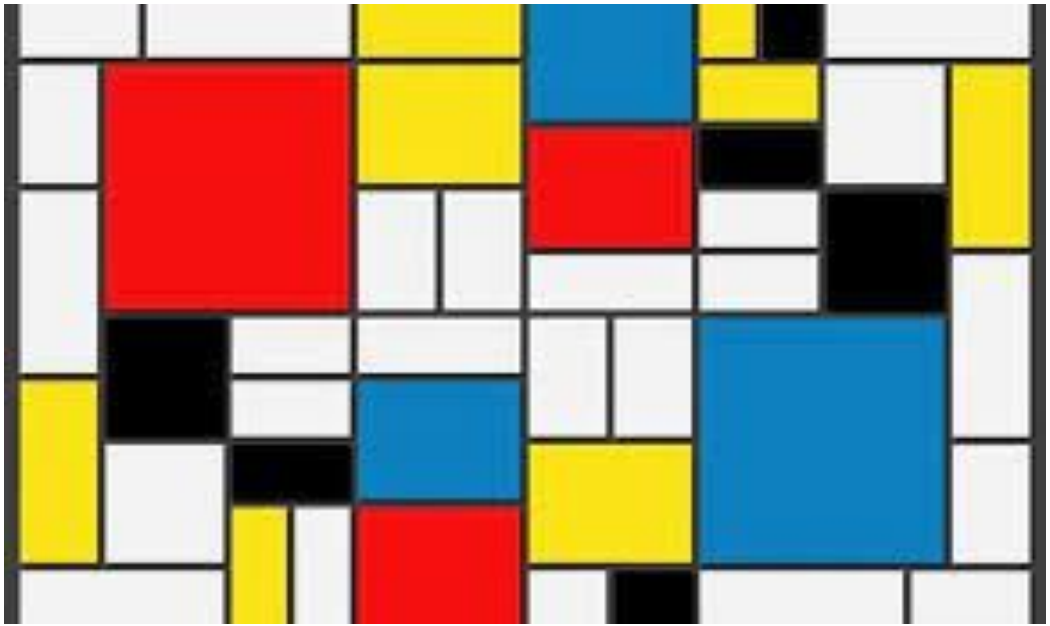
An Optimal Time and Minimal Space Algorithm for Rectangle Intersection Problem

D. T. Lee



組員: 林宏信 游家權 龔柏丞

報告大綱

- 問題描述
- 解法說明
- 解題步驟與範例
- 結論與時間複雜度

問題描述

- 問題:在二維平面上，給定N個長方形，找出所有長方形的交點。
- 假設:
  - 所有長方形的頂點不會共線。
  - 所有長方形的邊長都是水平或是垂直線。
- 時間複雜度:期望在O(nlog(n) + s)時間內解決，s是交點數量
- 空間複雜度 : O(n)
- 符號定義:

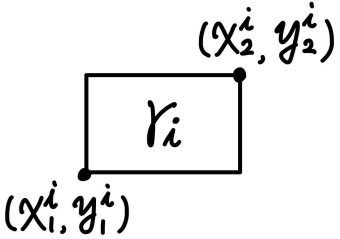
r_i:代表i-th長方形

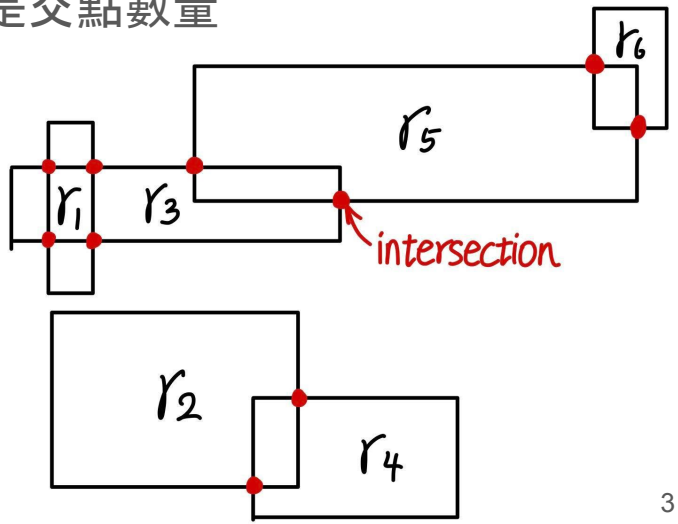
x_1^i:代表i-th長方形的左界

x_2^i:代表i-th長方形的右界

y_1^i:代表i-th長方形的下界

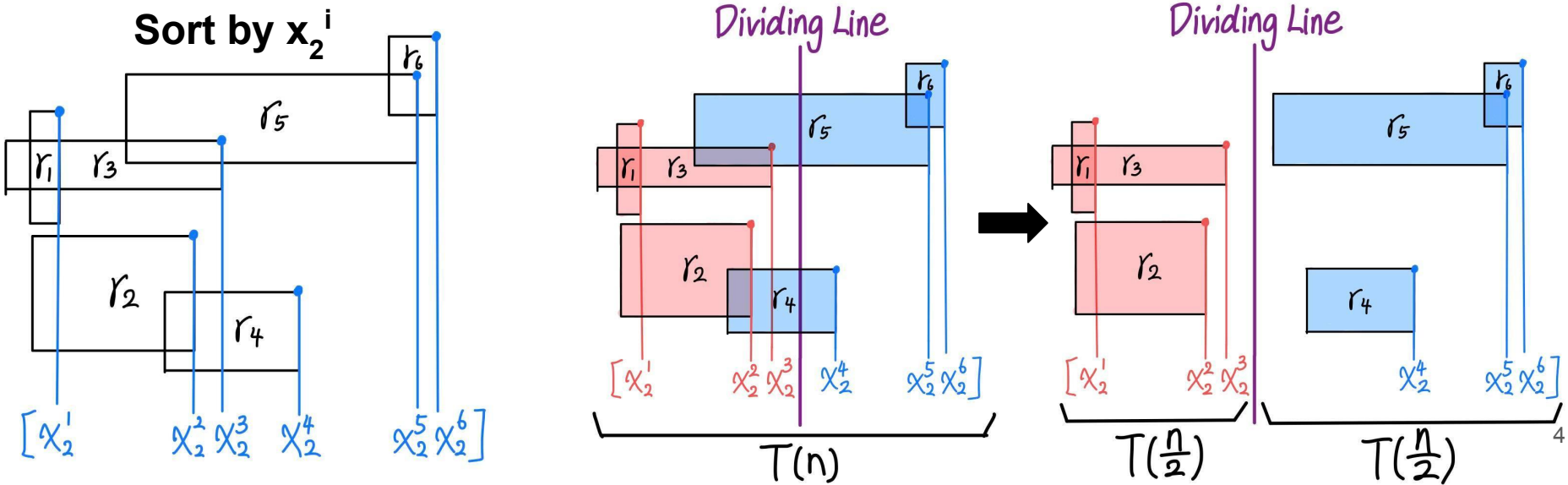
y_2^i:代表i-th長方形的上界





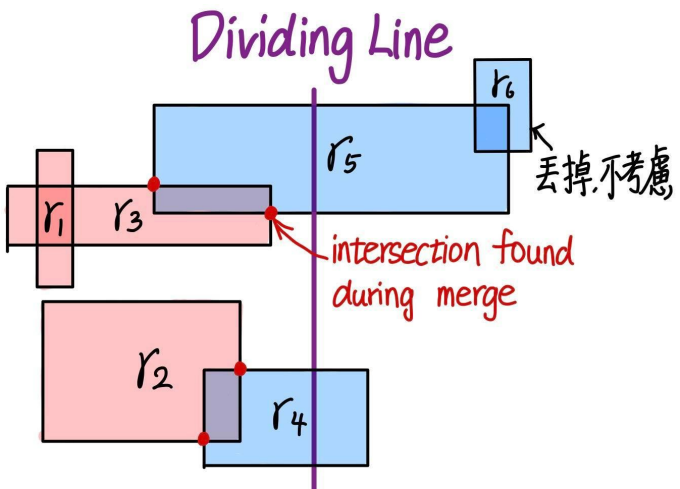
解法說明: Divide and Conquer

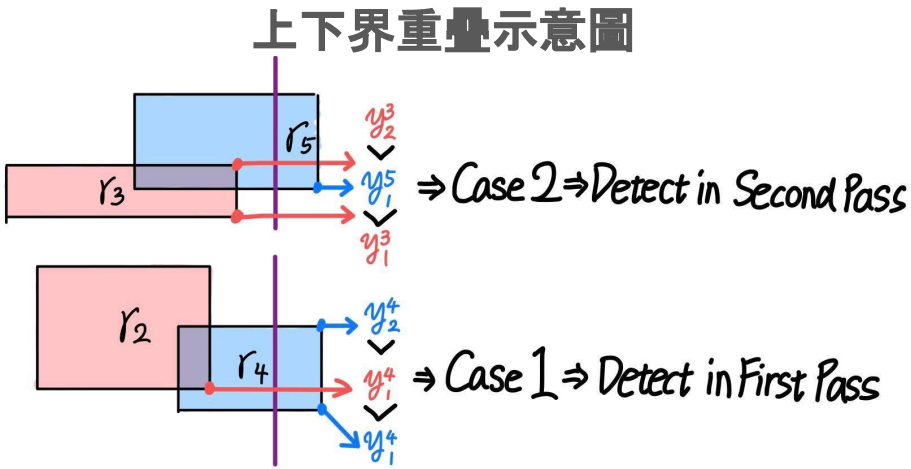
- Sorting: 依照各長方形的右界座標(x_2^i)與下界座標(y_1^i)進行排序。
- 找右界座標的中位數作為分界線，將大問題拆成兩個相同大小的子問題，重複此步驟，直到子問題只剩下一個或是兩個矩形時，直接解出答案。



解法說明: Merge

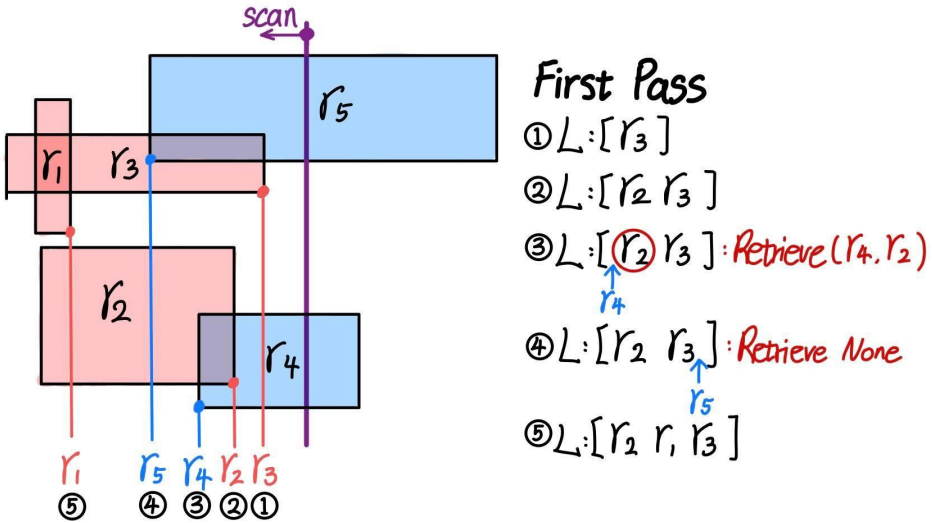
- Merge時，要考慮右邊子問題的長方形(藍色)中，是否與有左邊子問題的長方形(紅色)相交。有交點的話需要找出來並回傳。
- 符號定義：$S_1 = \{r_1, r_2, r_3\}, S_2 = \{r_4, r_5\}$
- 長方形相交必須同時滿足左右界重疊與上下界重疊，其中上下界重疊需要用兩次Plan Sweep檢查才能找出來





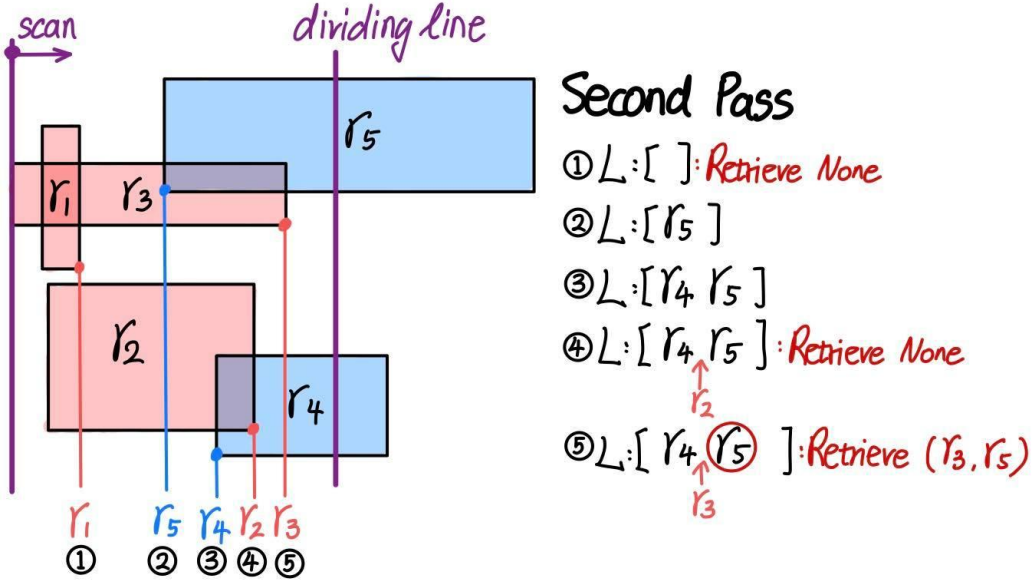
解法說明: Merge

- 建立一個ordered-list L來儲存目前為止遇到的右界，其中L內的所有長方形必須按照下界值排序。
- **First Pass**: 從分界線開始從右往左掃描。當遇到S_1的右界時，將該長方形(r_i)存到L裡。當遇到S_2的左界時，則去L裡檢查符合條件($y_1^i < y < y_2^i$)的長方形



解法說明: Merge

- **Second Pass**: 從最左邊的矩形開始，從左往右掃描。當遇到S_1的右界時，則去L裡檢查符合條件($y_1^i < y < y_2^i$)的長方形。遇到S_2的左界時，則將該長方形的下界值存到L裡。



但如何實作 ordered-list L?

