ЛЕКЦІЯ 4

ЗАДАЧІ РЕГІОНАЛЬНОГО ПОШУКУ

3.4.1. Загальні зауваження.

Запит у цій моделі визначає область (регіон) у d -мірному просторі, а результатом пошуку є *звіт про множину точок файлу*, яка міститься у цій області (***запит у режимі звіту***). Іншою метою пошуку є *підрахунок числа точок в області запиту* (***запит у режимі підрахунку***).

Припустимо, що файл є фіксований набір S із N записів. Відповіддю на запит буде $S' \subset S$. Відповіддю на запит у режимі підрахунку завжди буде єдине ціле число (k- потужність S'), в той час як відповіддю на запит у режимі звіту будуть імена усіх k елементів S'. Можна виділити два види дій при обробці запиту:

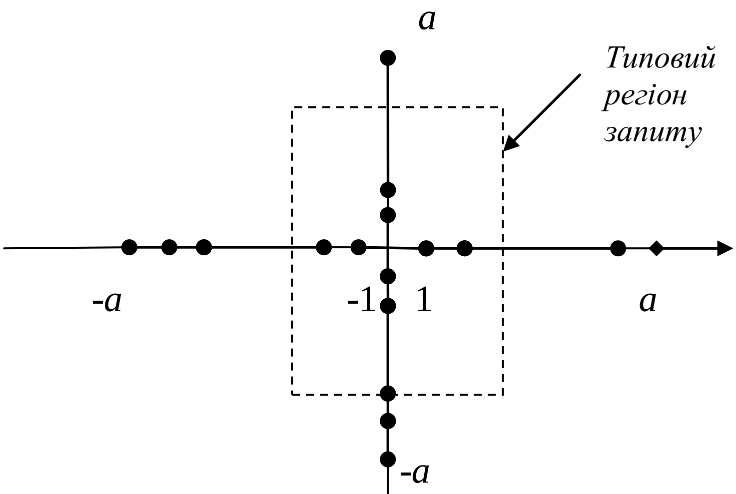
1. *Пошук* , т.т. дії, які приводять до елементів шуканої підмножини (зазвичай, послідовність порівнянь).
2. *Вибірка,* т.т. дії по складанню відповіді на запит (для запитів у режимі звіту - це фактичне визначення шуканої підмножини).
   - при аналізі запитів у режимі підрахунку всю обчислювальну роботу вбирає в себе "пошук"; при цьому очікується, що час запиту для гіршого випадку буде функцією f(N,d), яка залежить від розміру файла та розмірності;
   - при аналізі в режимі звіту, обчислювальна робота є комбінацією пошуку й вибірки, і та її частина, яка зв'язана з вибіркою, обмежена знизу числом k- потужністю S'.

   В загальному випадку верхня оцінка часу запиту в режимі звіту така - $O(f(N,d)+k\cdot g(N,d))$; f(N,d) - "час пошуку" і $k\cdot g(N,d)$ - "час вибірки".

   $\Omega(k)$ - тривіальна нижня оцінка *часу вибірки*; $\Omega(\log_2 Q(S))$ – нижня оцінка кількості порівнянь для *пошуку* (використовується модель двійкового дерева розв'язків, яка, підраховує число порівнянь, необхідних для доступу до елементів множини $S' \subset S$ у середині регіону запиту), де Q(S)- число різних підмножин S , отриманих як відповіді на запити.

   Розглянемо: S -множина усіх точок, які мають одну і лише одну ненульову цілочисельну координату в інтервалі [-*a*,*a*]; N=2*a*d (мал. 2.26). $x_1^i \in [-a, -1]$ і $x_2^i \in [1, a]$.

Цей вибір можна здійснити $a^{2d} = (N/(2d))^{2d}$ способами;$\Rightarrow$ нижня оцінка числа двійкових розв'язків рівна $\Omega(\log (N/(2d))^{2d}) = \Omega(d\log (N))$. $\Omega(Nd)$- тривіальна нижня оцінка пам'яті, яка зайнята під структуру даних пошуку. Для усіх алгоритмів, які описуються, час пошуку виражається у формі $O(f(N,d)+k)$.



Мал.2.26. Приклад множини S при d=2.

