurzió:** oldjuk meg a legközelebbi pontpárt a bal és jobb oldalon külön-külön.
3. **Egyesítés:** számítsuk ki a legkisebb távolságot a határsávon belül (szélessége 2δ), ahol δ a két részmegoldás közül a kisebb távolság.

Futásidő elemzése:

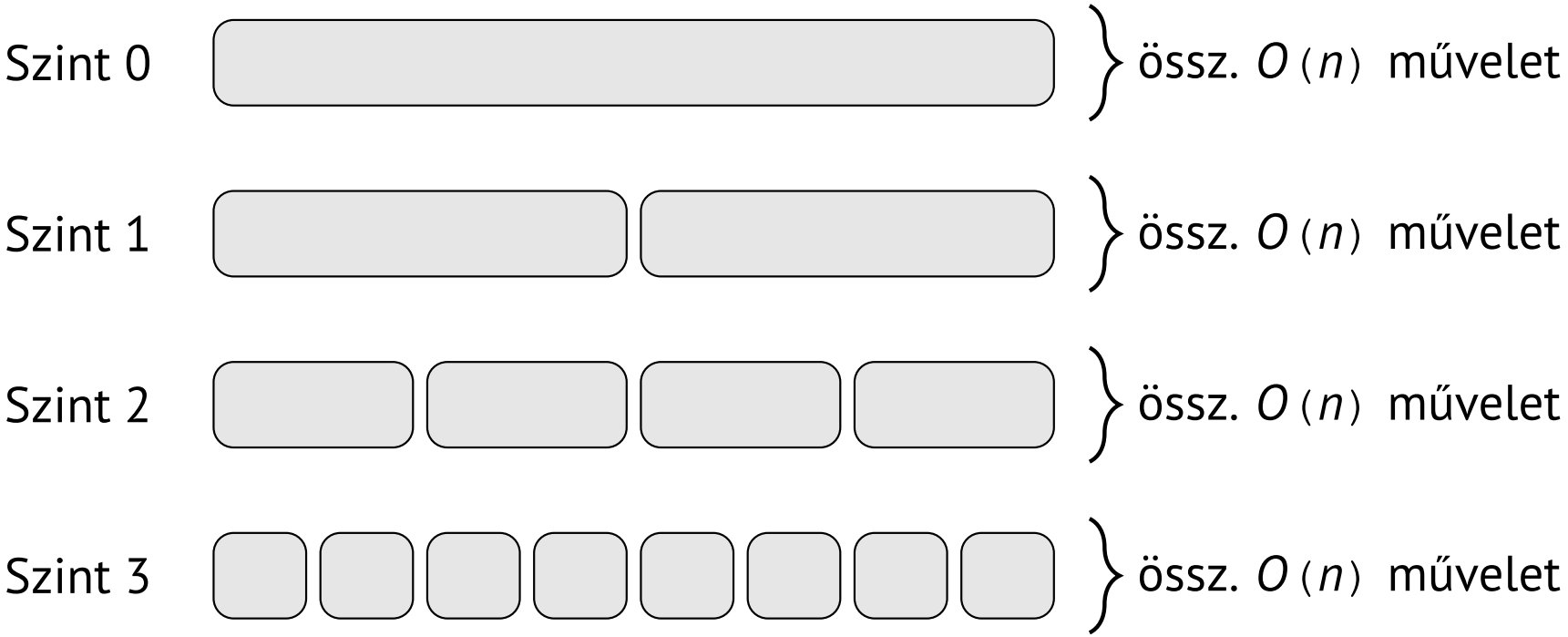
- A rekurzióban két részprobléma keletkezik, méretük kb. $n/2$.
- Az **egyesítés** lineáris időben megoldható, ha a pontok y szerint rendezve maradnak.
- Így: $T(n) = 2T(n/2) + O(n) \quad \Rightarrow \quad T(n) = O(n \log n)$

Összegzés: A **legközelebbi pontpár** probléma hatékonyan megoldható az **oszd meg és uralkodj** elvvel, $O(n \log n)$ futásidő érhető el, a naiv $O(n^2)$ helyett.