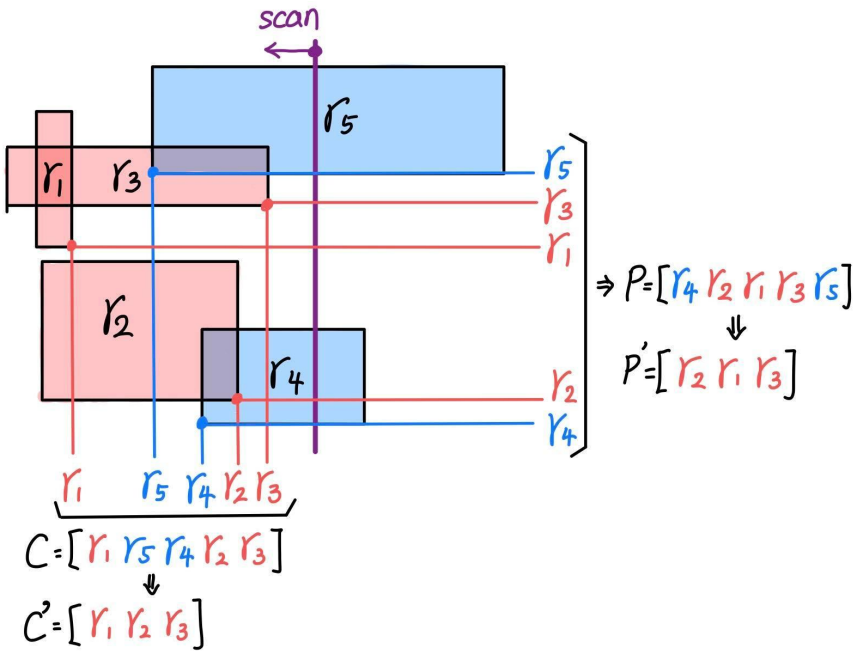
- 使用Binary Search Tree是能夠實現L，但是每次插入跟找SUCC都將花費O(log(n))的時間，會導致光是Merge Step就花費O(nlog(n))，整個演算法的時間複雜度會成為$O(n\log^2(n) + s)$
- 本篇論文想要達到O(nlog(n) + s)的時間複雜度，所以需要想辦法讓Merge Step在O(n)時間內完成。
- 解法：用Link-List來實現L，但需要事先建立一個額外表格 LINK，使得在掃描 之前，我們就能知道各個長方形應該要插在L的哪個位置。

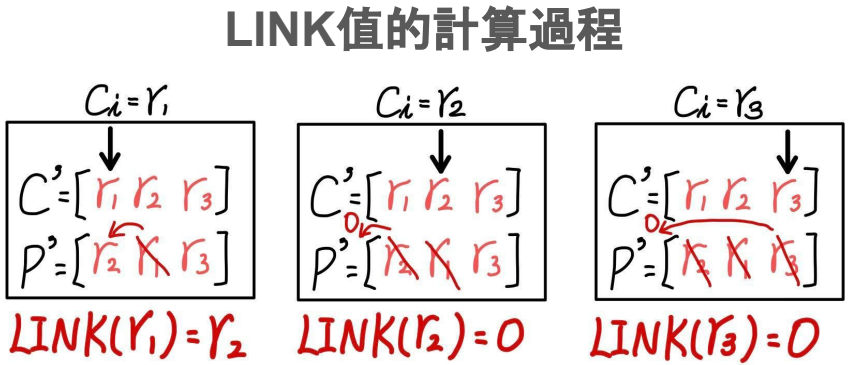
LINK 表格

	r_1	r_2	r_3	r_4	r_5
$\text{LINK}(r_i)$					

建立 LINK 表格

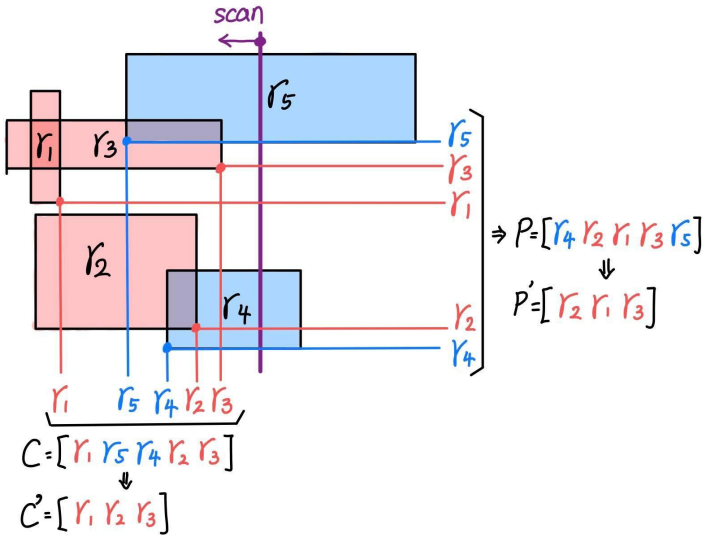
- 先計算S_1之中的LINK值。用pre-sorting結果來得到C', P'兩個array幫助運算
- r_i的LINK值就是P'裡r_i的前一個元素，找到LINK值後要把P'裡的r_i刪除



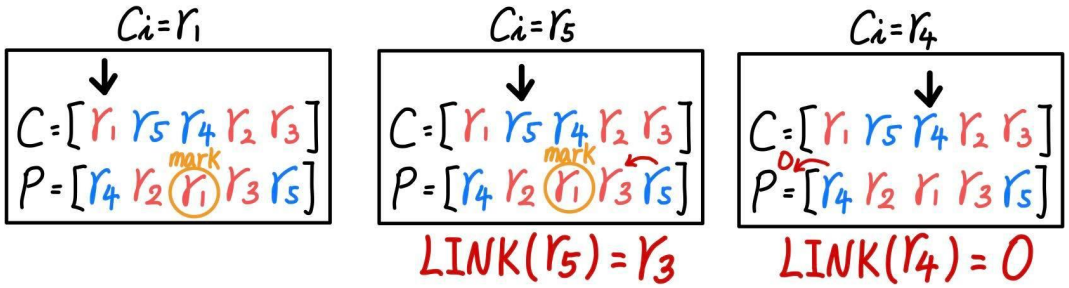


建立 LINK 表格

- 先計算S_2之中的LINK值，用pre-sorting結果來得到C, P兩個array幫助運算
- 遍歷C的元素時，如果遇到$r_i \in S_1$不計算LINK，標記該元素。
- 如果遇到$r_i \in S_2$，r_i的LINK值要從P裡的r_i往前找前一個元素，但如果遇到$r_i \in S_2$就跳過，或是遇到被標記過的也跳過。



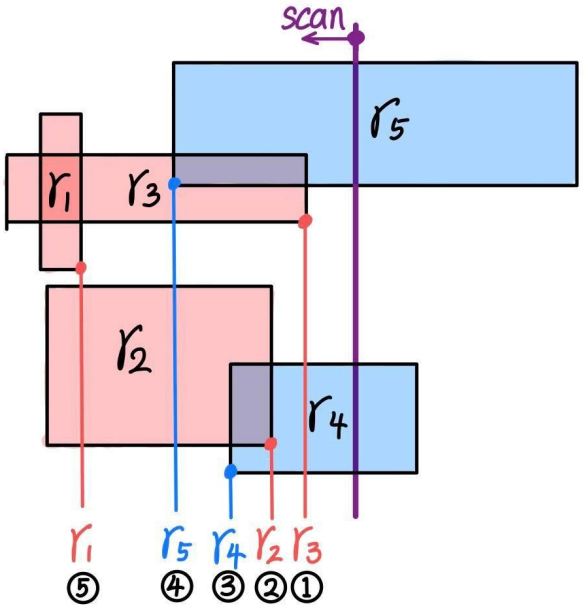
LINK值的計算過程



Details: 1.被標記的元素被訪問時，要執行 Path Compression
2. $r_i \in S_2$會指向前一個屬於S_1的元素

驗算LINK 表格

- LINK 表格標示了每個長方形應該要插入在L裡的正確位置。
- LINK 表格的計算皆是O(n)，使得merge step能在O(n)做完。



First Pass

①L:[r3]

②L:[r2 r3]

③L:[r2 r3]: Retrieve(r4, r2)

④L:[r2 r3]: Retrieve None

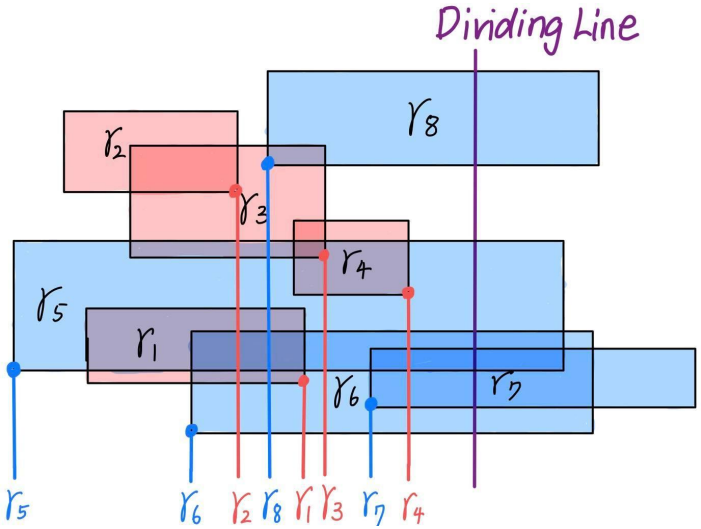
⑤L:[r2 r1 r3]

LINK 表格

	r_1	r_2	r_3	r_4	r_5
LINK(r_i)	r_2	0	0	0	r_3

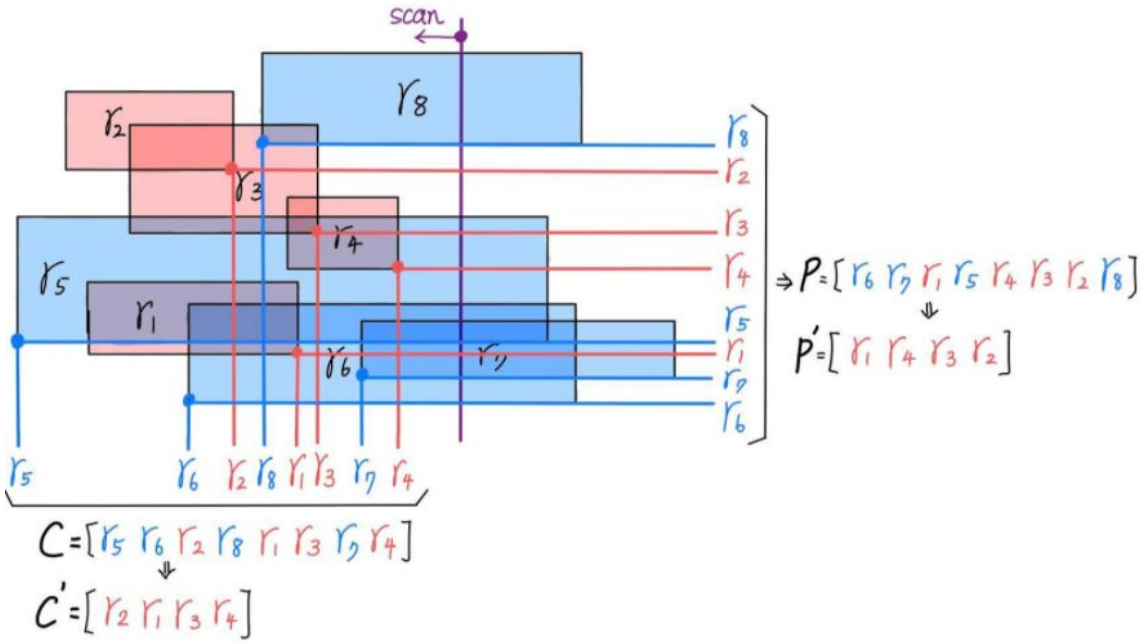
解題步驟

- **Step 1:** Pre-sort 依照各個長方形的右界座標與下界座標進行排序。
- **Step 2:** 找右界座標的中位數，將大問題拆成兩個相同大小的子問題，重複此步驟，直到子問題只剩下一個或是兩個矩形時，直接解出答案。
- **Step 3:** Merge 左邊子問題與右邊子問題



