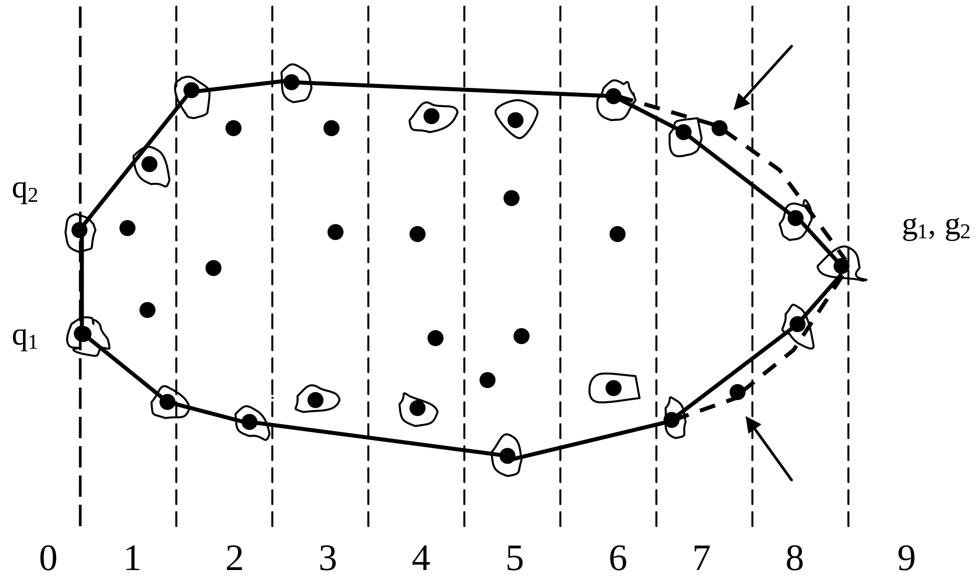
4.8.1.Алгоритм апроксимації опуклої оболонки



Мал.3.23. Апроксимація опуклої оболонки.

Основна ідея алгоритму. Полягає в тому, щоб виділити із заданої множини S точок деяку підмножину S^*, опукла оболонка якої являє собою апроксимацію опуклої оболонки заданої множини .

<u>*Побудова*.</u>1.Визначаються чотири екстремальні точки: $\boldsymbol{q}_1 = \min\limits_y \min\limits_x \boldsymbol{S}$, $\boldsymbol{q}_2 = \max\limits_y \min\limits_x \boldsymbol{S}$, $\boldsymbol{g}_1 = \min\limits_y \max\limits_x \boldsymbol{S}$, $\boldsymbol{g}_2 = \max\limits_y \max\limits_x \boldsymbol{S}$.Вибираємо точки які мають мінімальну та максимальну x-кординати:x_{max}, x_{min}. Вертикальна смуга між ними розбивається на k смуг рівної ширини. (Ці k смуг утворюють послідовність “комірок”, по яким будуть розподілятись N точок множини S.)

2.В кожній із цих k смуг визначаються точки з максимальними y-координатами: $\boldsymbol{S}_i^* = \{ \boldsymbol{P}^i_{\min y}, \boldsymbol{P}^i_{\max y} \}$ ($2k$ точок).

3. Множини точок з екстремальними координатами 1-ої та $к$- ої смуг такі (максимум 4 точки): $\boldsymbol{S}_1^* = \{ \boldsymbol{P}^1_{\min y}, \boldsymbol{P}^1_{\max y}, \boldsymbol{q}_1, \boldsymbol{q}_2 \}$, $\boldsymbol{S}_i^* = \{ \boldsymbol{P}^k_{\min y}, \boldsymbol{P}^k_{\max y}, \boldsymbol{g}_1, \boldsymbol{g}_2 \}$. Сформована множина містить не більш ніж $2k + 4$ точок і позначимо її через S^*.

4. Будується опукла оболонка множини S^*, яка є апроксимацією оболонки заданої множини S одним із відомих методів (наприклад, методом Грехема).

<u>*Реалізація методу.*</u>

1.Вказані k смуг відображаються в масиві із ($k+2$) елементів(0-й та ($k+1$) –й елементи містять дві точки із екстремальними значеннями x-координати: відповідно (x_{min}, x_{max})).

2.Смуга (номер смуги $\boldsymbol{i}$) у яку потрапляє деяка точка p, визначається співвідношенням: $i = \lfloor (x(p) - x_{min})/\Delta \rfloor$, де $\Delta = (x_{max} - x_{min})/k$.