<u>Одновимірний регіональний пошук.</u>

Множина з N точок на осі *x* являє файл, а запитним регіоном є відрізок [*x'*,*x''*] (який називається *x*- регіоном). Ефективний регіональний пошук реалізується через метод, який базується на методі *двійкового пошуку.* Структура даних, яка забезпечує вказану дію, є *прошите двійкове дерево*, т.т. таке збалансоване двійкове дерево, листки якого додатково зв'язані у списку, який виражає порядок абсцис; дерево і список обробляються на фазах відповідного пошуку й вибірки. Оцінки: оптимальна як по часу запиту θ(logN+k), так і по пам'яті θ(N).

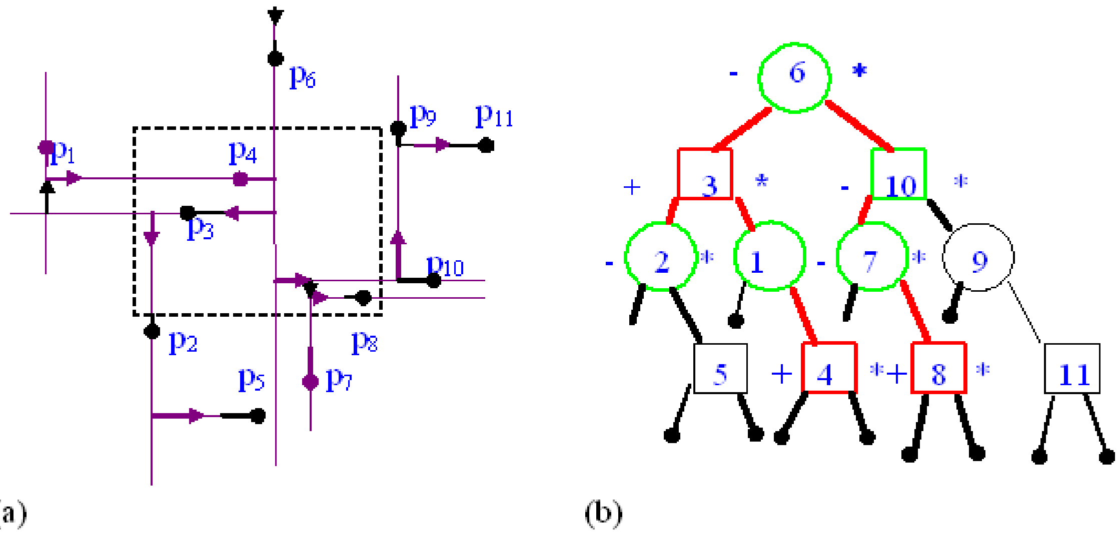
3.4. 2 МЕТОД БАГАТОВИМІРНОГО ДВІЙКОВОГО ДЕРЕВА (2-D-дерева)

Назвемо (узагальненим) *прямокутником* таку область на площині, яка визначена декартовим добутком $[x_1, x_2]\times[y_1,y_2]$ x-інтервалу $[y_1,y_2]$, включаючи граничні випадки, коли в любій комбінації допускається: $x_1 = -\infty$, $x_2 = \infty$, $y_1 = -\infty$, $y_2 = \infty$.

Побудуємо два відсортовані списки: $U_x = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}\}$, $U_y = \{p_5, p_7, p_2, p_8, p_{10}, p_3, p_4, p_1, p_{11}, p_9, p_6\}$.

Процес розбиття S шляхом розрізання площини краще всього ілюструвати в поєднанні з побудовою двовимірного двійкового дерева T: $\forall v\in T$ $\exists(R(v), S(v) \subseteq S, P(v)$ і "сікуча пряма" $l(v))$. Процес подрібнення завершиться, коли з'явиться прямокутник, який не містить всередині жодної точки, відповідний йому вузол являється листком дерева T.

Вузли трьох різних типів і мають позначення: *кружки* - не листові вузли з вертикальною лінією розрізу, *квадрати* - не листові вузли з горизонтальною лінією розрізу, *точки* - листки. Така структура даних часто називається *2-Д-деревом.*



Мал.2.27. Ілюстрація метода пошуку за допомогою двовимірного двійкового дерева.