分治 (左)

- 在進入D&C前會進行排序，在那之後都可以在O(n)得到序數列C、P



解法步驟-LINK for S1

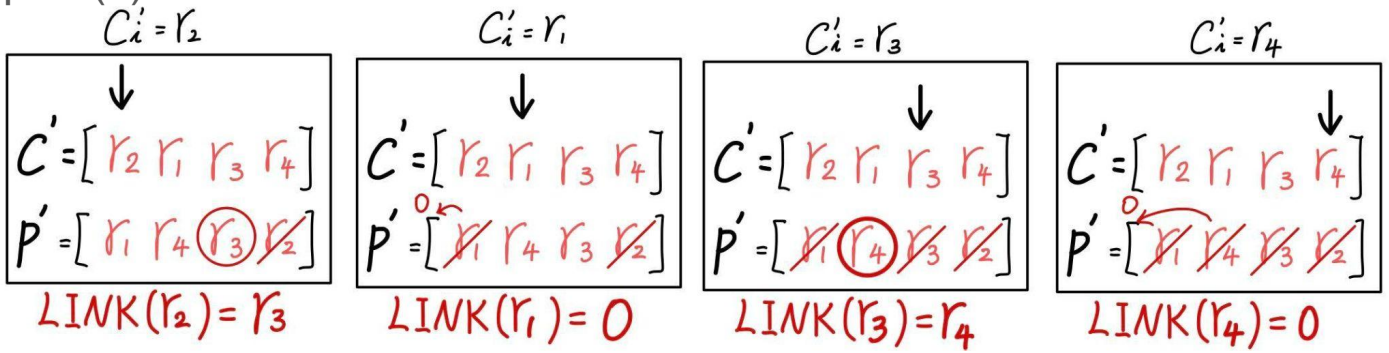
紀錄已經被掃描的r (紅色, 右往左) 的資料結構

從C'最後面開始建構, 因從右往左掃描

最左邊的 r 一定不會再被Link, 故可以刪除

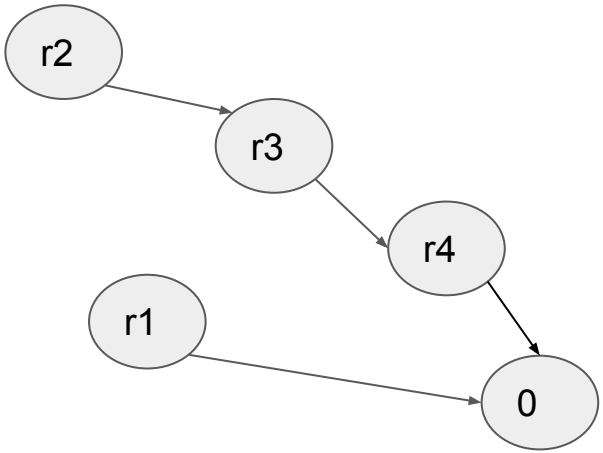
Link(ri) = prev(ri)

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0				



Link for S1

succ(0) = r1, succ(r1) = r4, succ(r4) = r3, succ(r3) = r2, 遞增



解法步驟-LINK for S2

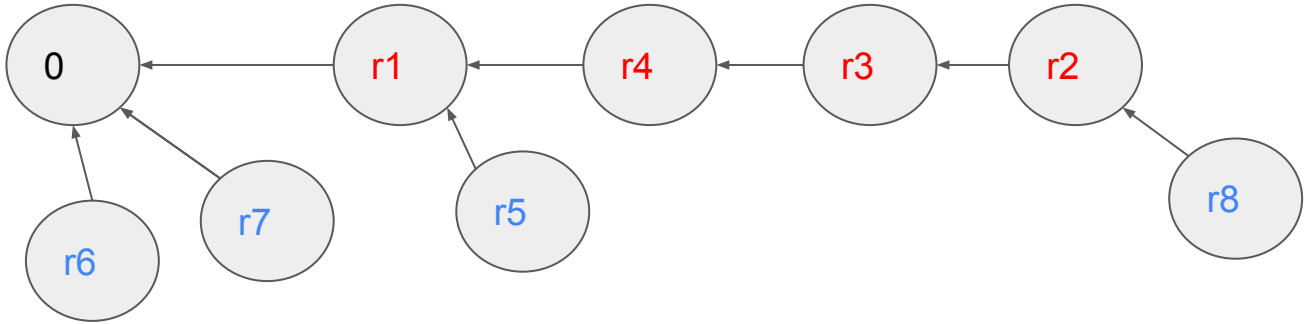
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0				

- 看到藍色的左邊界要找比目前藍色的y1高的紅色y1
- 可以從排序過後的y1得知
- 但排序的P沒有紀錄跟目前藍色在x有交集的紅色

解法步驟-LINK for S2

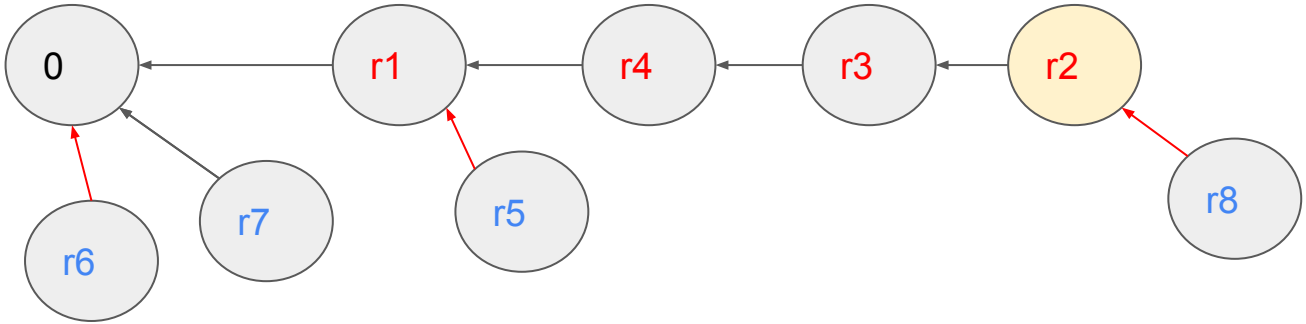
P’: sorted list of bottom boundary after grouping
P = {6 7 1 5 4 3 2 8} (sorted y1)

prev(r5) = r1



解法步驟-LINK for S2

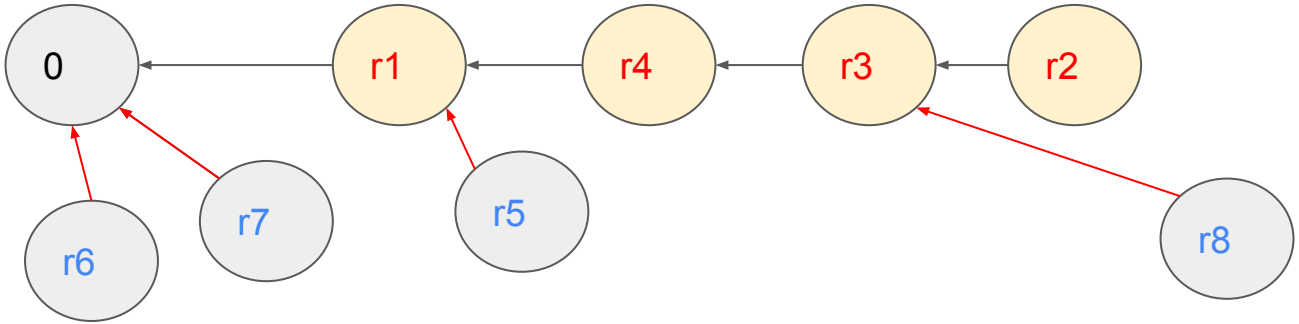
C = {5 6 2 8 1 3 7 4}



解法步驟-LINK

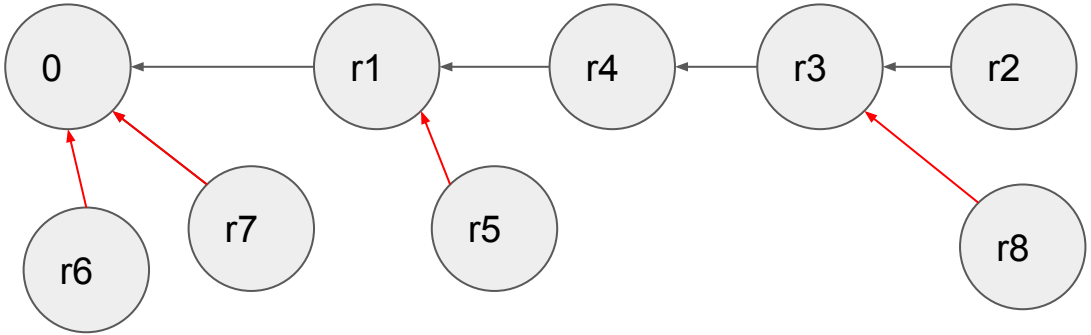
C = {5 6 2 8 1 3 7 4}

因為掃描的r8做邊界的時候r2的右邊界還沒被掃瞄到，所以把link指向離他更小一格且在遇到r8前會被掃瞄到的r3。



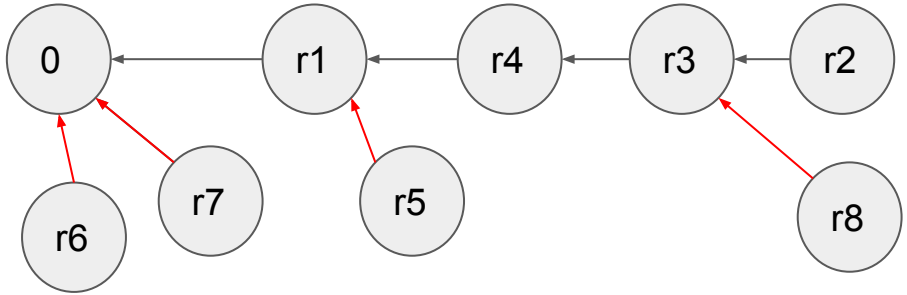
Link for S2

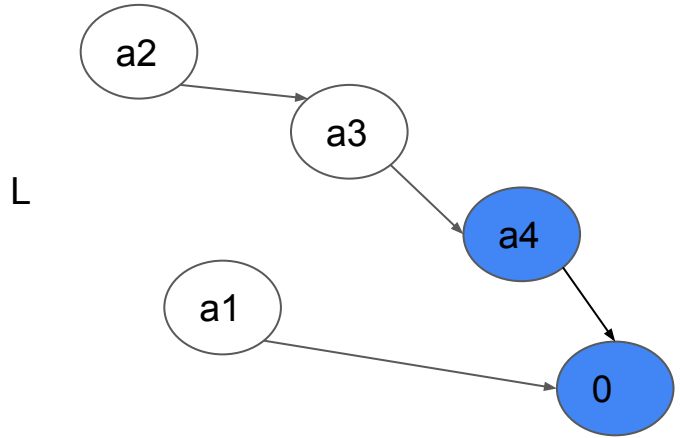
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0	r_1	0	0	r_3

