ЛЕКЦІЯ 8

БЛИЗКІСТЬ . ОСНОВНІ АЛГОРИТМИ.

5.1. Основні задачі.

Задача Б.1 (НАЙБЛИЖЧА ПАРА). *На площині задано N точок. Знайти дві із них, відстань між якими найменша (може виявитись, що таких пар може бути декілька, тоді достатньо знайти хоча б одну із них).*

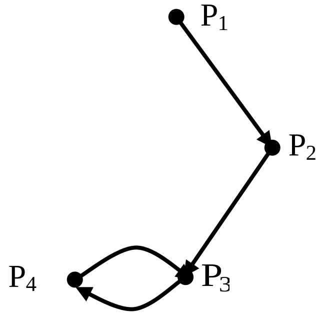
Звичайний перебір кожної пари точок у випадку d- мірного простору дає час $O(dN^2)$.

В одновимірному випадку існує більш швидкий алгоритм, який використовує той факт, що кожна пара найближчих точок повинна складатись із послідовних точок множини при упорядкуванні її по значенню координати. Таким чином, можна упорядкувати N заданих дійсних чисел за O(N log N) кроків, а потім здійснити лінійний перегляд упорядкованої послідовності (x_1, x_2, …, x_N) за час O(N), обчислюючи при цьому x_{i+1}- x_i, i=1,...,N-1.

Задача Б.1 (УСІ НАЙБЛИЖЧІ СУСІДИ). *На площині задано N точок. Знайти найближчого сусіда для кожної точки множини.*

Означення. *“Найближчий сусід” – це відношення на множині точок S, яке визначається таким чином: точка b є найближчим сусідом точки a (позначається $a \rightarrow b$) , якщо*

$\mathrm{dist}(a,b) = \mathrm{mindist}(a,c)$, $c \in S - a$, (мал.. 5.1)



(P_1,P_2), (P_2,P_3), (P_3,P_4), (P_4,P_3)

$P_4 \rightarrow P_3$ та $P_3 \rightarrow P_4$ – взаємна пара

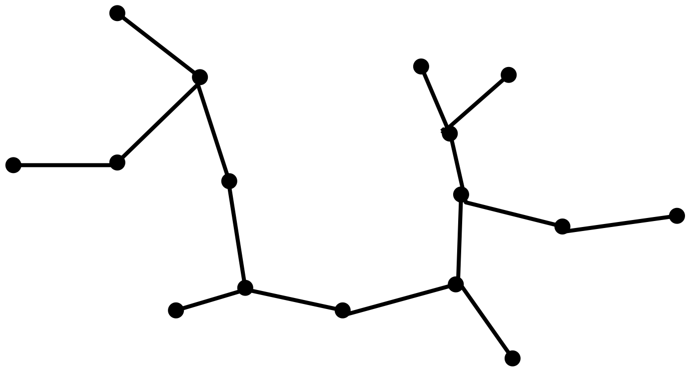
Мал..5.1. Граф відношення “Найближчий сусід” на множині точок.

Відношення необов’язково симетричне (Із того, що $a \rightarrow b$ не обов’язково$\Rightarrow$ $b \rightarrow a$).(Хоча точка може мати найближчим сусідом кожну із точок множини, вона може мати найближчим сусідом не більше 6 точок для d = 2 і не більше 12 для d =3). Розв’язком задачі являється сукупність упорядкованих пар (a,b), де $a \rightarrow b$.

Означення. Пара точок, яка задовольняє умові симетричності відношення називається *взаємною парою*.

Задача Б.3 (ЕВКЛІДОВО МІНІМАЛЬНЕ ОСТОВНЕ ДЕРЕВО). *На площині задано N точок. Побудувати дерево, вершинами якого є усі задані точки і сумарна довжина усіх ребер якого мінімальна.*

Розв'язком задачі являється список, який містить N-1 пару точок. Кожна пара представляє ребро дерева. На малюнку 5.2 показаний приклад такого дерева.



Мал..5.2. Мінімальне остове дерево множини точок на площині.