Kód: `https://cses.fi/paste/1165eb6646ab9b53e56d35/`

Futásidő elemzés - Általánosan - Egyesítés O(n)

Rekurzió: 2 részprobléma/szint, egyesítés: $O(n)$

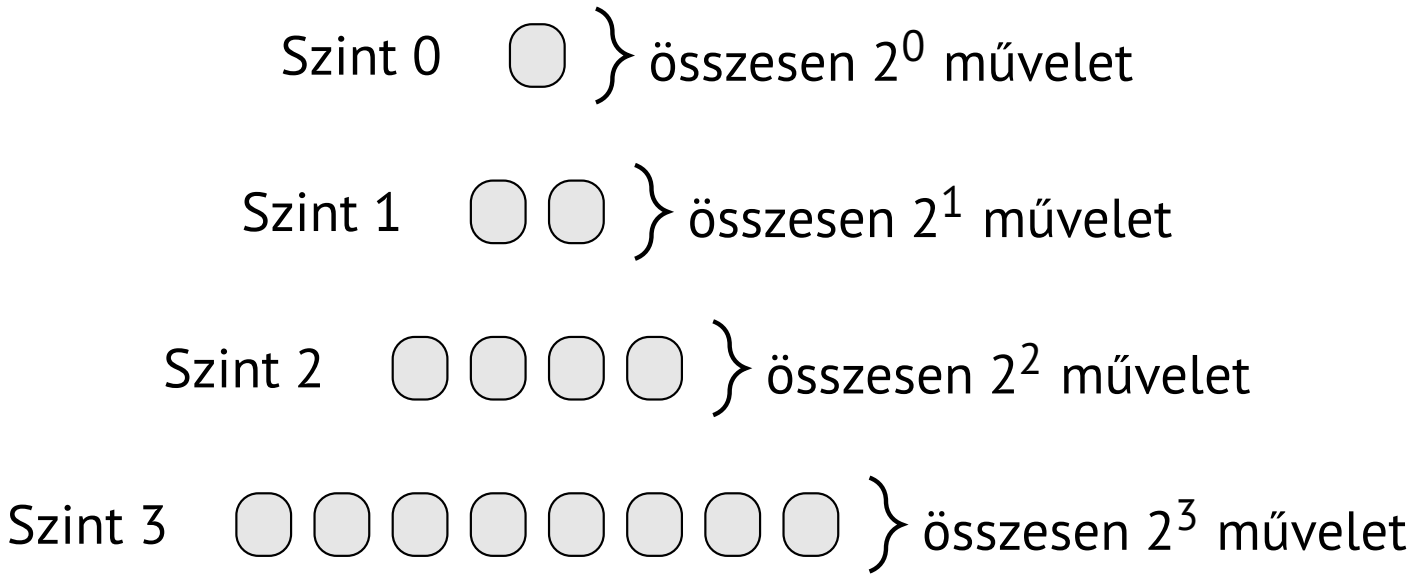


Minden szinten az összmunka $O(n)$, a szintek száma $\approx \log_2 n + 1$

$$\Rightarrow T(n) = \underbrace{O(n) + \cdots + O(n)}_{\log_2 n+1 \text{ szint}} = O(n \log n) .$$

Futásidő elemzés - Általánosan – Egyesítés $O(1)$

Rekurzió: 2 részprobléma/szint, egyesítés: $O(1)$



Minden részprobléma $O(1)$ művelet, a k-adik szinten 2^k részprobléma

$$\Rightarrow \; T(n) = \sum_{k=0}^{\lfloor \log_2 n \rfloor} O(2^k) = O(n) \; .$$

Karatsuba-szorzás

Probléma: A hagyományos („iskolai”) szorzás két n jegyű számra $O(n^2)$ műveletet igényel.

Cél: Gyorsabb algoritmus nagy számok szorzására az **oszd meg és uralkodj** elvvel.

Kulcsötlet (Karatsuba, 1960): A két számot két részre bontjuk, és a szorzást úgy szervezzük át, hogy **4 helyett csak 3 részszorzást** kelljen végezni (plusz összeadások/kiemelések).

Történeti érdekesség:

- Anatolij Karacuba 23 évesen találta meg az eljárást.
- Tanára, Kolmogorov szerint nem létezhet $O(n^2)$-nél gyorsabb módszer.
- Egy héten belül megcáfolta a hipotézist.