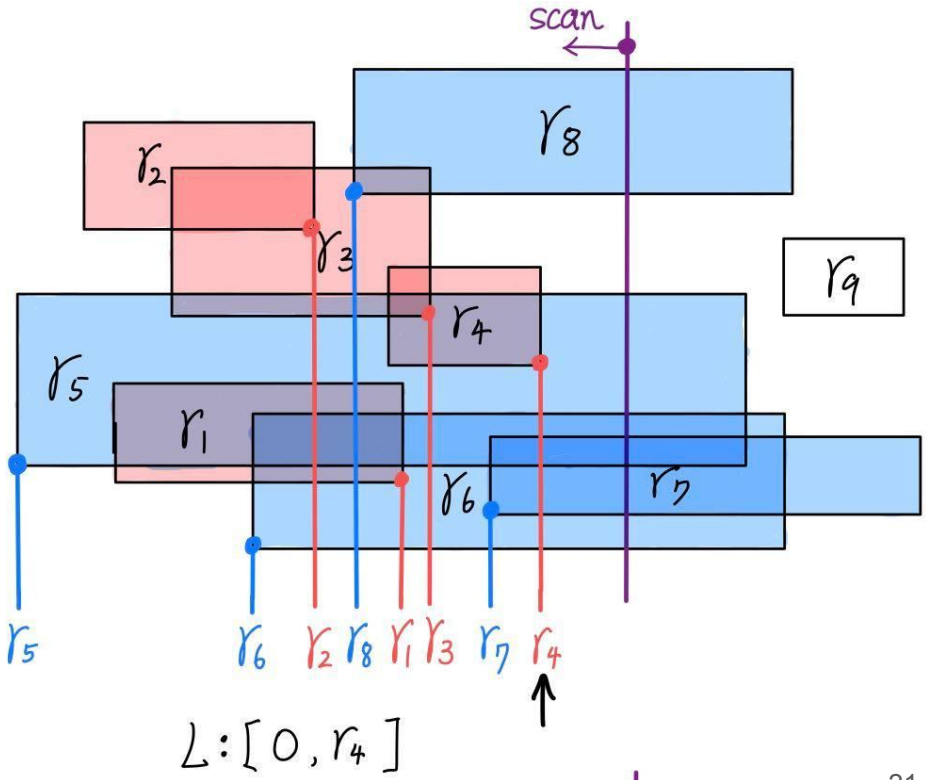


解法步驟

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0	r_1	0	0	r_3

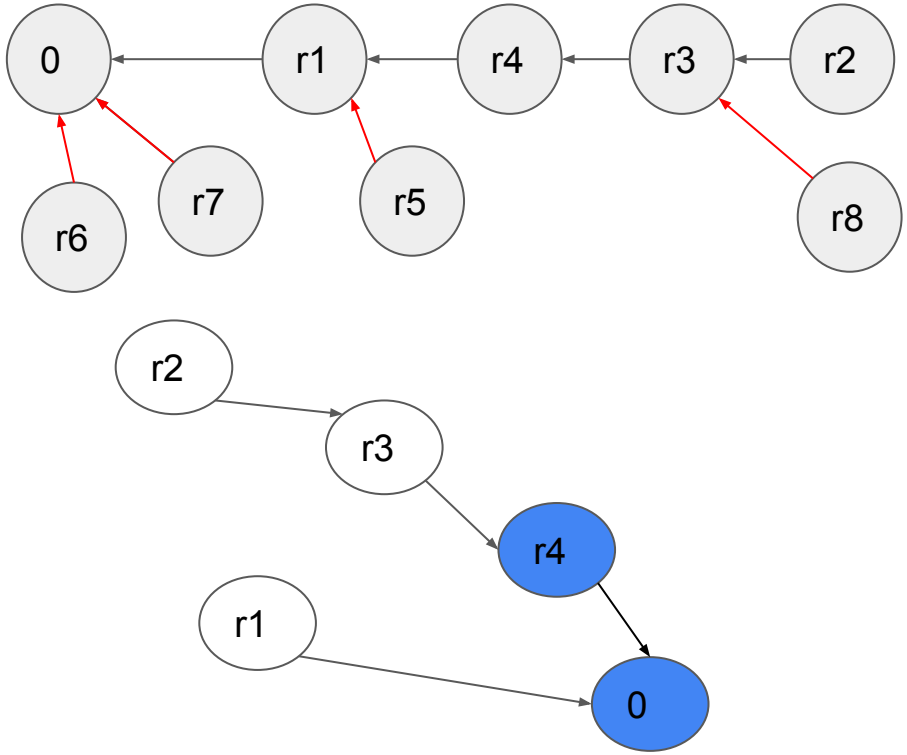


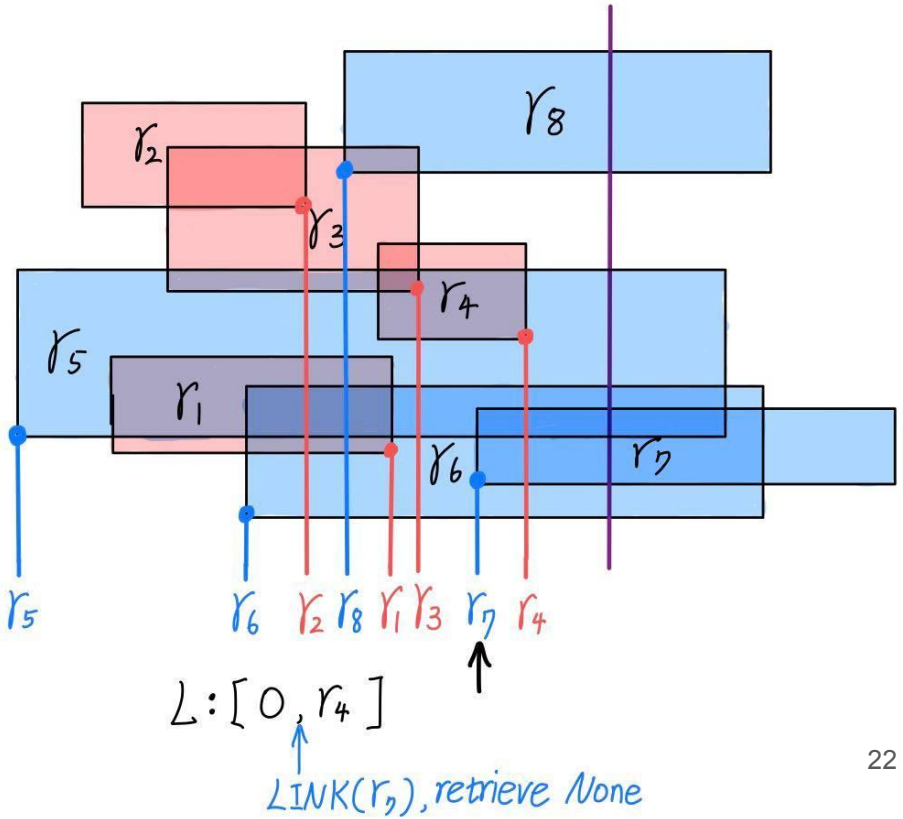




解法步驟

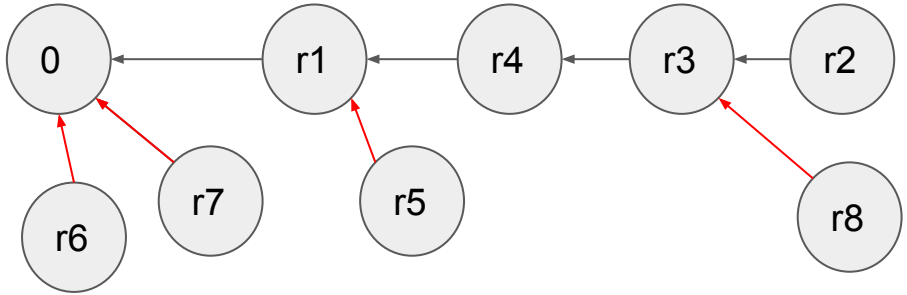
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0	r_1	0	0	r_3