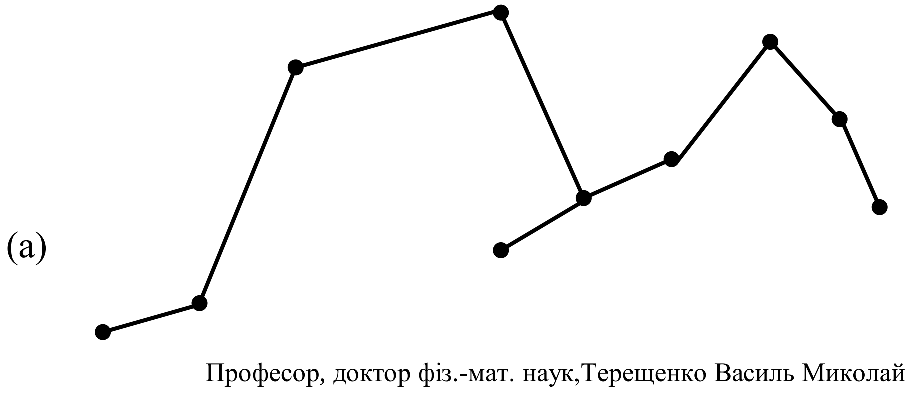
Означення. *Дерево, яке має найкоротшу довжину і задовольняє умові, що до вихідної множини не заборонено додавати нові точки, називається деревом Штейнера (мал. 5.3.).*

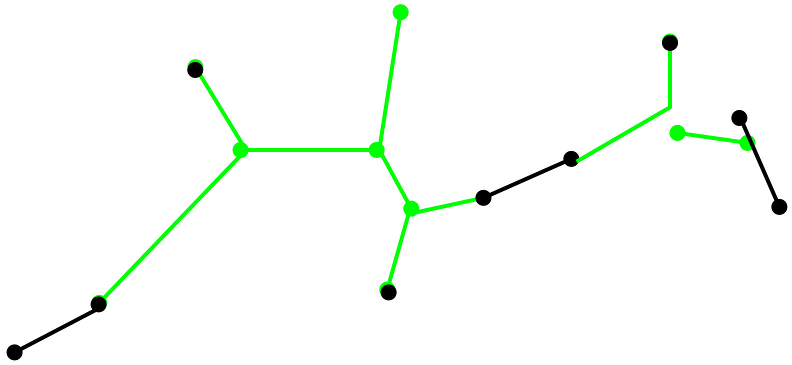
Задача про МОД зазвичай формулюється як задача теорії графів: задано зважений граф з N вершинами і E ребрами, необхідно знайти найкоротше піддерево графа G, яке включає усі його вершини.

У випадку евклідової задачі про МОД N вершин визначаються 2N координатами точок на площині, а відповідний граф містить ребра, які з'єднують кожну пару вершин.

Вага ребра дорівнює відстані між точками, з'єднаних ребром. В цьому випадку використання кращого із відомих алгоритмів розв'язання задачі про МОД вимагатиме часу $\theta(N^2)$.

І так як МОД *завжди містить саме коротке ребро графа* G, наведене значення являється нижньою оцінкою для довільного графа.



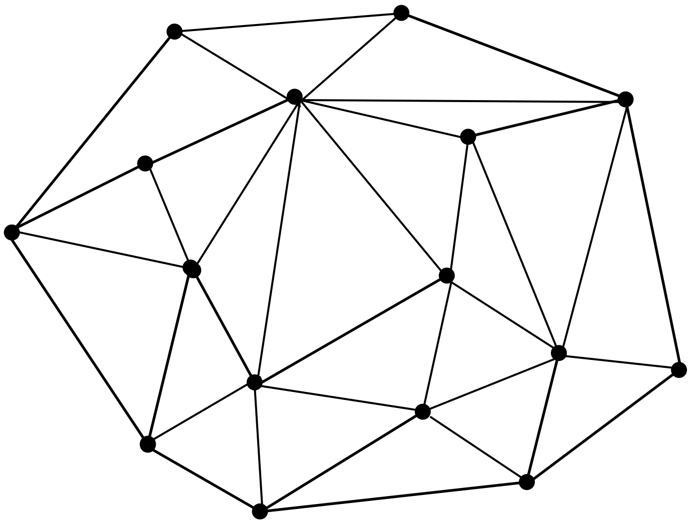


(б)

Мал.. 5.3. Сумарна довжина ребер дерева Штейнера (б) може бути меншою, ніж у МОД.

Задача Б.4 (ТРИАНГУЛЯЦІЯ). *На площині задано N точок. З'єднати їх неперетинаючими відрізками таким чином, щоб кожна область всередині опуклої оболонки цієї множини точок являлась трикутником.*

Граф тріангуляції множини із N точок, являючись планарним, має не більше 3N – 6 ребер. Результатом розв'язання сформульованої вище задачі повинен бути принаймні список цих ребер. Приклад тріангуляції мал.5.4.



Мал..5.4. Тріангуляція множини точок.