Алгоритм регіонального пошуку.

Алгоритмічна схема - метод " розділяй та володарюй". Любий вузол *v* завантажений такими параметрами: P(v), *R*(v), S(v), *l(v)*, D**.**

Де P(v)- поточна точка, *R*(v)- прямокутна область, S(v)- множина точок яка належить *R*(v), *l(v)*- розрізаюча пряма, яка проходить через точку P(v),D – запитний регіон.

Від взаємного розташування *R*(v), S(v), *l(v)*, D для кожного вузла v $\in$ T, залежить регіональний пошук: $\forall$v $\in$ T пряма *l(v)* розбиває $R(v) = R_1(v) \cup R_2(v)$, $R_1(v) \cap D$, $R_2(v) \cap D$.

1)$D \cap R_1(v) \neq \varnothing$ і $R_2(v) \cap D = \varnothing \Rightarrow$ лівий пошук (у $D \cap R_1(v)$),;

2)$D \cap R_2(v) \neq \varnothing$ і $R_1(v) \cap D = \varnothing \Rightarrow$ правий пошук (в $D \cap R_2(v)$);

3) $D \cap R_1(v) \neq \varnothing$ і $R_2(v) \cap D \neq \varnothing \Rightarrow$ лівий (в $D \cap R_1(v)$) і правий (у $D \cap R_2(v)$); перевірка $P(v) \notin D$.

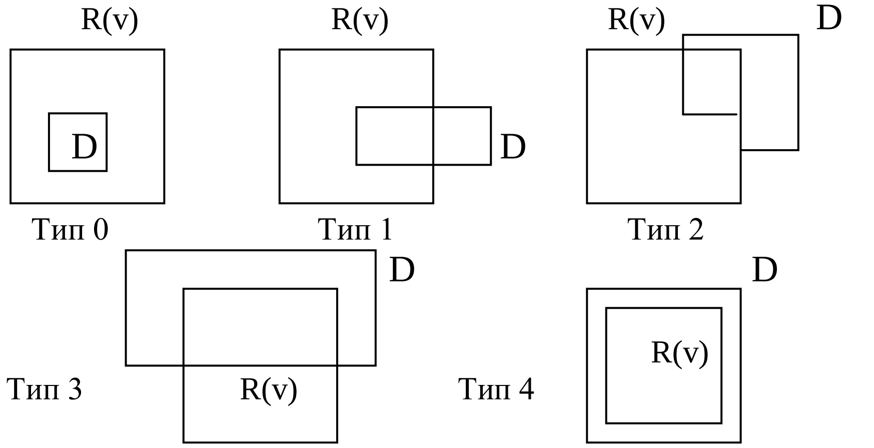
Більш строго. $\forall$v $\in$ T:(P(v), t(v), M(v)): (t(v), M(v)) визначають пряму *l(v):* t(v) визначає горизонтальність (y=M(v) (M(v)= y(P(v))) чи вертикальність (x=M(v) (M(v)= x(P(v))) *l(v);* алгоритм накопичує точки у списку U- зовнішньому до процедури і спочатку порожньому. Позначимо через $D = [x_1, x_2] \times [y_1, y_2]$ запитний регіон, тоді має місце:

procedure ПОШУК(v, D);
begin if (t(v) = вертикаль) **then** [l, r]:= $[x_1, x_2]$ **else** [l, r]:= $[y_1, y_2]$;
    if ($l \leq M(v) \leq r$) **then if** ($P(v) \in D$) **then** $U \Leftarrow P(v)$;
    if (v $\neq$ листок) **then**
        begin if (l < M(v)) **then** ПОШУК(ЛСИН[v], D);
            if (M(v) < r) **then** ПОШУК(ПСИН[v], D)
        end
end.