Karatsuba-szorzás – A számok felbontása

Jelölések:

- B: az alap (pl. 10, 2, 16, ...)
- m: a szétválasztás helye (körülbelül $n/2$ jegy)

Bontsuk a számokat két részre:

$$x = x_1 B^m + x_0, \quad y = y_1 B^m + y_0$$

Példa: $x = 12345$, $y = 6789$, $B = 10$, $m = 3$

$$x_1 = 12, \quad x_0 = 345, \quad y_1 = 6, \quad y_0 = 789$$

Hagyományos szorzás:

$$xy = x_1 y_1 B^{2m} + (x_1 y_0 + x_0 y_1) B^m + x_0 y_0$$

Ez **4 szorzást** igényel: $x_1 y_1$, $x_1 y_0$, $x_0 y_1$, $x_0 y_0$

Karatsuba-szorzás – A trükk: csak három szorzás

Megfigyelés:

$$(x_1 + x_0)(y_1 + y_0) = x_1y_1 + x_1y_0 + x_0y_1 + x_0y_0$$

Innen a középső tagot $(x_1y_0 + x_0y_1)$ ki tudjuk számolni:

$$z_1 = (x_1 + x_0)(y_1 + y_0) - x_1y_1 - x_0y_0$$

Három szorzás:

$$z_0 = x_0y_0$$

$$z_2 = x_1y_1$$

$$z_3 = (x_0 + x_1)(y_0 + y_1)$$

$$z_1 = z_3 - z_2 - z_0$$

Összeállítás:

$$xy = z_2B^{2m} + z_1B^m + z_0$$

Eredmény: 3 szorzás + néhány összeadás = gyorsabb!

Karatsuba-szorzás – Példa és futásidő

Példa: $x = 12345$, $y = 6789$, $B = 10$, $m = 3$

$$z_2 = 12 \times 6 = 72,$$
$$z_0 = 345 \times 789 = 272\,205,$$
$$z_3 = (12 + 345) \times (6 + 789) = 357 \times 795 = 283\,815,$$
$$z_1 = z_3 - z_2 - z_0 = 11\,538$$

$$xy = z_2 B^{2m} + z_1 B^m + z_0 = 72 \cdot 10^6 + 11\,538 \cdot 10^3 + 272\,205 = 83\,810\,205$$

Futásidő (bizonyítás nélkül közöljük):

$$T(n) = 3T(n/2) + O(n)$$

$$\Rightarrow T(n) = O(n^{\log_2 3}) \approx O(n^{1.585})$$

Megjegyzés: Kis számoknál a többletösszeadások miatt **lassabb**, de nagy számoknál már gyorsabb, mint az „iskolai” módszer.

Kitérő: Meet in the Middle elv

Alapelv

- A probléma keresési tere túl nagy (2^n vagy $n!$).
- **Felosztjuk** a bemenetet két részre, pl. két félhalmazra.
- Mindkét részre **összes lehetséges részmegoldást** kiszámítjuk.
- Ezeket **összevetjük** vagy **illesztjük**, hogy megkapjuk a teljes megoldást.

Tipikus hatás

A komplexitás:

$$O(2^n) \Rightarrow O(2^{n/2}) \text{ idő és memória.}$$

Összehasonlítás

Nem rekurziót, hanem **két részprobléma teljes előszámítását** használja.

Meet in the Middle – Versenyfeladat példa

Codeforces 1006F – XOR Paths

Adott egy $n \times m$ rács, ahol minden mezőben egy egész szám van ($n, m \leq 20$). A bal felső $(1, 1)$ mezőtől a jobb alsó (n, m) mezőig haladunk csak jobbra vagy lefelé. Meg kell számolni, hány út mentén lesz a mezők XOR-ja éppen k.

- Összes út: $\binom{n+m-2}{n-1}$ – nagyon gyorsan nő.

Meet in the Middle – Versenyfeladat példa

Codeforces 1006F – XOR Paths

Adott egy $n \times m$ rács, ahol minden mezőben egy egész szám van ($n, m \leq 20$). A bal felső $(1, 1)$ mezőtől a jobb alsó (n, m) mezőig haladunk csak jobbra vagy lefelé. Meg kell számolni, hány út mentén lesz a mezők XOR-ja éppen k.