Задача Б.5 (ПОШУК НАЙБЛИЖЧОГО СУСІДА). *На площині задано N точок. Як швидко можна знайти найближчого сусіда для деякої нової точки q, при умові що допускається попередня обробка?*

У просторі розмірності d цю задачу можна розв'язати за час O(dN).

Задача Б.6 (k- НАЙБЛИЖЧИХ СУСІДІВ). *На площині задано N точок. Як швидко можна знайти k – точок, найближчих до деякої нової точки q, при умові що допускається попередня обробка ?*

5.2. НИЖНІ ОЦІНКИ СКЛАДНОСТІ .ЗАДАЧІ ПРОТОТИПИ.

Означення. *Задачі, з визначеними оцінками складності, які можна звести до розглядуваних називаються* ***задачами – прототипами****. Вони є базовими для певних класів.*

В обчислювальній геометрії використовуються три важливі задачі-прототипи, які мають невелику складність $\Omega(N\log N)$:*сортування, визначення крайніх точок* та перевірка *унікальності елементів множини*.

Задача.(УНІКАЛЬНІСТЬ ЕЛЕМЕНТІВ). *Задані N дійсних чисел. Визначити, чи існують серед них хоча б два рівних.*

Нижня оцінка часової складності для задачі УНІКАЛЬНІСТЬ ЕЛЕМЕНТІВ визначається в рамках моделі алгебраїчних дерев розв'язків.

Множину $\{x_1, \ldots, x_N\}$із N дійсних чисел можна розглядати як точку $(x_1, \ldots, x_N)$ в E^N. Використовуючи термінологію, введену в розділі 1.4 позначимо через $W \subseteq E^N$ множина істинності для задачі УНІКАЛЬНІСТЬ ЕЛЕМЕНТІВ на множині $\{x_1, \ldots, x_N\}$ (т.т. W містить усі точки, будь-яка пара координат яких різна). Стверджується, що W має N! компонент зв'язності. Дійсно, будь-якій перестановці π множини $\{x_1, \ldots, x_N\}$ відповідає множина точок у E^N

$W_\pi=\{(x_1, \ldots, x_N): x_{\pi(1)} < x_{\pi(2)}<\ldots< x_{\pi(N)}\}$.

Зрозуміло, що $W= \cup W_\pi$, по усім π.

І при цьому W_π являються компонентами зв'язності, а #(W)=N!.

Звідси як наслідок теореми 1.2. одержимо:

Теорема 8.1. *В рамках моделі алгебраїчних дерев обчислень будь-який алгоритм, який визначає, чи є елементи множини із N дійсних чисел різними, вимагає* $\Omega(N\log N)$ *перевірок.*

Нижні оцінки складності задач на близькість.

1. *ПОШУК НАЙБЛИЖЧОГО СУСІДА*

ДВІЙКОВИЙ ПОШУК $\propto_{o(1)}$ НАЙБЛИЖЧИЙ СУСІД

Нехай задано N дійсних чисел $x_1, \ldots, x_N$. В результаті двійкового пошуку визначається число x_i, найближче до числа q, яке задається в запиті на пошук. (При цьому допускається попередня обробка!). Але цю саму задачу можна подати і в геометричному формулюванні, поставивши у відповідність кожному числу x_i точку на площині з координатами $(x_i, 0)$. І таким чином, пошук найближчого сусіда приведе до тієї ж відповіді, що і двійковий пошук.

Терема 5.1. *Для пошуку найближчого сусіда точки в просторі довільної розмірності необхідно виконати* Ω(logN) *порівнянь (в гіршому випадку).*

2. *k- НАЙБЛИЖЧИХ СУСІДІВ*.

ПОШУК НАЙБЛИЖЧОГО СУСІДА ∝ k- НАЙБЛИЖЧИХ СУСІДІВ.

Що стосується задачі k- НАЙБЛИЖЧИХ СУСІДІВ, то, поклавши k=1, відразу одержимо зведення ПОШУК НАЙБЛИЖЧОГО СУСІДА ∝ k- НАЙБЛИЖЧИХ СУСІДІВ. Таким чином, теорема 5.1 застосовується і до задачі k- НАЙБЛИЖЧИХ СУСІДІВ.