Оцінка складності: пам'ять - O(N), побудова дерева - O(N log N) (вертикальний розріз множини S проводиться в результаті обчислення медіани множини x- координат точок з S за час O(|S|), і шляхом формування розбиття S з такою ж оцінкою часу;аналогічно і для горизонтального розрізу; за час O(N) вихідна множина розбивається, в результаті чого отримуємо півплощини, в кожній із яких по N/2 точок; тривіально отримуємо рекурентне відношення для часу T(N) роботи алгоритму побудови дерева:

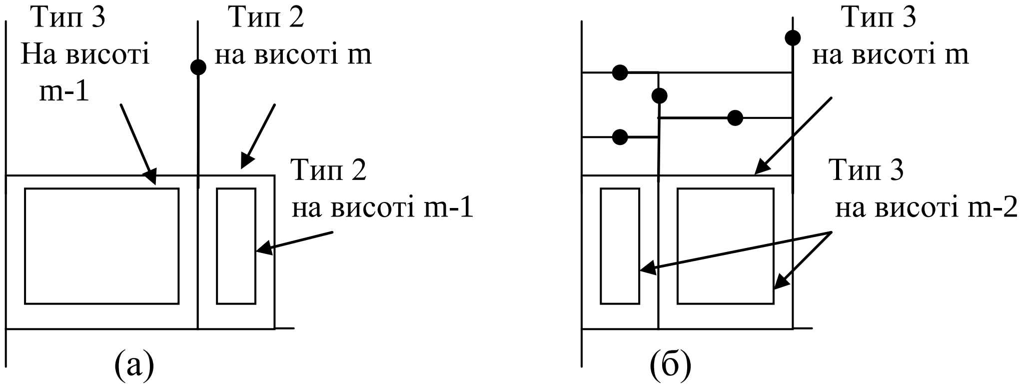
$$T(N) \le 2T(N/2) + O(N).$$

Аналіз часу запиту для *найгіршого випадку.* Якщо час витрачений у вузлі v не даремно, то точка P(v) вибирається (*продуктивний вузол*); інакше, цей вузол є *непродуктивним.* Перетин запитного регіону D з R(v) може бути віднесений до різних "типів" у залежності від числа сторін R(v) , які мають не порожні перетини з D (мал.2.29).



Мал. 2.29. Приклади різних типів перетинів D і R(v) , коли $D \cap R(v) \neq \varnothing$.

На мал. 2.30 (а),(б) розглянуто ситуації, коли перетин типу 2 на висоті m непродуктивний і породжує перетин типу 2 і перетин типу 3 на висоті (m -1) (обидва можна зробити непродуктивними) та, при тому самому обмежені на типи непродуктивних вузлів , коли перетин типу 3 на висоті m в T породжує пару перетинів типу 3 на висоті (m -2), відповідно.



Мал.2.30.Ситуації, які відповідають непродуктивним вузлам в T.

Позначаючи $U_i(m)$- число пройдених непродуктивних вузлів у піддереві висотою m, корінь якого має тип $i(i=2,3)$, одержимо рекурентні співвідношення:

$$U_2(m)= U_2(m-1)+ U_3(m-1)+1$$
$$U_3(m)=2U_3(m-2)+3 \qquad (2.3)$$

Результат же такий: для $U_2(m)$ і $U_3(m-1)$ оцінка однакова-$O(\sqrt{N})$.

Теорема . *Для файлу потужністю N у найгіршому випадку час запиту становить $O(\sqrt{N})$, навіть якщо вибрана множина виявиться порожньою.*

Результуюче розбиття d-вимірного простору моделюється двійковим деревом із N вузлами, яке називається *багатовимірним двійковим деревом* або *k-D-деревом.* Аналіз ефективності також можна узагальнити на випадок d вимірів.

Теорема. *За допомогою метода k-D-дерева регіональний пошук на d-вимірній множині (при $d \geq 2$) з N точок можна провести за час $O(dN^{1-1/d} + k)$ із використанням O(dN) пам'яті, якщо витратити O(dN log N) часу на попередню обробку. Через це метод k-D-дерева дає $(dN, dN^{1-1/d})$-алгоритм.*

3.4.3. МЕТОД ДЕРЕВА РЕГІОНІВ

В d=2 будь-який відрізок розбивається на більше ніж три стандартні відрізки. Мінімізація кількості стандартних відрізків - головна ідея методу *дерева регіонів.*