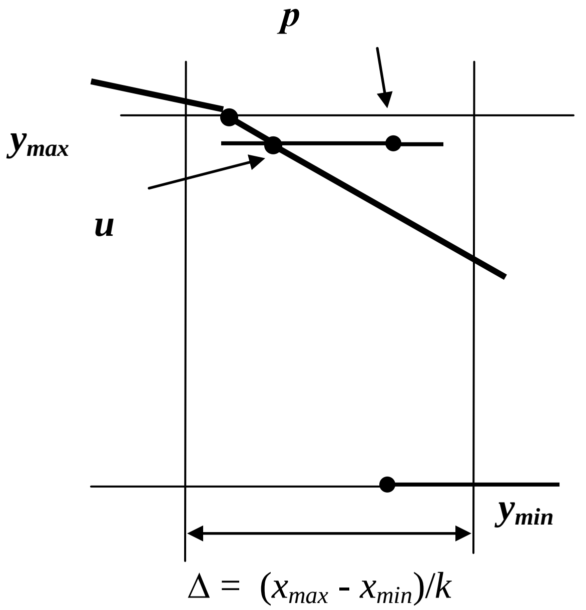
3. Мінімум та максимум у кожній смузі можна визначати паралельно.

4. Для побудови упорядкованої множини точок порівнюємо у кожній смузі значення x - координати двох точок множини S^*, які належать цій смузі. Повний час роботи алгоритму рівний $\theta(N + k)$.

Питання: Як далеко від наближеної опуклої оболонки може бути точка із S , яка розташована за її межами ? Відповідь на це дає наступна теорема.

Теорема. *Довільна точка $p \in S$, яка не потрапила в середину наближеної опуклої оболонки, розташована на відстані, не більшій ніж $\Delta=(x_{max} - x_{min}) / k$.*

Доведення. Розглянемо смугу. яка містить точку p. Так, як точка p розташована за межами наближеної оболонки, то вона не може мати ні найбільшу ні найменшу y-координату серед точок, які потрапили у цю ж смугу.$\Rightarrow$ $y_{min} \le y(p) \le y_{max}$. Якщо u це точка перетину горизонтальної прямої, яка проходить через точку p, з ребром наближеної оболонки, то довжина відрізка pu обмежує зверху відстань від точки p до оболонки. Довжина відрізку pu сама обмежена зверху шириною смуги ($\Delta=(x_{max} - x_{min}) / k$).

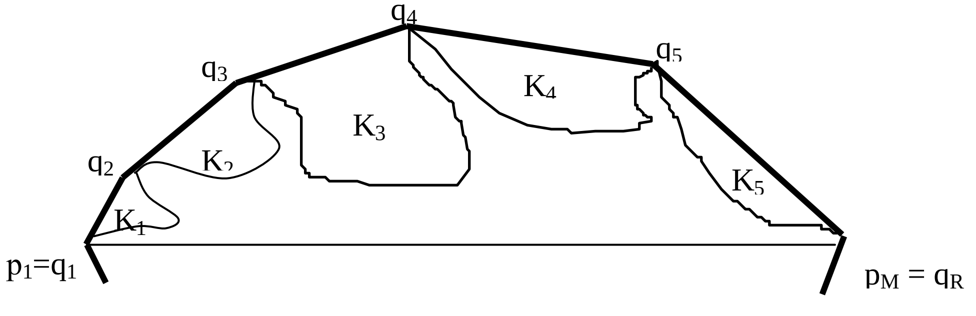


Мал. 3.24. Аналіз алгоритму апроксимації.

Побудова опуклої оболонки простого многокутника

Шукатимемо алгоритми, оцінка яких у гіршому випадку, менша *$O(N\log N)$*.

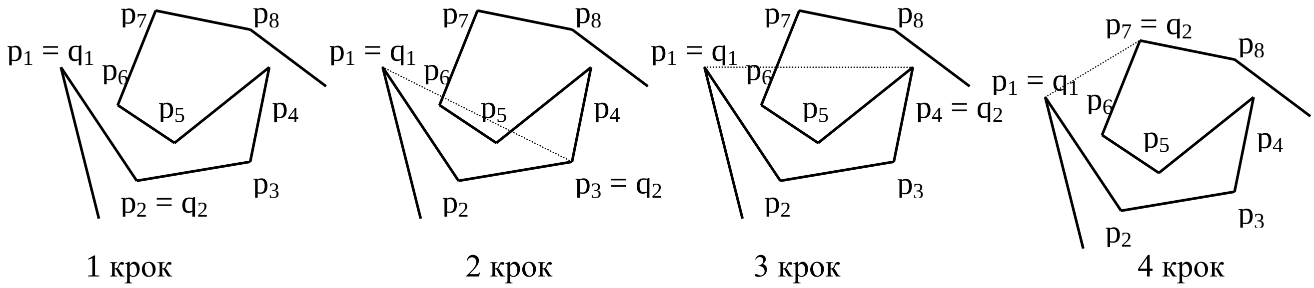
Алгоритм Лі. Нехай p_1 – сама ліва вершина заданого простого многокутника P, а (p_1, p_2, ..., p_N) – впорядкована циклічна послідовність його вершин (за вершиною p_N йде p_1). Нехай внутрішня частина P залишається праворучу при обході його границі в указаному порядку (множина вершин многокутника орієнтована проти годинникової стрілки). Нехай p_M – сама права вершина. p_1 та p_M граничні точки опуклої оболонки многокутника P. Вони розбивають послідовність вершин многокутника на два ланцюги: один від p_1 до p_M, другий – від p_M до p_1. Достатньо дослідити побудову опуклої оболонки для ланцюга (p_1, p_2, ..., p_M), яку будемо називати *верхньою оболонкою.*



Мал.3.25. Верхня опукла оболонка простого многокутника.