3. *ЗАДАЧІ Б.1-Б.4.*

На малюнку 5.6 показана діаграма зводимості задач (на діаграмі символ ∝ замінений стрілкою).

$$\begin{tikzcd}
 & \textit{УНІКАЛЬНІСТЬ ЕЛЕМЕНТІВ} \arrow[r] \arrow[dd, "N"'] & \textit{НАЙБЛИЖЧА ПАРА (3.1)} \arrow[d] \\
 & & \text{УСІ НАЙБЛИЖЧІ СУСІДИ(3.2)} \\
\textit{СОРТУВАННЯ} \arrow[r, "N"] \arrow[dr] & \textit{ЕМОД (3.3)} & \\
 & \text{ТРИАНГУЛЯЦІЯ(3.4)} &
\end{tikzcd}$$

Мал. 5.6. Зв’язок задач про близькість з основними задачами, які використовуються як обчислювальні прототипи.

3.1. *УНІКАЛЬНІСТЬ ЕЛЕМЕНТІВ* $\propto_N$ *НАЙБЛИЖЧА ПАРА.*

Нехай задана множина дійсних чисел $\{x_1, \ldots, x_N\}$. Розглядатимемо їх як точки на прямій y=0, намагаючись знайти найближчу пару точок ($\{x_1, \ldots, x_N\} \propto_N \{(x_1,0),\ldots, (x_N,0)\}$). Якщо відстань між точками, які утворюють найближчу пару, не дорівнює нулю, то усі точки множини різні (d=0⇒”НІ”, d≠0⇒”ТАК”). Так як множину точок, задану в одновимірному просторі, завжди можна вкласти в простір розмірності k, то природно одержується узагальнення цього зведення.

3.2. НАЙБЛИЖЧА ПАРА $\propto_N$ УСІ НАЙБЛИЖЧІ СУСІДИ.

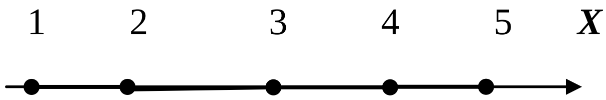
Зведення НАЙБЛИЖЧА ПАРА $\propto_N$ УСІ НАЙБЛИЖЧІ СУСІДИ очевидне, так, як одна із пар, одержаних в результаті розв'язання попередньої задачі, буде найближчою і вона може бути визначена за допомогою O(N) порівнянь.

3.3. НАЙБЛИЖЧА ПАРА $\propto_N$ ЕМОД(Б.3).

Так як, ЕМОД містить найкоротше ребро евклідового графа на множині із N точок, то задача НАЙБЛИЖЧА ПАРА тривіально зводиться до ЕМОД за лінійний час.

3.4. СОРТУВАННЯ $\propto_N$ ЕМОД.

Розглянемо множину із N дійсних чисел $\{x_1, \ldots, x_N\}$. Розглядаючи кожне число x_i як точку $(x_i, 0)$ на площині ($\{x_1, \ldots, x_N\} \propto_N \{(x_1,0),\ldots, (x_N,0)\}$), побудуємо відповідну множину ЕМОД.

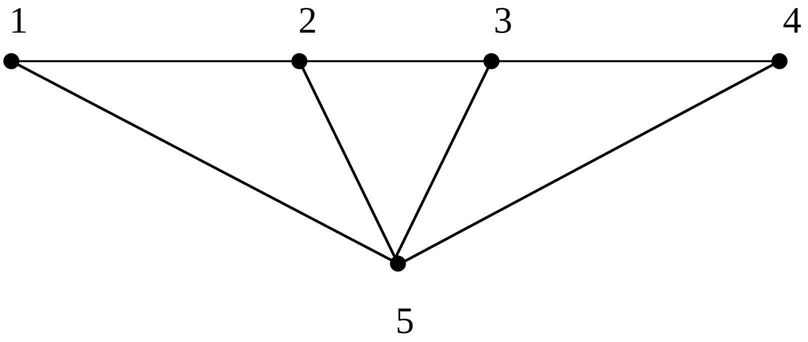


ЕМОД: (1,2), (2,3), (3,4), (4,5) ⇒ *(O(N))* (1, 2,3 4, 5,)

В одержаному ЕМОД вершини, які відповідають числам x_i і x_j, з'єднані ребром ⇒ коли x_i і x_j утворюють пару послідовних чисел у впорядкованій множині. Розв'язком задачі ЕМОД є список, який містить N-1 пар (*I,j*), кожна із яких визначає ребро дерева. Не важко перетворити цей список в упорядкований список чисел x_i, витративши на це часу O(N).

3.5. СОРТУВАННЯ $\propto_N$ ТРИАНГУЛЯЦІЯ(Б.4).