- Összes út: $\binom{n+m-2}{n-1}$ – nagyon gyorsan nő.
- Meet in the Middle:
  - Az út első felét és második felét külön bejárjuk.
  - A középső átlónál (ahol $i + j$ állandó) „találkozik” a két fél.
  - Az egymást k-ra kiegészítő XOR-értékeket illesztjük össze.

Meet in the Middle – Versenyfeladat példa

Codeforces 1006F – XOR Paths

Adott egy $n \times m$ rács, ahol minden mezőben egy egész szám van ($n, m \leq 20$). A bal felső $(1, 1)$ mezőtől a jobb alsó (n, m) mezőig haladunk csak jobbra vagy lefelé. Meg kell számolni, hány út mentén lesz a mezők XOR-ja éppen k.

- Összes út: $\binom{n+m-2}{n-1}$ – nagyon gyorsan nő.
- Meet in the Middle:
  - Az út első felét és második felét külön bejárjuk.
  - A középső átlónál (ahol $i + j$ állandó) „találkozik” a két fél.
  - Az egymást k-ra kiegészítő XOR-értékeket illesztjük össze.
- Időkomplexitás: $O(2^{(n+m)/2})$, jól működik $n, m \leq 20$ esetén.

Feladatok

Feladatok

2 feladat elméletben + 1 kódolásra, két különböző szinten: normál vagy emelt szint.
Mindegyik feladatnál Divide-and-Conquer típusú megoldást kérünk akkor is, ha lenne más módszerrel is megoldás.

Normál szint

- N1. (Elméletben) Maximális haszon
- N2. (Elméletben) Majoráns elem
- N3. (Kódolva) Permutation Graph

Emelt szint

- E1. (Elméletben) Részhalmaz összegek
- E2. (Elméletben) Rendezett intervallumok tartalmazása
- E3. (Kódolva) Intrinsic Interval

N1. Maximális haszon

Egy vállalat részvényeinek napi árfolyamát ismerjük egy N elemű tömbben: $A[1], A[2], \ldots, A[N]$, ahol $A[i]$ az i-edik nap záróára. Szeretnénk egyetlen alkalommal **vásárolni**, majd később **eladni** pontosan egy darab részvényt. A cél a **lehető legnagyobb haszon** elérése.

Formálisan: a keresett érték

$$\max_{1 \le i < j \le N} (A[j] - A[i])$$

Példa

Bemenet	Kimenet
6 7 1 5 3 6 4	5

Korlátok

$1 \le N \le 200\,000, \quad 1 \le A[i] \le 10^9$

N2. Majoráns elem

Van egy rejtett, N elemű tömb, amelynek elemei ismeretlenek, de tetszőleges két indexre, i és j, meg tudjuk kérdezni, hogy az elemek **azonosak-e**:

$$\texttt{query}(i, j) \Rightarrow \begin{cases} \texttt{true}, & \text{ha } A[i] = A[j], \\ \texttt{false}, & \text{különben.} \end{cases}$$

A tömbben létezhet egy **majoráns elem**, amely több mint $\frac{N}{2}$ alkalommal fordul elő.
A feladat: állapítsuk meg, **van-e majoráns elem**, és ha igen, **adjunk meg egy olyan indexet**, ahol ez az érték található.

Korlátok

$1 \leq N \leq 200\,000$
A lekérdezések száma legyen minél kevesebb!

N3. Permutation Graph

https://codeforces.com/contest/1696/problem/D

Adott az $1..n$ számok egy permutációja: $a = [a_1, a_2, \ldots, a_n]$.
Hozzunk létre egy **gráfot** n **csúccsal** $(1, 2, \ldots, n)$. Minden $1 \leq i < j \leq n$ párra tegyünk élt i és j közé, ha az

$$mn(i, j) = \min_{k=i..j} a_k, \quad mx(i, j) = \max_{k=i..j} a_k$$

értékekre teljesül, hogy $(mn(i, j), mx(i, j)) = (a_i, a_j)$ vagy (a_j, a_i).

Milyen hosszú a **legrövidebb út** a gráfban 1-ből n-be?