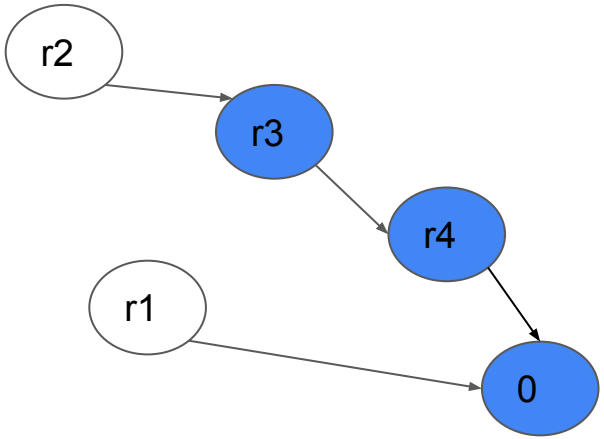


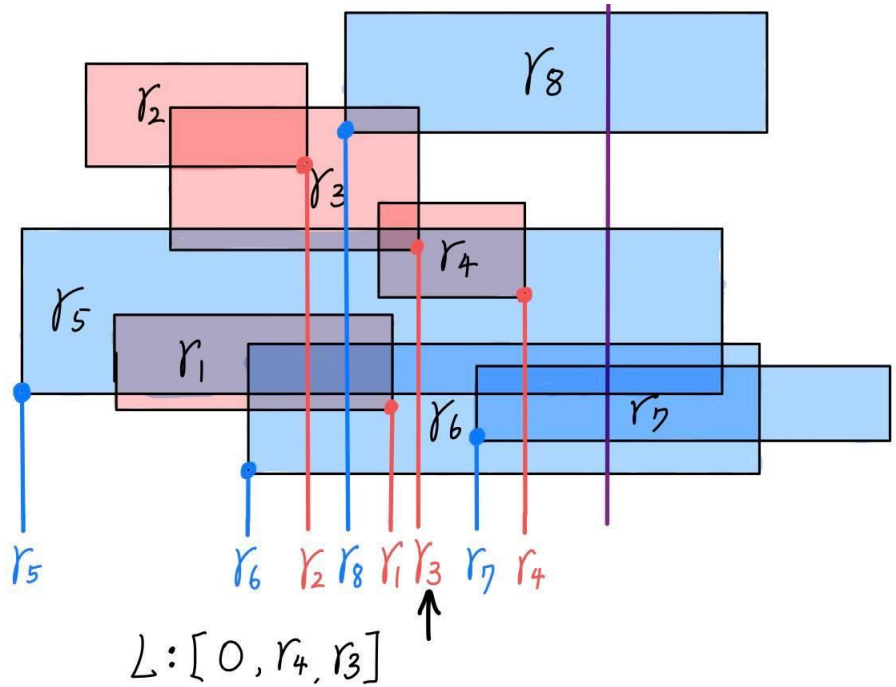


解法步驟

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0	r_1	0	0	r_3

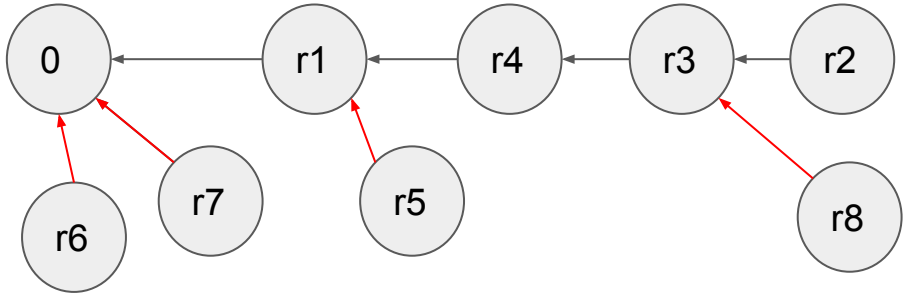


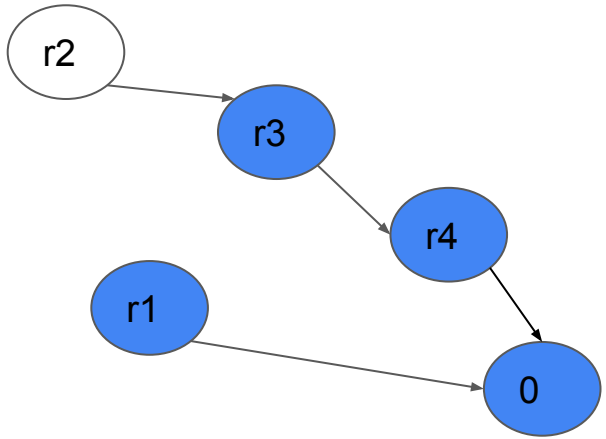


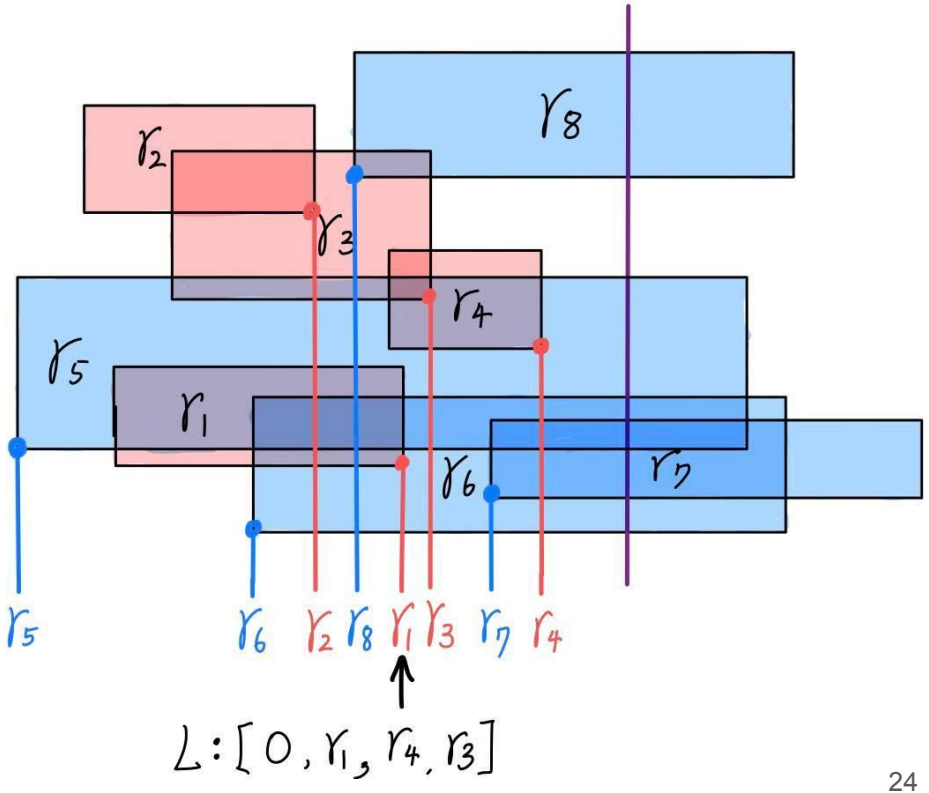


解法步驟

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0	r_1	0	0	r_3

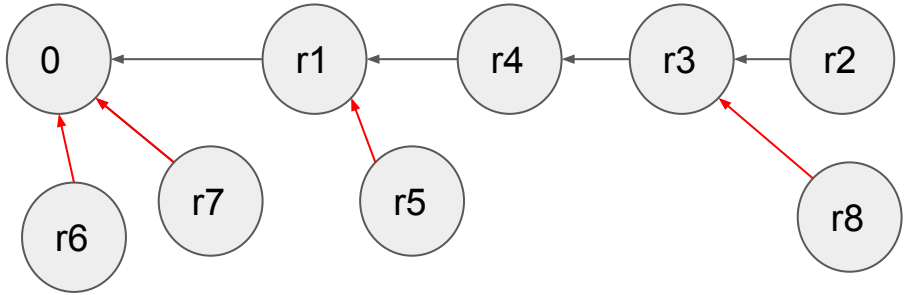


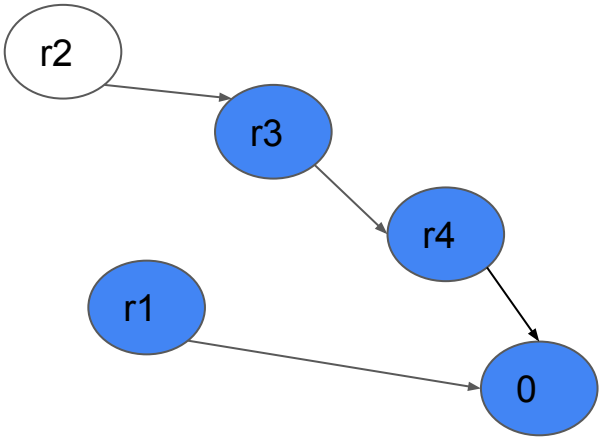


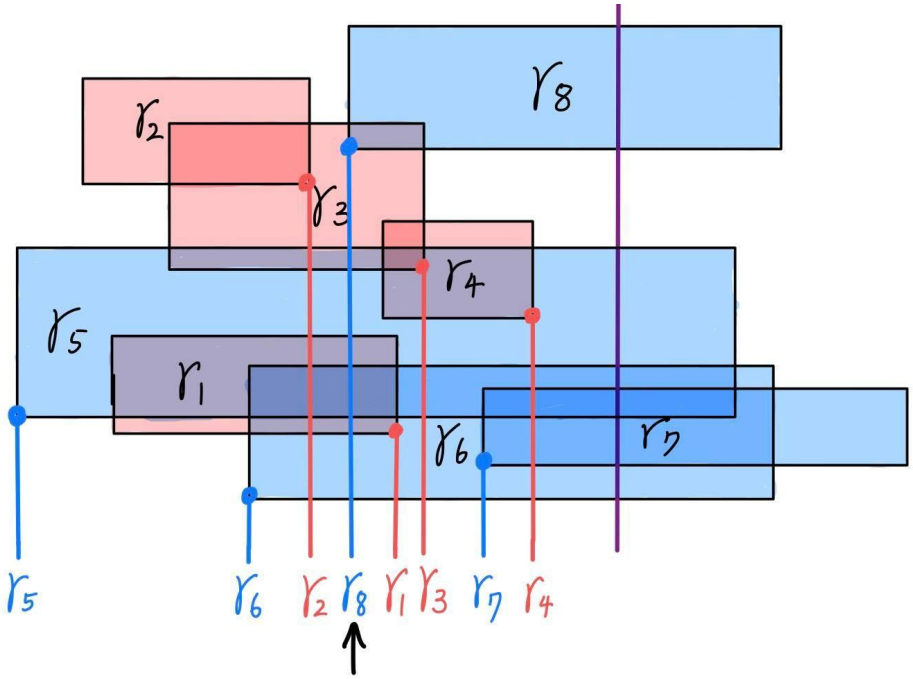


解法步驟

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0	r_1	0	0	r_3



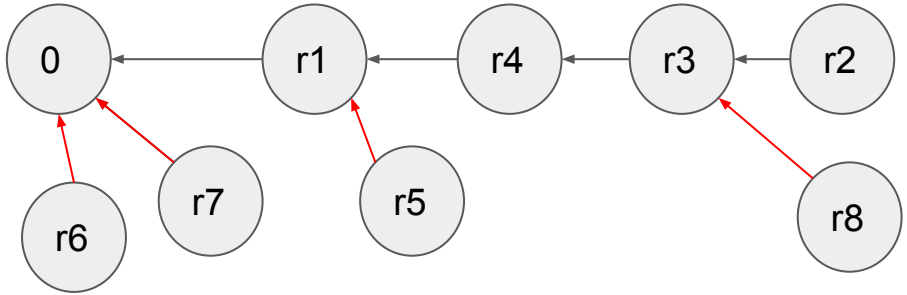


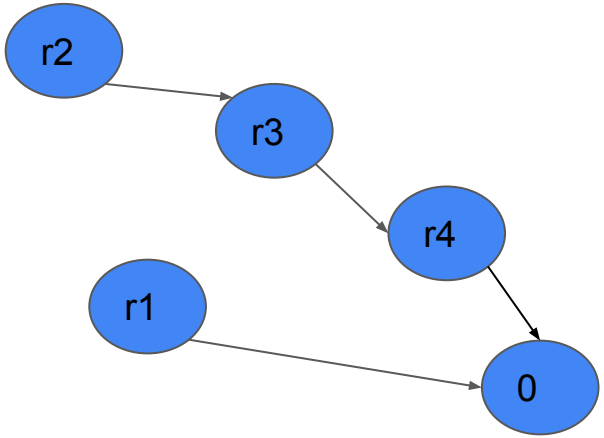


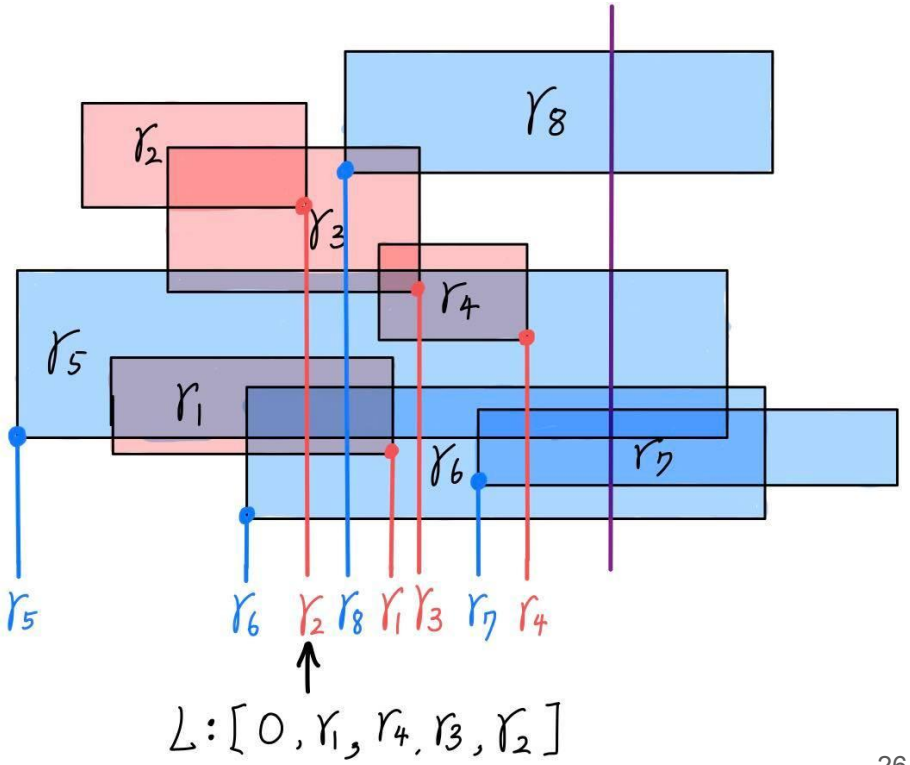
link(r8) = r3,
succ(a) = ""