Розглянемо множину із N точок $\{x_1, \ldots, x_N\}$, показану на малюнку 5.7. N-1 точок множини лежать на одній прямій, одна із точок розташована за межами прямої ($\{x_1, \ldots, x_N\} \propto_N \{(x_1,0),\ldots, (x_{N-1},0), (x_N,-a)\}$). Тріангуляція цієї множини може бути виконана згідно малюнку єдиним способом. Списком ребер, який породжується алгоритмом тріангуляції, можна скористатись для одержання упорядкованого списку чисел x_i, витративши на це додатково O(N) операцій. Таким чином, необхідно зробити Ω(NlogN) порівнянь.



Мал.. 5.7. Ілюстрація до одержаної нижньої оцінки складності розв'язання задачі ТРИАНГУЛЯЦІЯ.

Зауваження. Еквівалентне зведення УПОРЯДКОВАНА ОБОЛОНКА $\propto_N$ ТРІАНГУЛЯЦІЯ- базується на тому, що тріангуляція множини S є планарний граф, зовнішня границя якого є опукла оболонка множини S.

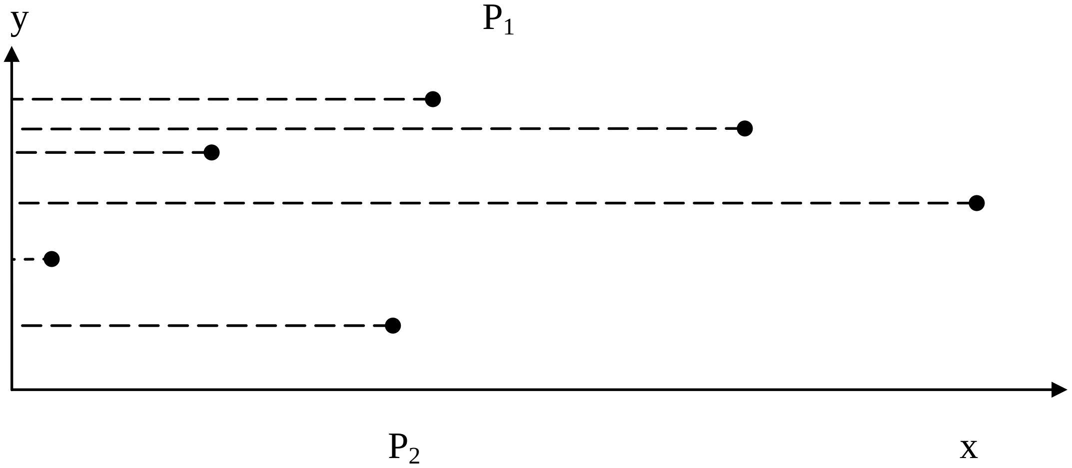
Враховуючи, що в рамках моделі дерев обчислень обидві задачі УНІКАЛЬНІСТЬ ЕЛЕМЕНТІВ і СОРТУВАННЯ на множині із N елементів мають нижню оцінку складності Ω(NlogN), має місце така теорема.

Теорема 5.2. *В рамках моделі дерев обчислень будь-який алгоритм, який розв'язує одну із задач* - НАЙБЛИЖЧА ПАРА, ЕМОД, ТРИАНГУЛЯЦІЯ, УСІ НАЙБЛИЖЧІ СУСІДИ- *вимагає* Ω(NlogN) *операцій.*

5.3. Розв'язання задачі про найближчу пару методом "розподіляй та володарюй".

Нижня оцінка задачі найближча пара має складність Ω (N log N). Для побудови алгоритмів з такою оцінкою є два шляхи: безпосереднє використання сортування і використання метода ''розподіляй та володарюй''.

Перший підхід можна зразу відкинути, так як сортування зручне лише в умовах повної впорядкованості, яка полягає у проектуванні усіх точок на деяку пряму, але при цьому втрачається суттєва в даному випадку інформація. Це демонструє малюнок 5.8., на якому точки p_1, p_2 утворюють найближчу пару, але при цьому дають максимальну відстань при проектуванні на вісь *y*.



Мал..5.8. Точки p_1 і p_2, які утворюють найближчу пару мають найбільшу відстань по *y*-координаті.