Нехай (q_1, q_2, ..., q_R) – підпослідовність (p_1, p_2, ..., p_M), в якій $q_1 = p_1$ та $q_R = p_M$ – шукана опукла оболонка многокутника. Кожне ребро q_iq_{i+1} є "кришкою" "кармана" K_i, де карман K_i – це ланцюг послідовності (p_1, p_2, ..., p_M), першою та останньою вершинами якої є q_i та q_{i+1} відповідно.

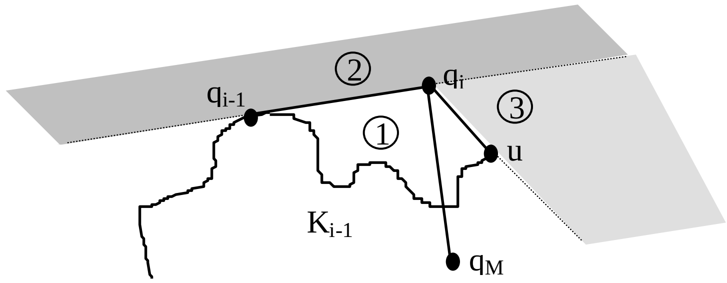
Алгоритм проходить ланцюг та послідовно будує кришки усіх карманів. *Критичною* будемо називати вершину, яка з останньою знайденою вершиною типу q утворює карман. *Кроком просунення* будемо називати перехід від однієї критичної вершини до іншої (мал.3.26). Наприклад, на третьому кроці критичною точкою є p_4. Наступною критичною точкою буде p_7. При цьому критична точка p_4 не належить опуклій оболонці.



Мал.3.26. Кроки просунення алгоритму.

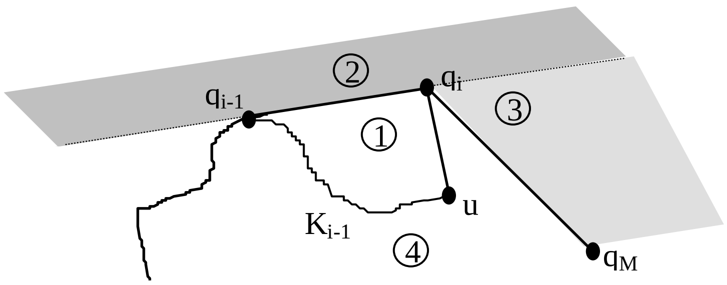
Припустимо, що границя многокутника переглядається від вершини p_1 до p_s ($s \leq M$) і вершина $p_s = q_i$ є критичною. Позначимо через *u* вершину границі *P*, яка передує q_i. В залежності від положення *u* відносно орієнтованого відрізка q_Mq_i мають місце два випадки:

1.*Вершина **u** знаходиться праворуч q_Mq_i або на ньому.* У цьому випадку у вертикальній смузі, визначеній вершинами q_M і q_i, досліджуються три області (1,2,3), які визначаються: прямою, що проходить через точки q_{i-1} та q_i; променем, який є продовженням відрізку q_iu та частиною границі многокутника P, яка відповідає карману K_{i-1}.



Мал. 3.27 (1)

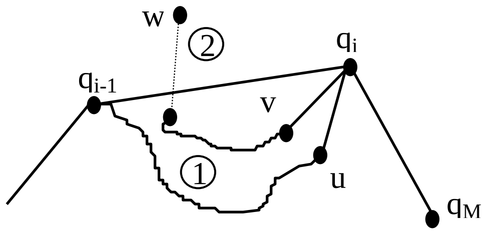
2. *Вершина **u** знаходиться ліворуч від q_Mq_i.* У цьому випадку додається до розгляду четверта область (4).



Мал. 3.27 (2)

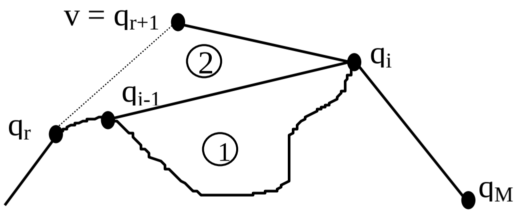
Позначимо через *v* вершину, яка слідує за q_i на границі многокутника *P*. Ця вершина *v* може бути в одній із областей. В областях 2 та 3 вершина *v* буде критичною, в інших – ні. Розглянемо випадки розташування вершини *v* в кожній із цих областей **(**мал. 3.28).