解法步驟

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0	r_1	0	0	r_3

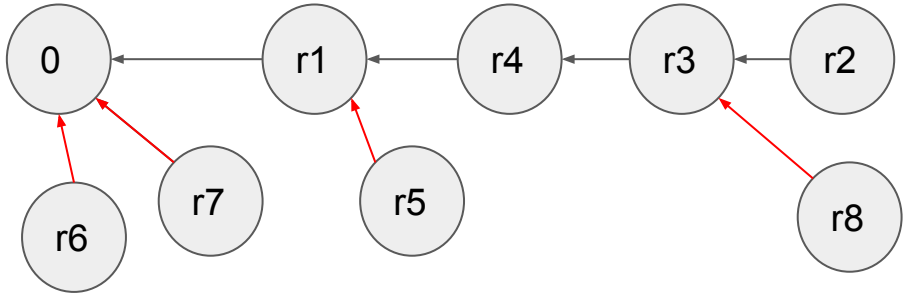


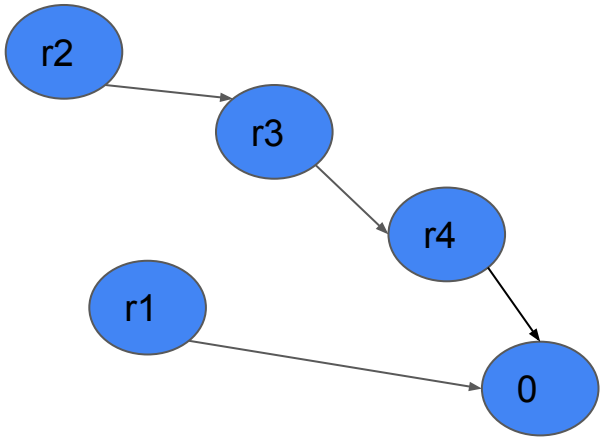


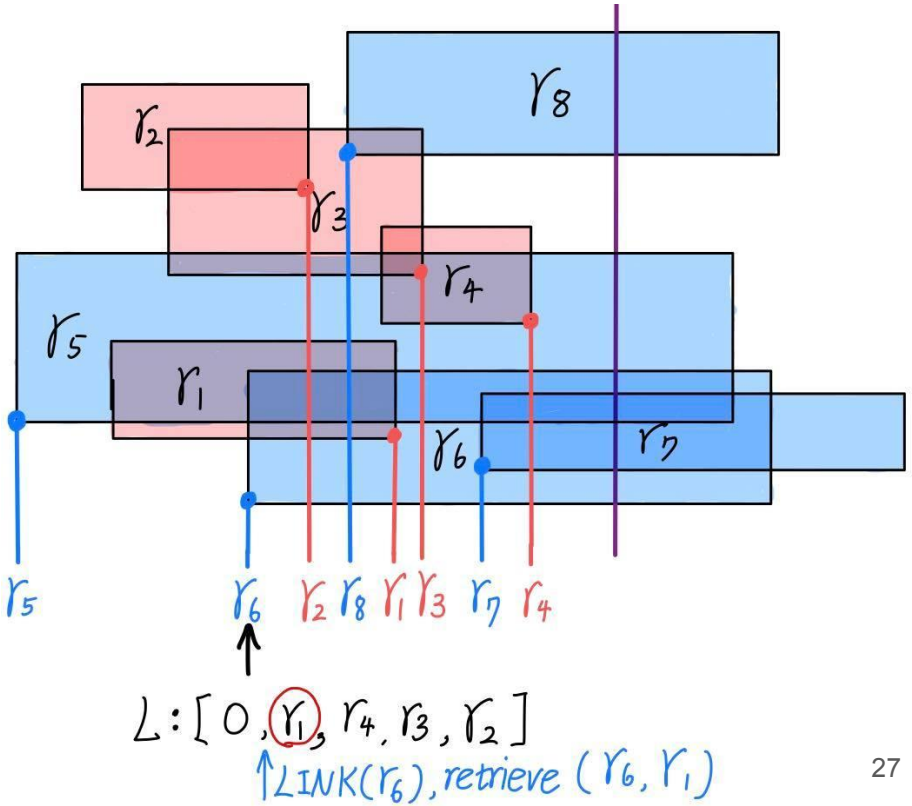


解法步驟

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0	r_1	0	0	r_3

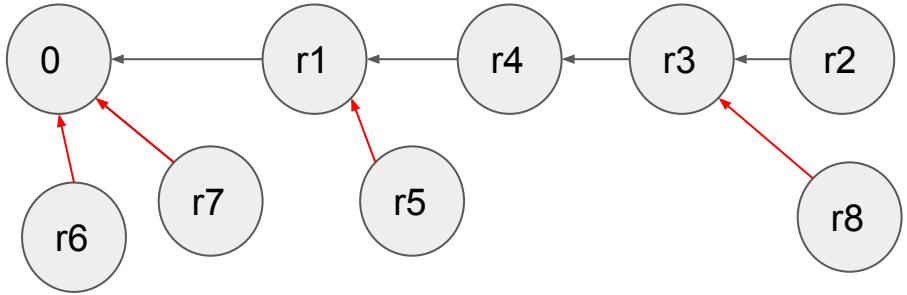


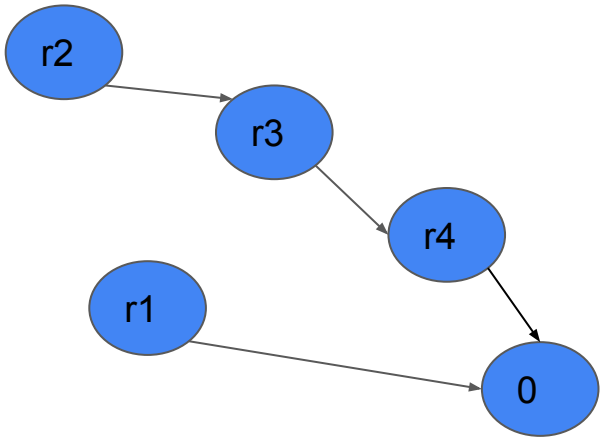


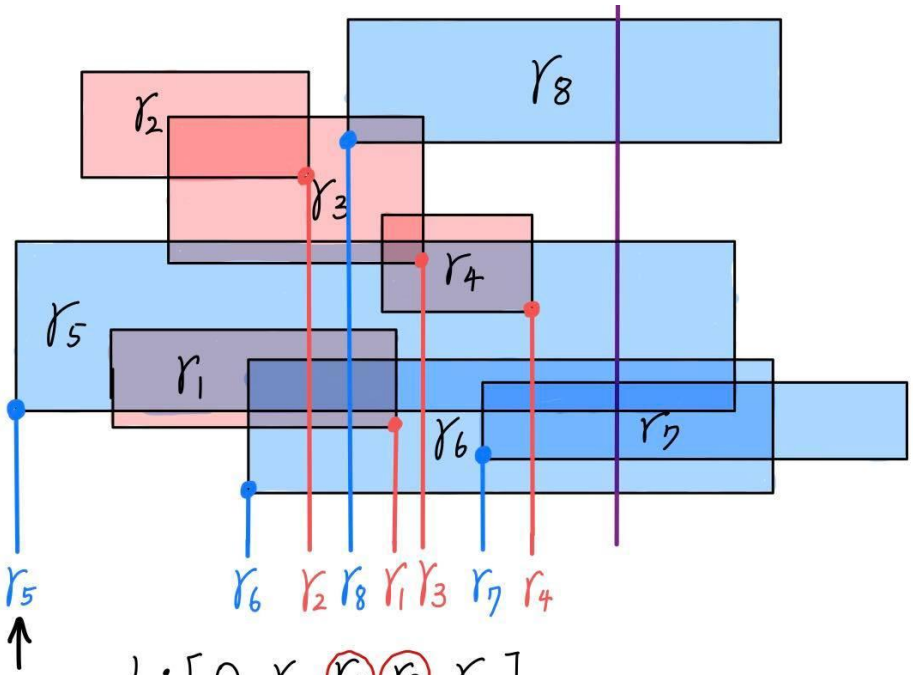


解法步驟

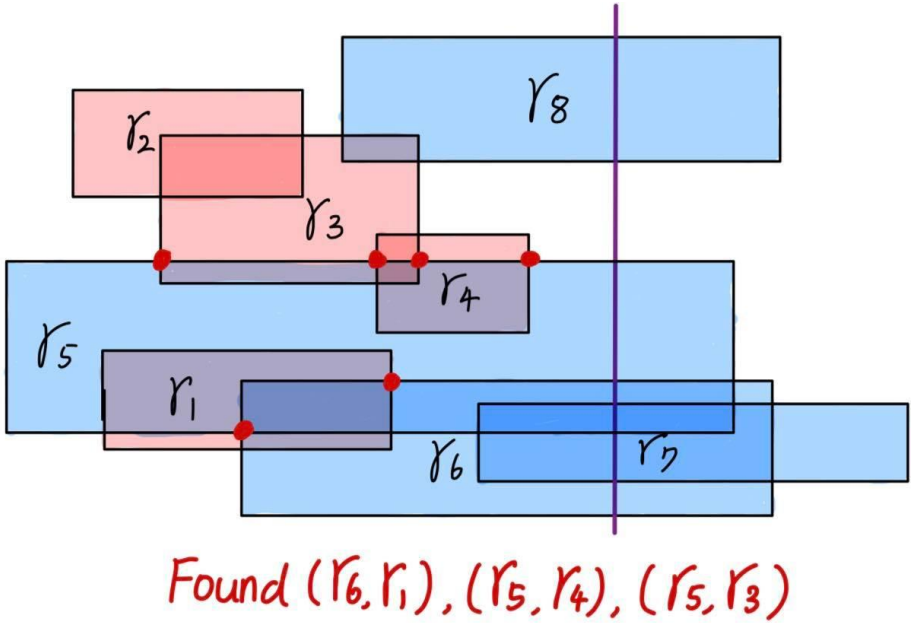
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	0	r_3	r_4	0	r_1	0	0	r_3