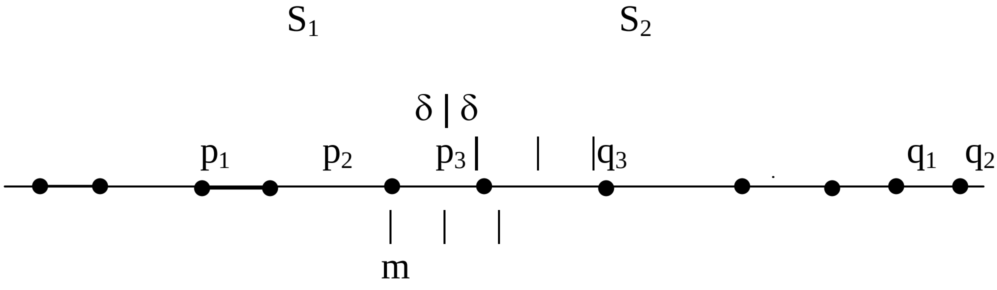
Другий шлях для досягнення складності θ (N log N) полягає в розбитті задачі на дві підзадачі, розв'язок яких можна об'єднати за лінійний час, отримавши рішення вихідної задачі. Позначивши через P(N, 2) час роботи алгоритму, який шукає найближчу пару точок на площині, отримаємо рекурентне співвідношення:

$$P(N, 2) = 2P(N/2, 2) + O(N^2).$$

Розв'язком цього співвідношення є $P(N, 2) = O(N^2)$, що не дає бажану оцінку.

Алгоритм “розподіляй та володарюй”

Одновимірний випадок. Алгоритм впорядковує точки множини, а потім проглядає його за лінійний час.



Мал..5.9. Метод “розподіляй та володарюй” в одновимірному випадку.

Нехай точка *m* розбиває множину на дві підмножини S_1 та S_2 і при цьому $p < q$ для всіх $p \in S_1$ і $p \in S_2$. Розв’язавши окремо рекурсивно задачу про найближчу пару для множин S_1 та S_2, отримаємо дві найближчі пари точок $\{p_1, p_2\}$ і $\{q_1, q_2\}$, відповідно.

Нехай $\delta_1 = \min(S_1)=| p_2 - p_1 |$ і $\delta_2 = \min (S_2)=| q_2 - q_1 |$ відстані для знайдених пар, відповідно. Позначимо через δ найменшу серед знайдених δ_1 і δ_2 відстань:

$$\delta = \min (\delta_1, \delta_2) = \min(| p_2 - p_1 |, | q_2 - q_1 |).$$

Найближчою парою є $\{p_1, p_2\}$, або $\{q_1, q_2\}$, або $\{p_3, q_3\}$, де $p_3=\max(S_1)$, $q_3=\min(S_2)$.

Для того щоб відстань, яку дає пара $\{p_3, q_3\}$, була менше δ, p_3 і q_3 повинні бути на відстані, яка не перевищує δ від точки m ($|p_3 – q_3| < \delta \Rightarrow |p_3 – m| < \delta$ або $|q_3 – m| < \delta$). Відкладемо ліворуч і праворуч відносно точки m відрізки довжиною δ.

Скільки ж точок множини S_1 можуть міститись в інтервалі $(m - \delta, m]$? Так як кожен напіввідкритий інтервал довжиною δ містить не більше однієї точки множини S_1, то інтервал $(m - \delta, m]$ містить не більше однієї точки. Аналогічно інтервал $[m, m + \delta)$.Очевидно, що усі точки, які потрапляють в інтервали $(m - \delta, m]$ та $[m, m + \delta)$, можна визначити, переглянувши множину за лінійний час. Отже визначивши

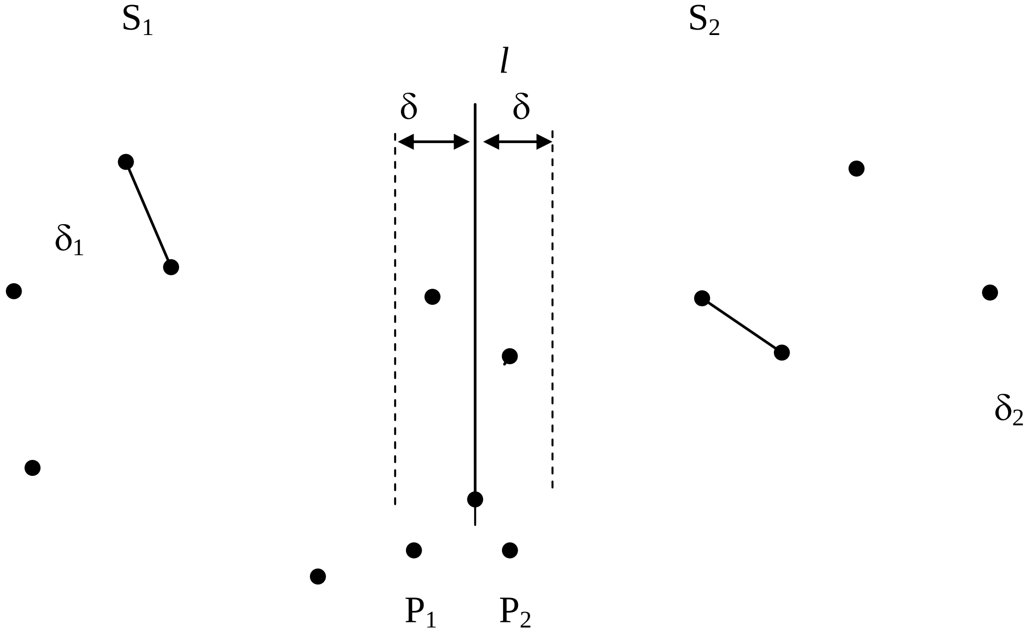
$\mathrm{dist}(\max(S_1), \min(S_2)) = |p_3 – q_3|$ знайдемо остаточно