Область 1. Границя многокутника заходить у карман **(**Мал. 3.28 (а)). Рухаємось по границі до тих пір, поки не досягнемо першого ребра границі, одна з вершин w якого знаходиться зовні кармана (в області 2). Так, як *P,* карман та його кришка утворюють прості многокутники, то згідно теореми про Жорданову криву, границя многокутника *P* обов’язково перетинає кришку кармана. Переходимо до обробки w, а значить до наступного випадку.



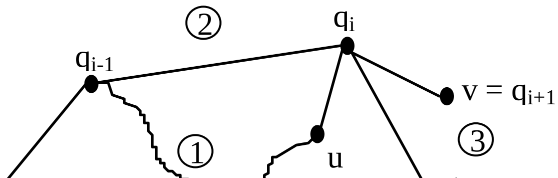
Мал. 3.28 (а)

Область 2. *v* в області 2 є критичною **(**Мал. 3.28 (б)). Шукається опорна пряма з вершини *v* до ланцюга ($q_1, q_2, ..., q_{i-1}$). Якщо ця пряма містить q_r ($r < i$), то вершини ($q_{r+1}, q_{r+2}, ..., q_i$) вилучаються, а *v* береться як нова q_{r+1}. *v* - вершина опуклої оболонки, так як вона зовнішня відносно оболонки ($q_1, q_2, ..., q_M$).



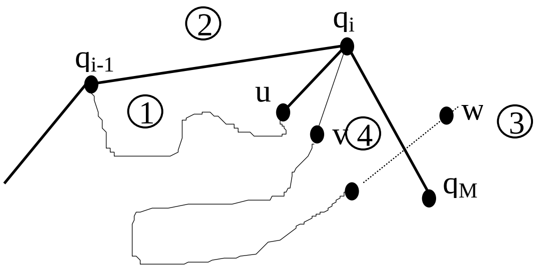
Мал. 3.28 (б)

Область 3. Вершина *v* є критичною і вибирається як q_{i+1} **(**Мал. 3.28 (в)).



Мал. 3.28 (в)

Область 4. У цьому випадку границя многокутника заходить всередину опуклої оболонки. Як і в першому випадку рухаємось по границі многокутника до тих пір, доки не досягнемо першого ребра, яке має властивості: одна з його вершин є зовнішньою до області 4 або співпадає з q_M. В останньому випадку процедура завершується. У першому ж випадку вершина *v* міститься в області 3 або 2 і обробляється відповідно. **(** Мал. 3.28 (д)).



Мал. 3.28 (г)

<u>*Алгоритм*</u>

procedure ОПУКЛА_ОБОЛОНКА_ПРОСТОГО_МНОГОКУТНИКА
(p_1, ..., p_M)
begin P $\Leftarrow$ (p_2, ..., p_M);
Q $\Leftarrow$ q_0;
Q $\Leftarrow$ p_1
while (P $\neq \varnothing$) **do**
begin v $\Leftarrow$ P;
if (($q_{i-1}q_i$v) - правий поворот)
(* області ς, ˉ *) **then**
if (uq_1v) - правий поворот (*області̧ , ˉ *) **then**
if (q_Mq_iv) - правий поворот
(* область̧ *) **then** Q $\Leftarrow$ v
else (* область ˉ *)
while (ПЕРЕДНІЙ(P) знаходиться
зліва від q_Mq_i або на ньому) **do**
ВИШТОВХНУТИ P
else (* область ς *)
while (ПЕРЕДНІЙ(P) знаходиться зліва
від q_iq_{i-1} або на ньому) **do**
ВИШТОВХНУТИ P
else (* область ƶ *)
begin while (($q_{i-1}q_i$v) - лівий поворот)
do ВИШТОВХНУТИ Q;
Q $\Leftarrow$ v
end
end
end.

Аналіз складності алгоритму ОПУКЛА_ОБОЛОНКА_ПРОСТОГО_ МНОГОКУТНИКА простий. Після ініціалізації кожна вершина відвідується рівно один раз, перш ніж вона буде прийнята, або виштовхнута. Обробка кожної вершини многокутника здійснюється за постійний час. При побудові опорної прямої в циклі *while* на кожну операцію вилучення витрачається постійний час. Враховуючи, що послідовністі (p_1, ..., p_M) і (q_1, ..., q_R) містять O(M) елементів, то маємо наступну теорему:

Теорема. *Опукла оболонка простого многокутника з N вершинами може бути побудована за оптимальний час $\theta(N)$ при використанні пам'яті об'ємом $\theta(N)$.*