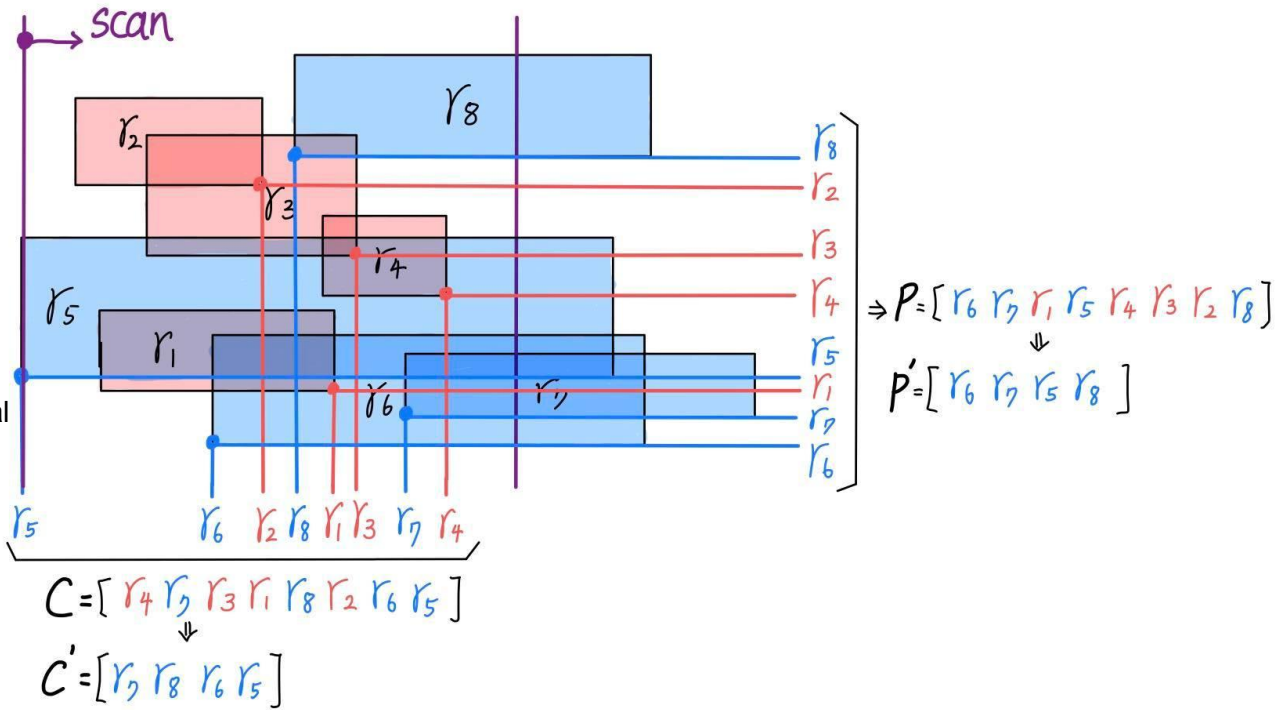
解法步驟-Finish first pass



Second Pass - 左往右掃描

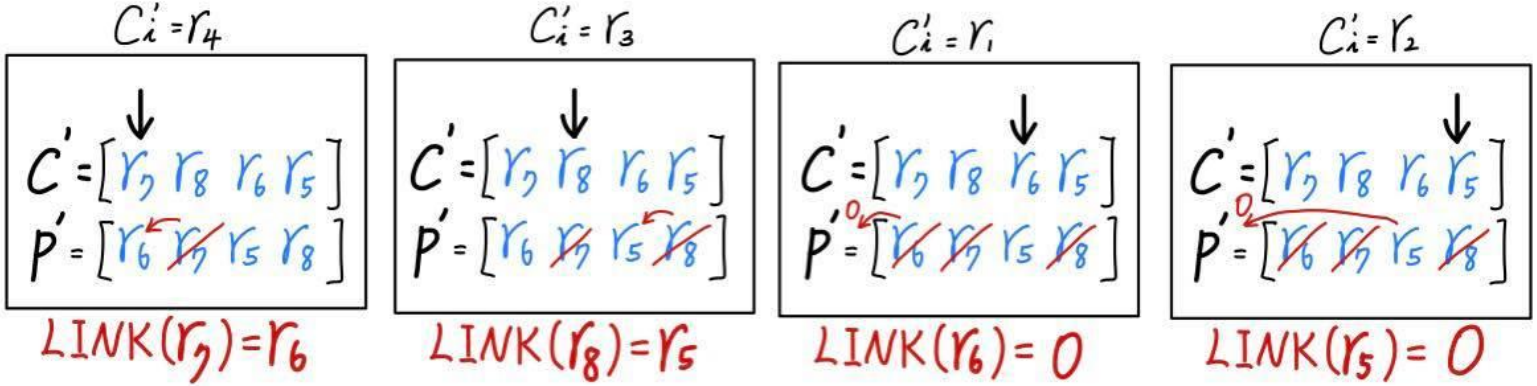
右集合的左邊界 : 放入L

左集合的右邊界 : retrieval

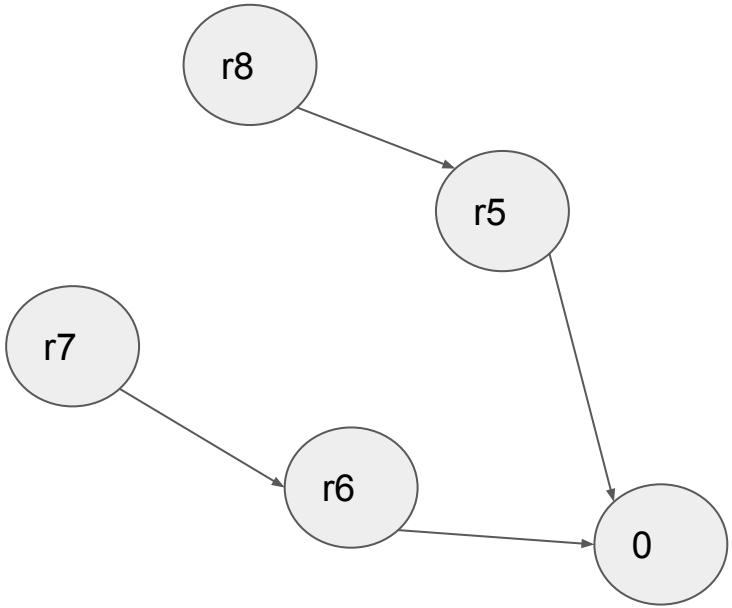


解法步驟-Second Pass

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)					**0**	**0**	r_6	r_5

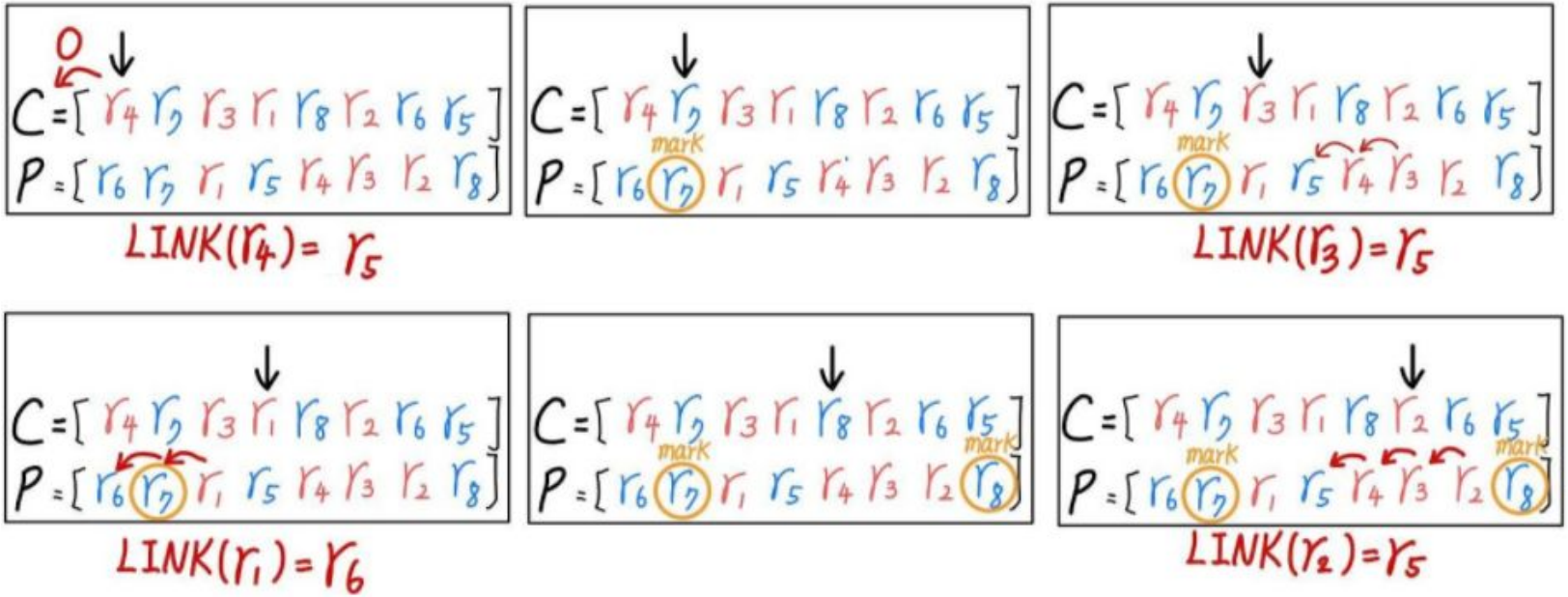


Link for S2

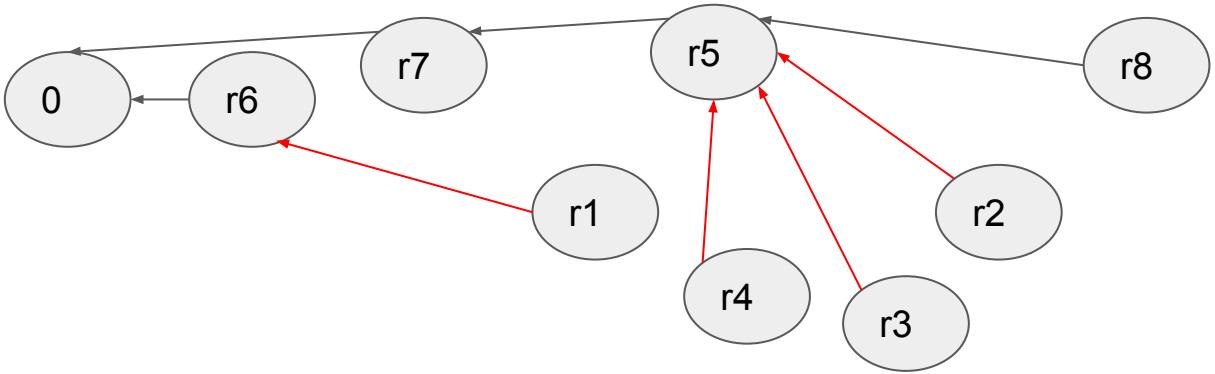


解法步驟-Second Pass

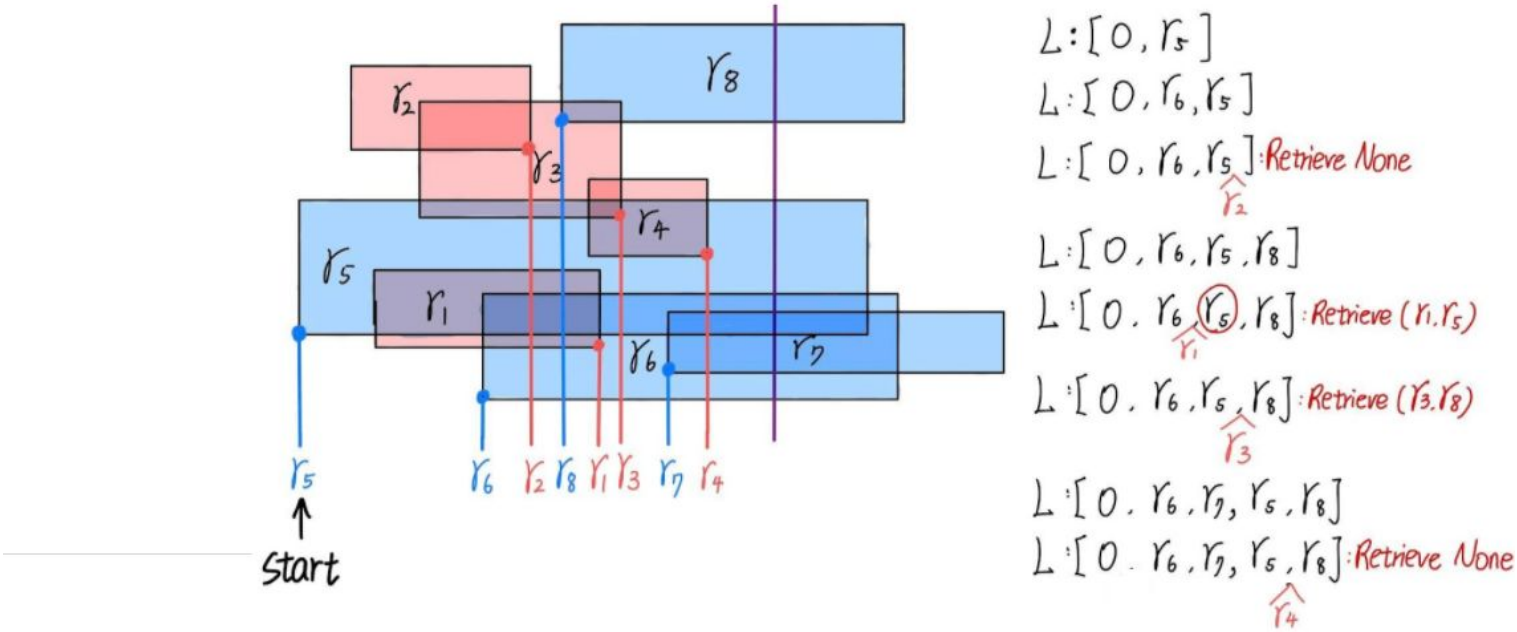
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
LINK(r_i)	r_6	r_5	r_5	r_5	**0**	**0**	r_6	r_5



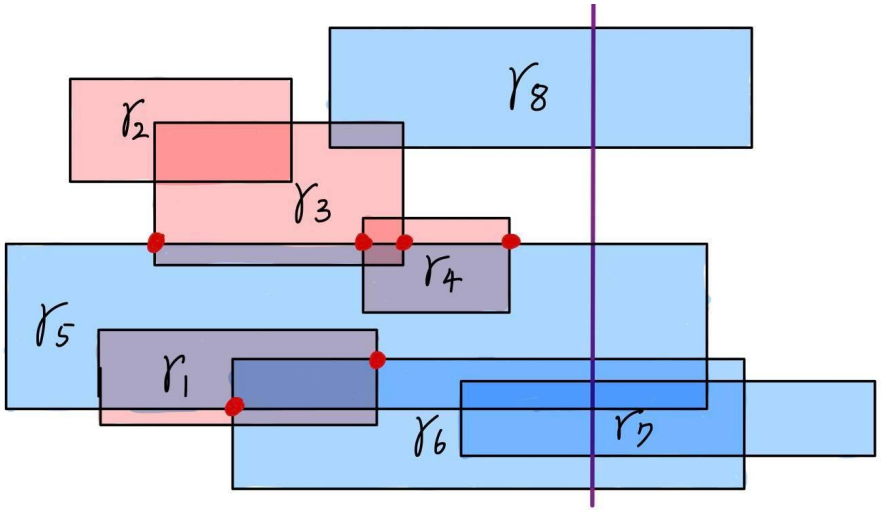
Link for S1



解法步驟


r8
r2
r3
r4
r5
r1
r6
r7
r5 r6 r2 r8 r1 r3 r7 r4
Start
L:[0, r5]
L:[0, r6, r5]
L:[0, r6, r5] Retrieve None
L:[0, r6, r5, r8]
L:[0, r6, r5, r8] Retrieve (r1, r5)
L:[0, r6, r5, r8] Retrieve (r3, r8)
L:[0, r6, r7, r5, r8]
L:[0, r6, r7, r5, r8] Retrieve None

解法步驟 -finish



First Pass: Found (r_6, r_1), (r_5, r_4), (r_5, r_3)

Second Pass: Found (r_1, r_5), (r_3, r_8)