$$\delta^* = \min (\delta(S_1), \delta(S_2), \mathrm{dist}(\max(S_1), \min(S_2))) = \min(| p_2 - p_1 |, | q_2 - q_1 |, |p_3 – q_3|)$$

а значить і найближчу пару точок. Таким чином отримаємо наступний алгоритм зі складністю O(N log N):

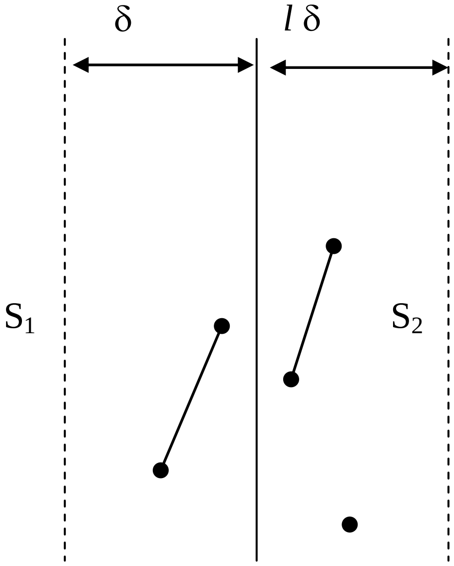
Двовимірний випадок. Узагальнення на двовимірний випадок можна виконати безпосередньо. Розіб'ємо множину точок на площині S на дві підмножини S_1 і S_2 вертикальною прямою l , яка є *медіаною* множини S по *x*-координаті, так, щоб кожна точка S_1 лежала лівіше будь-якої точки S_2. Розв'язавши рекурсивно задачу для S_1 і S_2 , одержимо числа δ_1, δ_2 – мінімальні відстані для множин S_1 і S_2 відповідно. Покладемо $\delta = \min(\delta_1, \delta_2)$.

Якщо найближчу пару утворюють точки $p \in S_1$ і $q \in S_2$, то, відстань від точок *q* і *p* до *l* не перевищує δ.Позначимо через P_1 і P_2 вертикальні смуги шириною δ, розташовані відповідно ліворуч та праворуч від *l*, то $p \in S_1$ і $q \in S_2$, мал.5.10



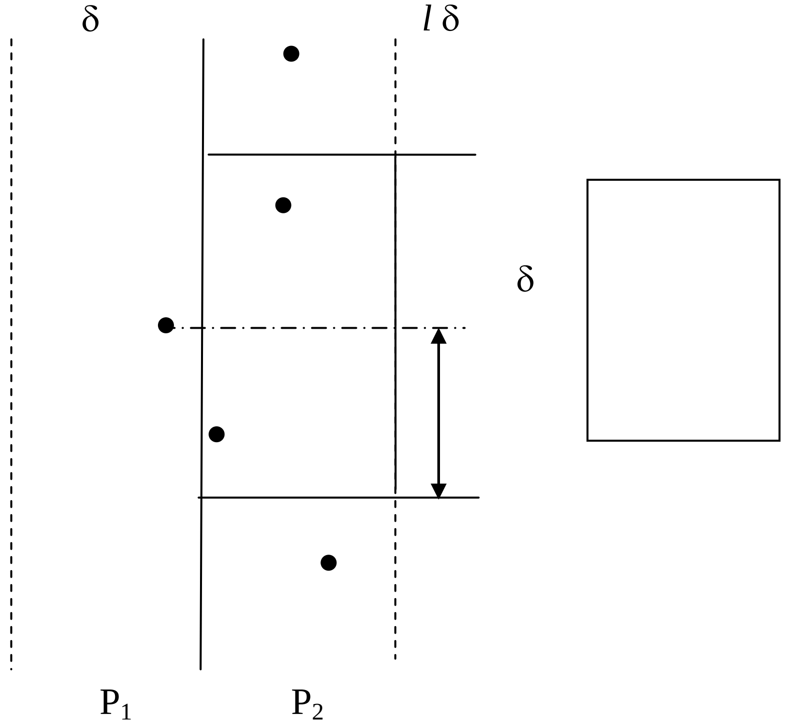
Мал..5.10. Метод "розподіляй та володарюй" у випадку площини.