Korlátok

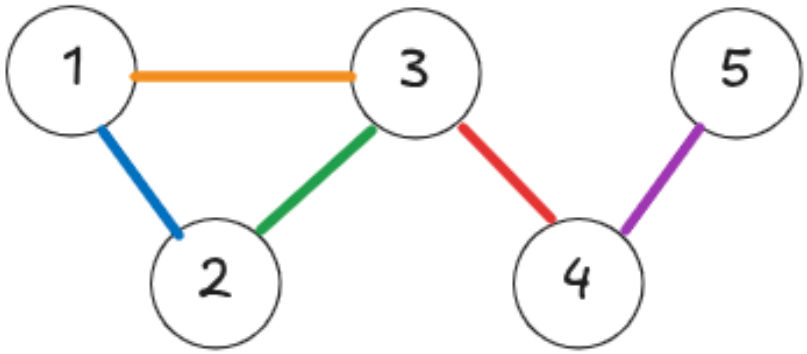
$1 \leq n \leq 2.5 \cdot 10^5$

N3. Permutation Graph

Példa

Bemenet	Kimenet
5 1 4 5 2 3	3

Magyarázat: az alábbi ábrán be vannak jelölve azok az (i, j) intervallumok, amelyekre igaz, hogy a minimum és maximum a két szélén van, és ez alapján látható a létrejövő gráf, amiben az 1. és 5. csúcs távolsága 3.



E1. Részhalmaz összegek (Sum over Subsets)

Adott egy 2^n elemű tömb: $a_0, a_1, \ldots, a_{2^n-1}$, amelynek indexeit bináris alakban, bitmaszkokként képzeljük el. Azt szeretnénk kiszámítani, hogy minden i bitmaszkhoz mi az

$$A_i = \sum_{i \text{ fedi } j\text{-t}} a_j,$$

vagyis az összeg azon a_j értékekre, amelyekre j minden 1-es bite az i-ben is 1. Például $n = 2$ esetén:

$$A_0 = a_0,\ A_1 = a_0 + a_1,\ A_2 = a_0 + a_2,\ A_3 = a_0 + a_1 + a_2 + a_3$$

Korlátok

$1 \leq n \leq 20$
$|a_i| \leq 10^9$

E2. Rendezett intervallumok tartalmazása

Adott n darab intervallum: $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$. Számoljuk meg azokat az (i, j) párokat, amelyekre:

$$i < j, \quad l_i \leq l_j, \quad r_i \geq r_j.$$

Vagyis a kisebb indexű intervallum teljesen tartalmazza a nagyobb indexűt.

Példa

Bemenet	Kimenet
5 1 7 2 5 -3 6 -4 0 5 5	4

A példában a négy érvényes (i,j) pár: $(1, 2)$, $(1, 5)$, $(2, 5)$, $(3, 5)$.

Korlátok

$1 \leq n \leq 200\,000$; $\quad -10^9 \leq l_i \leq r_i \leq 10^9$.

E3. Intrinsic Interval

`https://codeforces.com/gym/101620` – Problem I

Adott az $1..n$ számok egy **permutációja**: $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$
Egy **valódi intervallum** a permutációban olyan *folytonos szakasz*, amelynek elemeit növekvő sorrendbe rendezve **egymást követő számokat** kapunk.

Példa: $\pi = (3, 1, 7, 5, 6, 4, 2)$ esetén $\pi_3^6 = (7, 5, 6, 4)$ valódi intervallum (4–7 közötti számokat tartalmaz), de $\pi_1^3 = (3, 1, 7)$ nem az.

Feladat: Adott m darab szakasz: $\pi_{x_j}^{y_j}$. Mindegyikhez keressünk egy $\pi_{a_j}^{b_j}$ **legszűkebb valódi intervallumot**, ami tartalmazza azt ($a_j \leq x_j \leq y_j \leq b_j$).

E3. Intrinsic Interval – példa

Példa

Bemenet	Kimenet
7 3 1 7 5 6 4 2 3 3 6 7 7 1 3	3 6 7 7 1 7

Korlátok

$1 \leq n, m \leq 100\,000$

$1 \leq \pi_i \leq n$, a számok páronként különbözőek.

Időlimit: 3 másodperc **Memórialimit:** 512 MiB