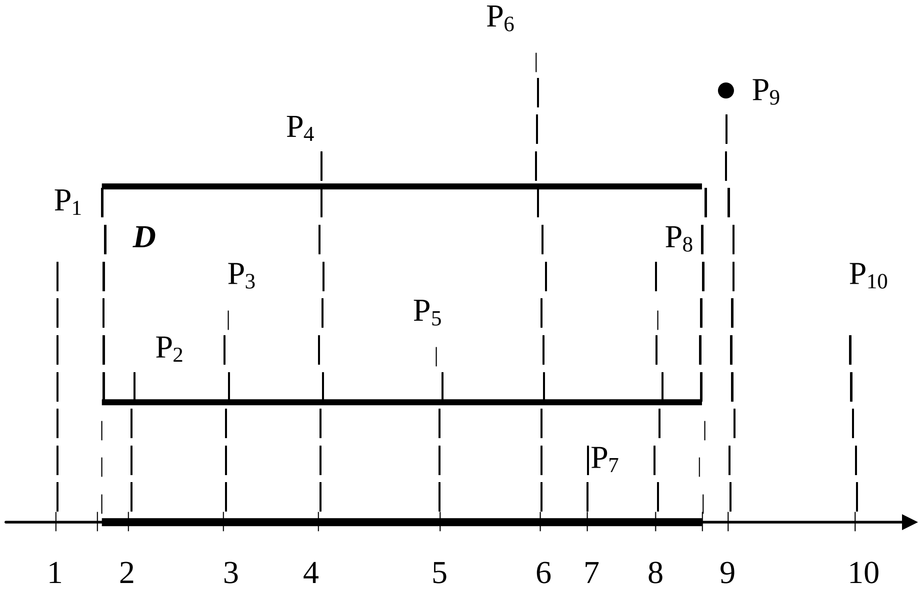
Розглянемо множину на осі x, яка складається з N абсцис, які нормалізуються до нормальних чисел з інтервалу [1, N] по всій величині. N абсцис визначають N -1 елементарних відрізків [i, i + 1] для i = 1, 2, ... ,N - 1.

Будь-який відрізок, кінці якого належать множині із N заданих абсцис, може бути розбитий деревом відрізків T(1, N) на більше ніж $2\lceil \log_2 N\rceil - 2$ стандартних відрізки. Кожен стандартний відрізок віднесений до одного з вузлів T(1, N), а ті вузли, які визначають розбиття відрізка [i, j], називаються *вузлами віднесення.*

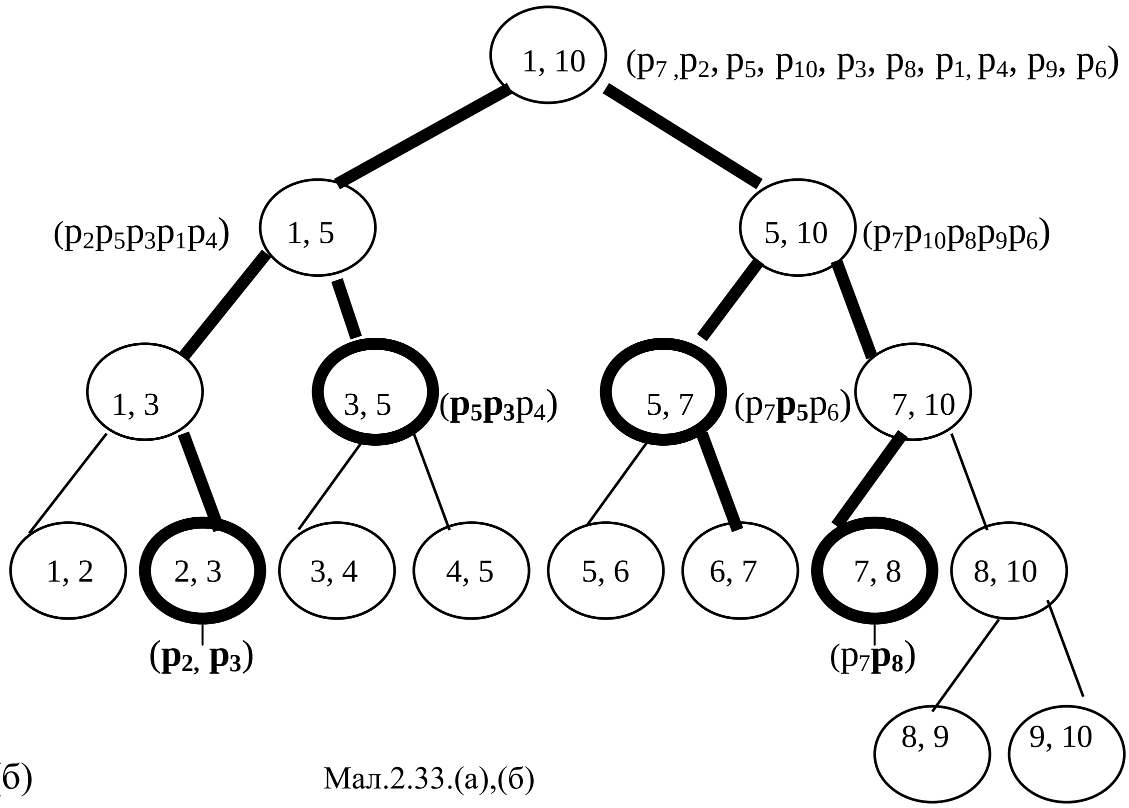
Дерево відрізків T(1, N) використовується при пошуку по x-координаті. Цей пошук визначає універсальну множину вузлів (*вузлів віднесення*). Кожен такий вузол v відповідає множині із (E[v] - B[v]) абсцис, тобто, множині з (E[v] - B[v]) точок на площині. Ординати цих точок утворюють звичайне двійкове дерево для регіонального пошуку в y-напрямку. В результаті побудована нова структура даних, називається *деревом регіонів*; його первинною структурою є дерево відрізків, заданих на абсцисах точок вихідної множини S. В кожному вузлі цього дерева є вказівник на прошите двійкове дерево пошуку (вторинні структури), мал.2.33, 2.34.Дерево регіонів будується рекурсивно:

(1) Початкове дерево відрізків T* відповідає множині $\{x_1(p)\colon p \in S\}$. Для кожного вузла v із T* позначимо через $S_d(v)$- множина точок із S. Визначимо (d -1)-вимірну множину: $S_{d-1}(v) = \{(x_2(p), \ldots, x_d(p))\colon p \in S_d(v)\}$.

(2) Вузол v із T* має вказівник на дерево регіонів для $S_{d-1}(v)$.

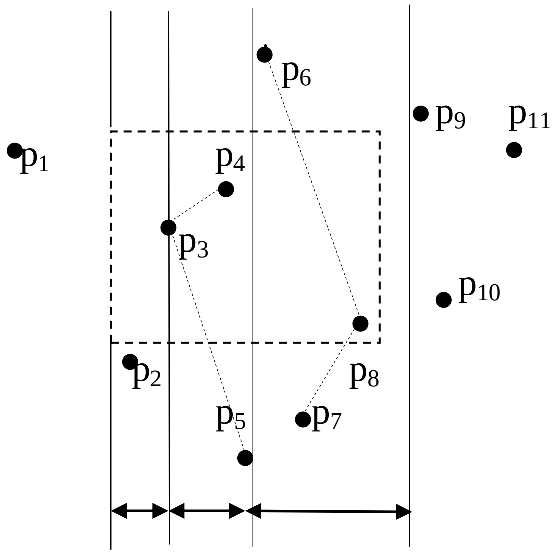


(а)