На прямій було не більше одного кандидата для *q* і *p*. У процедурі БПАРА1 є точно один кандидат для *p*: *p*=max(S_1). На площині таким кандидатом може бути будь-яка точка, якщо вона знаходиться на відстані не більшій за δ, від прямої *l*. На мал.. 5.11 наведено приклад такої множини.



P_1 P_2

Мал..5.11. Усі точки можуть знаходитсь на відстані, яка не перевищує δ від прямої *l*.



Мал..5.12. Для кожної точки із P_1, необхідно перевірити не більше шести точок із P_2.

Розглянемо в смузі P_1 довільну точку *p*. Необхідно знайти усі точки *q* із P_2, які віддалені від *p* не більше ніж на δ. Усі вони розташовуються у прямокутнику R розміром δ×2δ. Максимальна кількість точок, які можна помістити в такий прямокутник так, щоб відстань між ними була не менша за δ, рівна 6. Це означає, що для кожної точки із P_1 необхідно досліджувати лише не більше 6 точок із P_2 . Тому на кроці злиття розв'язків підзадач необхідно виконати не більше 6×N/2=3N порівнянь відстаней.

Спроектуємо точку *p* і усі точки із P_2 на пряму *l*. Для визначення точок із P_2, які потрапили в R, можна розглянути лише проекції точок, які на відстані не більшій за δ від проекції точки *p*. Якщо точки впорядковані по y-координаті, то для усіх точок із P_1 «кандидати» на місце їх найближчого сусіда із P_2 визначаються за один прохід впорядкованого списку.

procedure НПАРА2(S)

1. розбити S на дві підмножини S_1 та S_2 вертикальною прямою *l*(медіаною).
2. Рекурсивно знайти відстань для найближчих пар δ_1 та δ_2.
3. $\delta := \min(\delta_1, \delta_2)$.
4. Нехай P_1 - множина точок із S_1, які лежать в смузі на відстані δ від розділяючої прямої *l*, а P_2 - аналогічна підмножина в S_2. Спроектувати P_1 та P_2 на *l* та впорядкувати проекції по y-координаті. Нехай P_1^* та P_2^* - відповідні впорядковані послідовності.
5. ''Злиття'' можна виконати, переглядом кожної точки з P_1^*, вивчаючи точки з P_2^*, які на відстані не перевищуючій δ. Поки вказівник просувається послідовністю P_1^*, вказівник на P_2^* переміщується вперед-

назад, лишаючись в інтервалі шириною 2δ. Нехай δ_l - мінімальна відстань між парою точок.

6. $\delta_S := \min(\delta, \delta_l)$.

Якщо позначити через T(N) час обробки алгоритмом множину із N точок, то час, який пішов на обробку на кроці 1 та 5 дорівнює O(N), на кроці 3 та 6 O(1), а крок 2 потребує часу 2T(N/2). Скориставшись попереднім сортуванням для часу обробки P(N, 2) алгоритму пошуку найближчої пари отримаємо співвідношення:

$$P(N, 2) = 2P(N/2, 2) + O(N) = O(N \log N).$$

На основі співвідношення маємо теорему:

Теорема. *Найкоротша відстань, яка визначається N точками на площині, може бути знайдена за час* $\theta(N \log N)$ *і є оптимальна.*

```
function НПАРА1(S)
begin if (| S | = 2) then $\delta$:= | X[2] - X[1] |
      else if (| S | = 1) then $\delta$:= $\infty$
        else  begin m:= Медіана(S);
             Побудувати ($S_1$, $S_2$) (*$S_1 = \{p: p \le m\}$,  $S_2 = \{p: p > m\}$*);
             $\delta_1$:= НПАРА($S_1$);
             $\delta_2$:= НПАРА($S_2$);
             p:= max ($S_1$);
             q:= min ($S_2$);
             $\delta$:= min ($\delta_1$, $\delta_2$, q - p)
      end;
   return $\delta$
end.
```