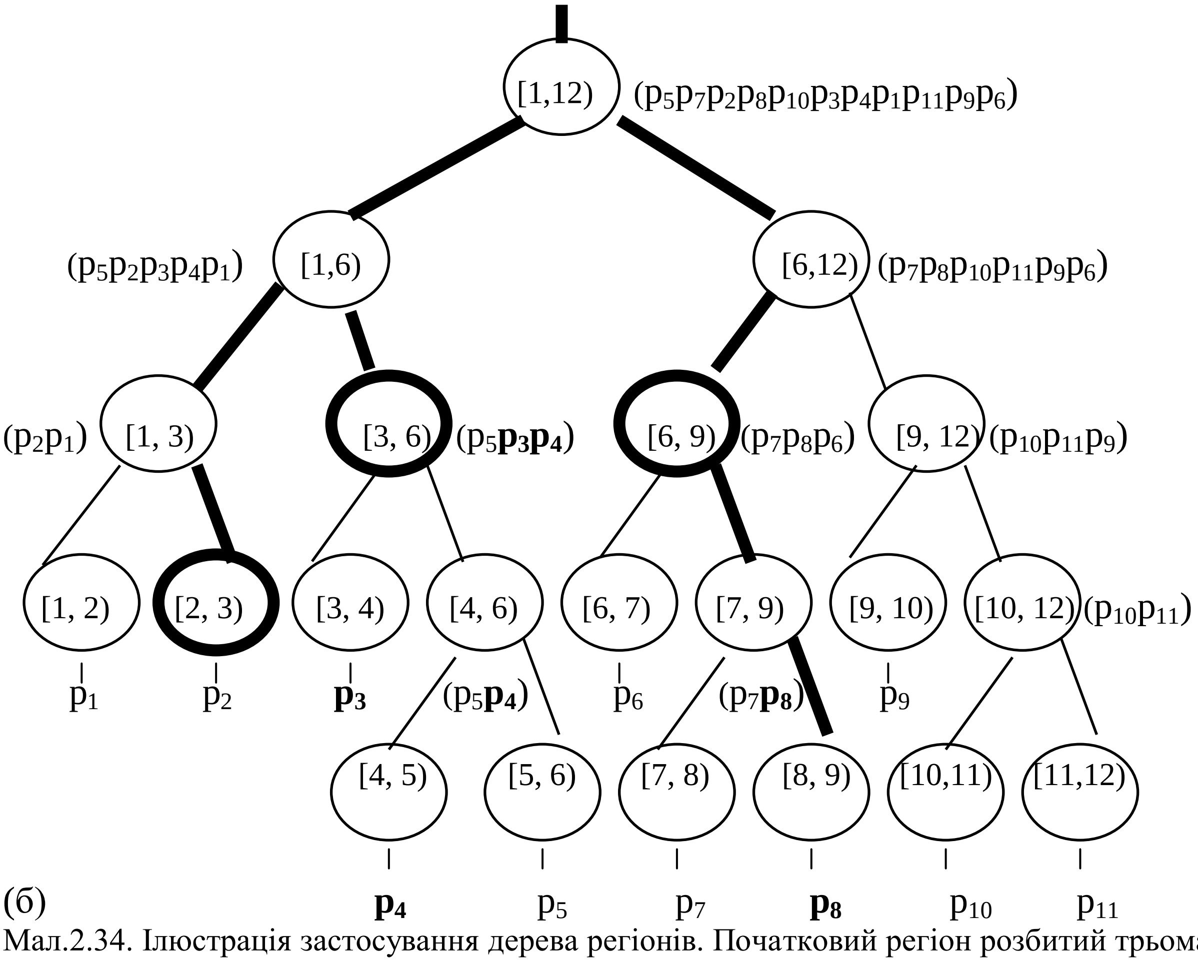


(б) Мал.2.33.(а),(б)

Теорема. *Регіональний пошук на d-вимірному файлі з N точок можна провести $(N \log^{d-1} N, \log^{d} N)$ - алгоритмом, якщо затратити $O(N \log^{d-1} N)$ часу на попередню обробку. Цей алгоритм заснований на методі дерева регіонів,і, зокрема, для d = 2 час запиту, пам'ять і час попередньої обробки такі: $O(\log^2 N + k)$, $O(N \log N)$ і $O(N \log N)$ відповідно.*



(а)



(б)

Мал.2.34. Ілюстрація застосування дерева регіонів. Початковий регіон розбитий трьома стандартними відрізками (а). Пошукові дії показані на мал. (б).

Теорема. *Регіональний пошук на d-вимірному файлі з N точок можна провести* $(N \log^{d-1} N, \log^d N)$ *- алгоритмом, якщо затратити* $O(N \log^{d-1} N)$ *часу на попередню обробку. Цей алгоритм заснований на методі дерева регіонів,і, зокрема, для* $d = 2$ *час запиту, пам’ять і час попередньої обробки такі:* $O(\log^2 N + k)$, $O(N \log N)$ *і* $O(N \log N)$ *відповідно.*