結論與時間複雜度分析

T(n) <= 2T(n/2) + M(n/2, n/2) + O(n), M(n/2, n/2) = O(n) for merge

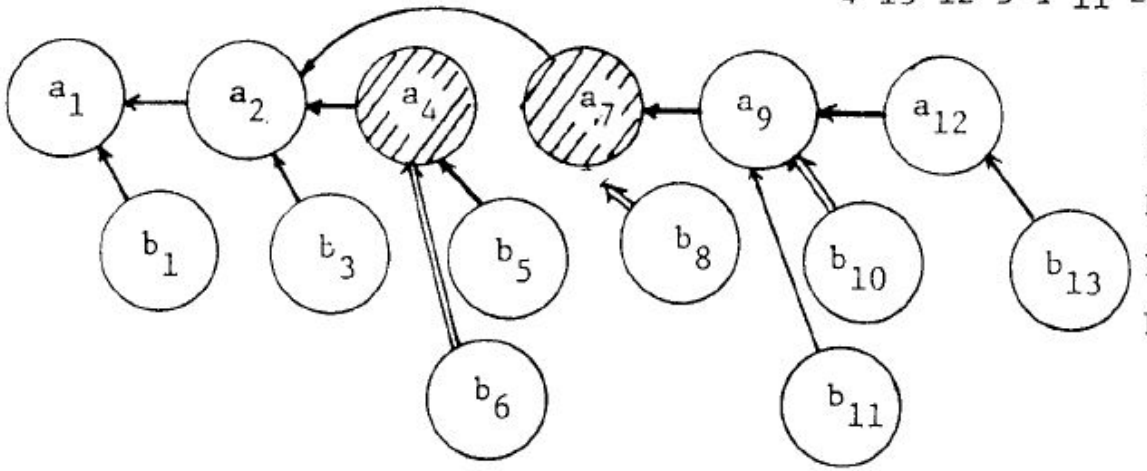
T(n) = O(nlogn) by master theorem

T’(n) = O(nlogn) + s

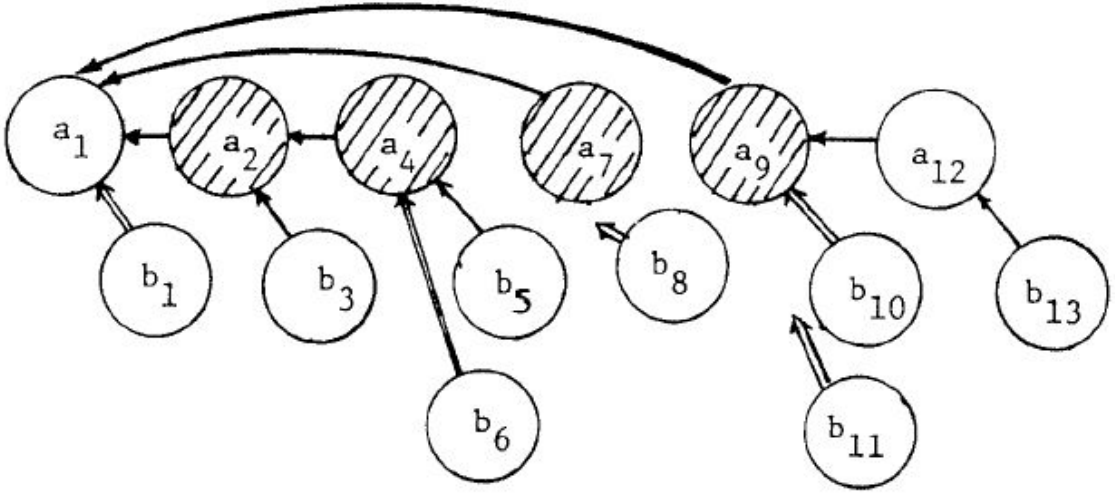
以下為附錄

以下為附錄

C : $(b_4 b_{13} a_{12} b_5 b_1 b_{11} a_2 a_9 b_{10} b_8 a_4 a_7 b_6)$

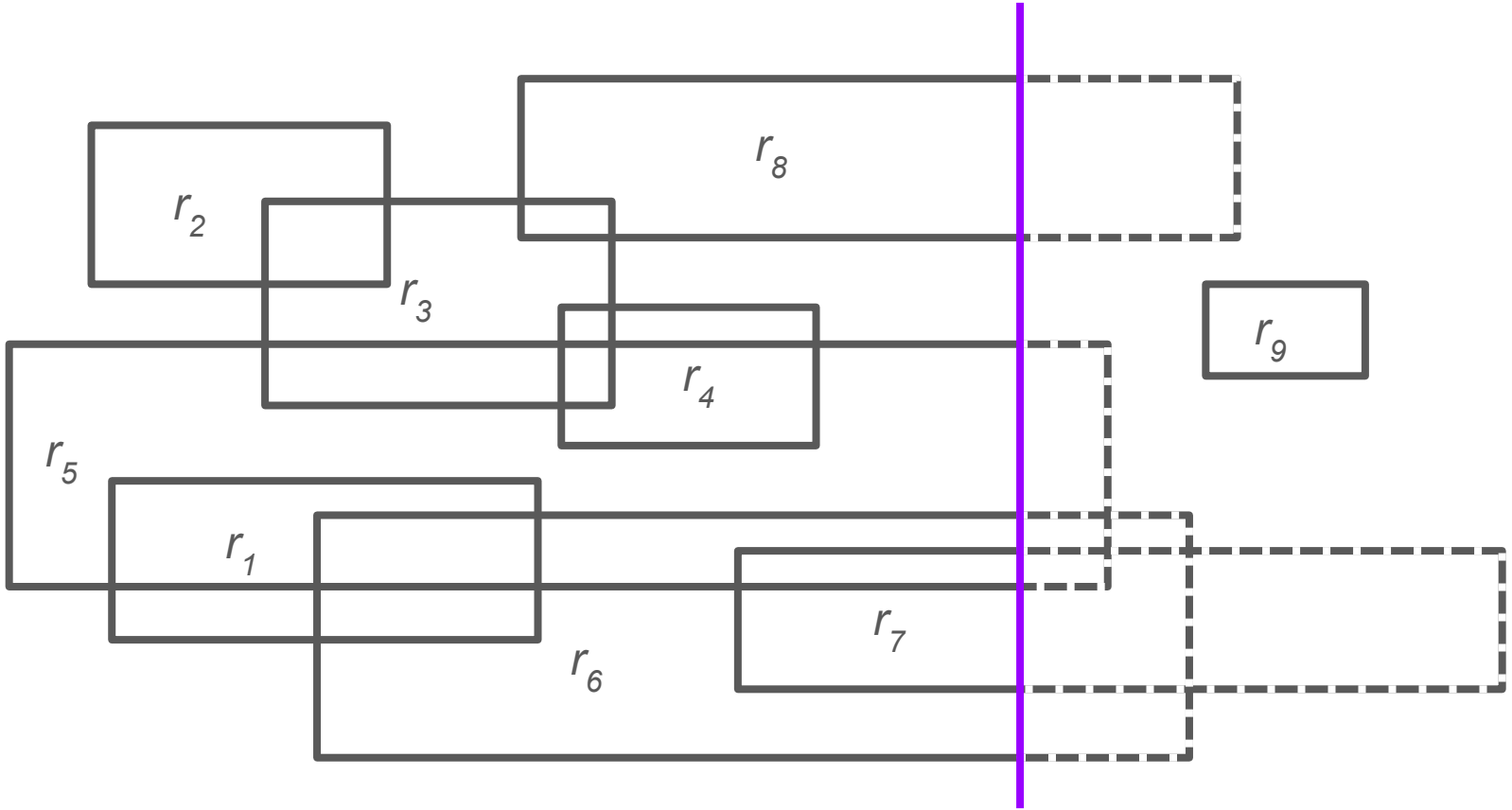


LINK(b_6) = a_4
Mark a_7
Mark a_4
LINK(b_8) = a_2
LINK(b_{10}) = a_9

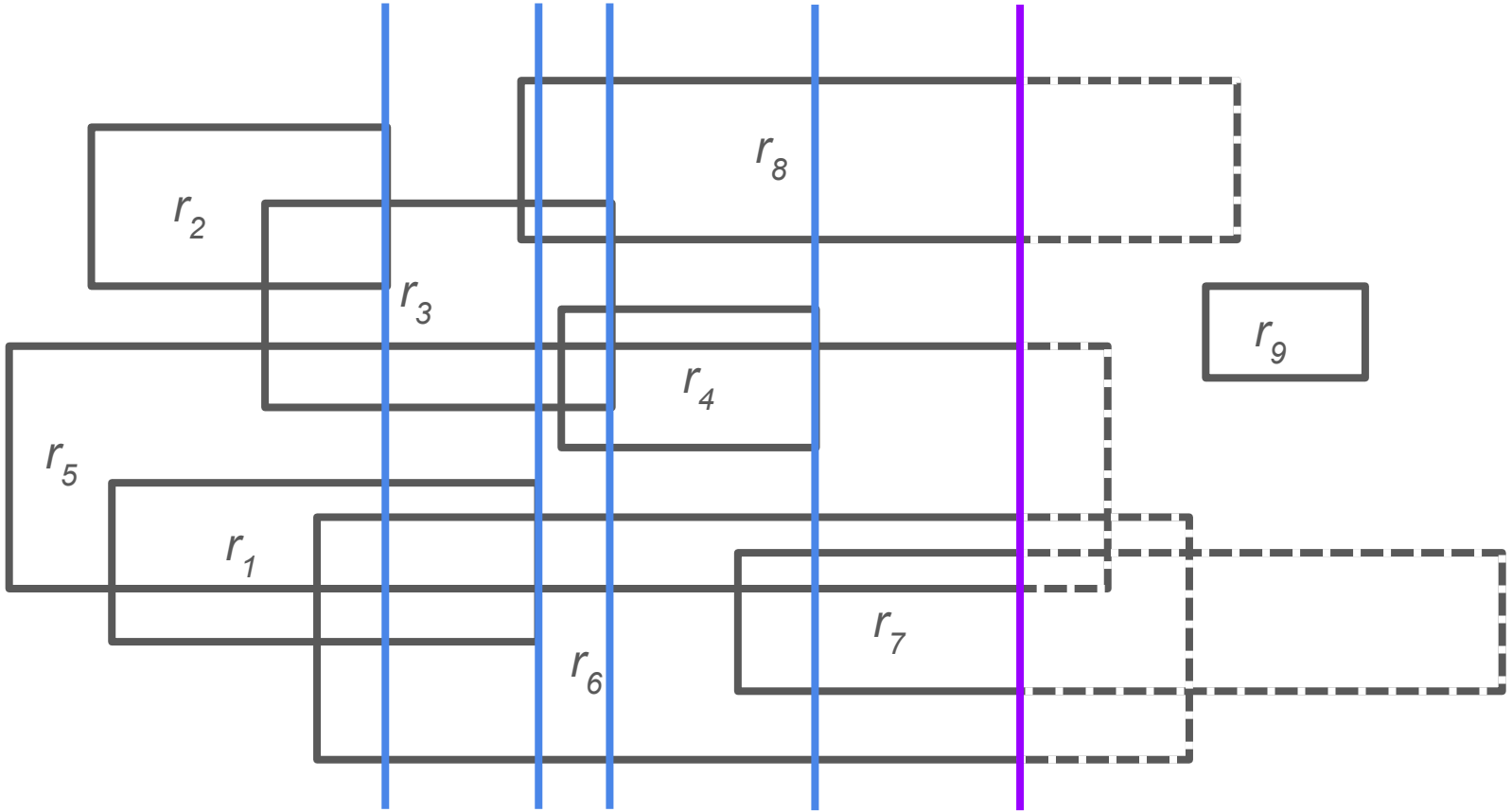


Mark a_9
Mark a_2
LINK(b_{11}) = 0
LINK(b_1) = 0

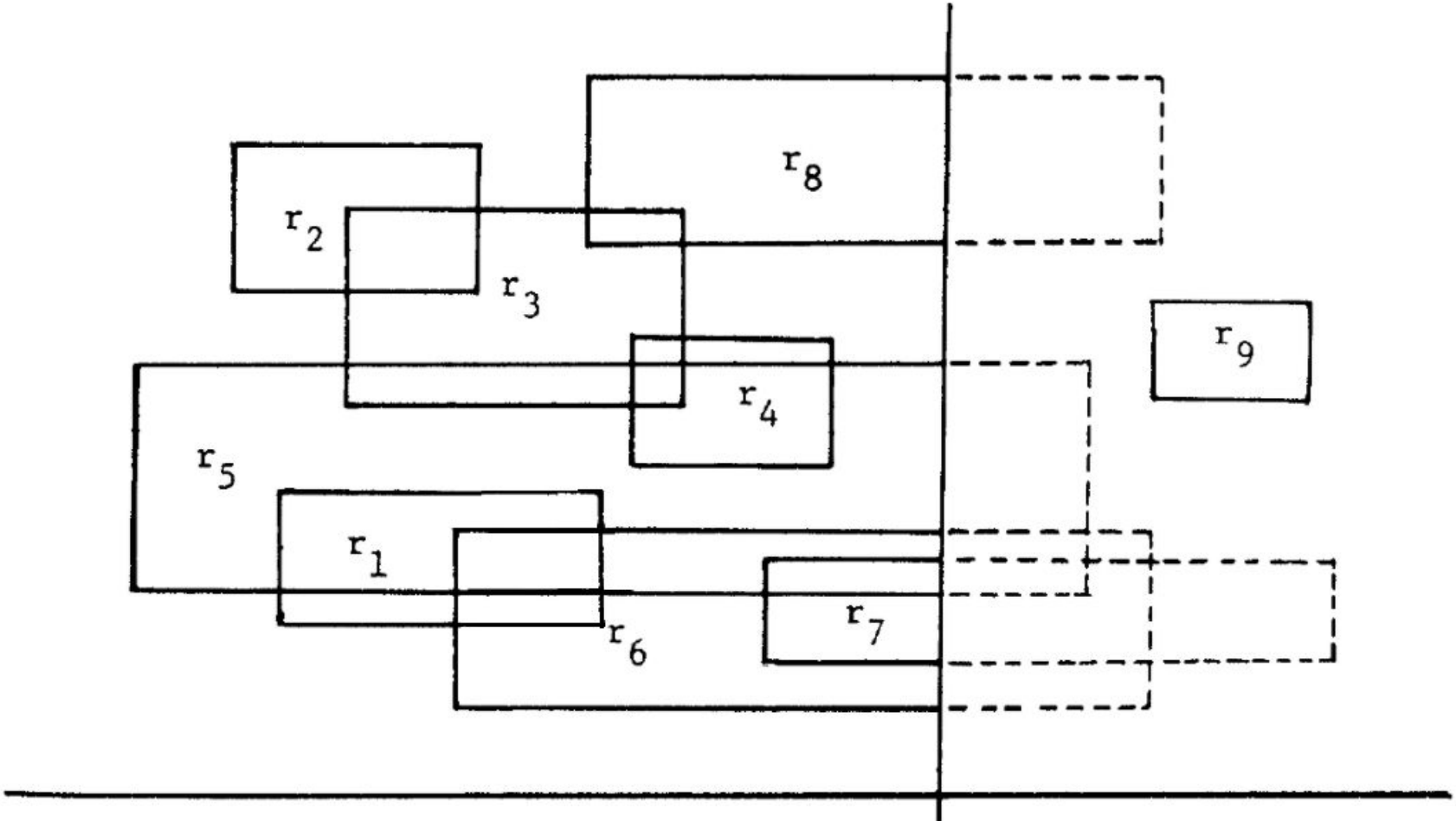
$C=(y_1^4, y_1^7, y_1^3, y_1^1, y_1^8, y_1^2, y_1^6, y_1^5)$ and $P=(y_1^6, y_1^7, y_1^1, y_1^5, y_1^4, y_1^3, y_1^2, y_1^8)$



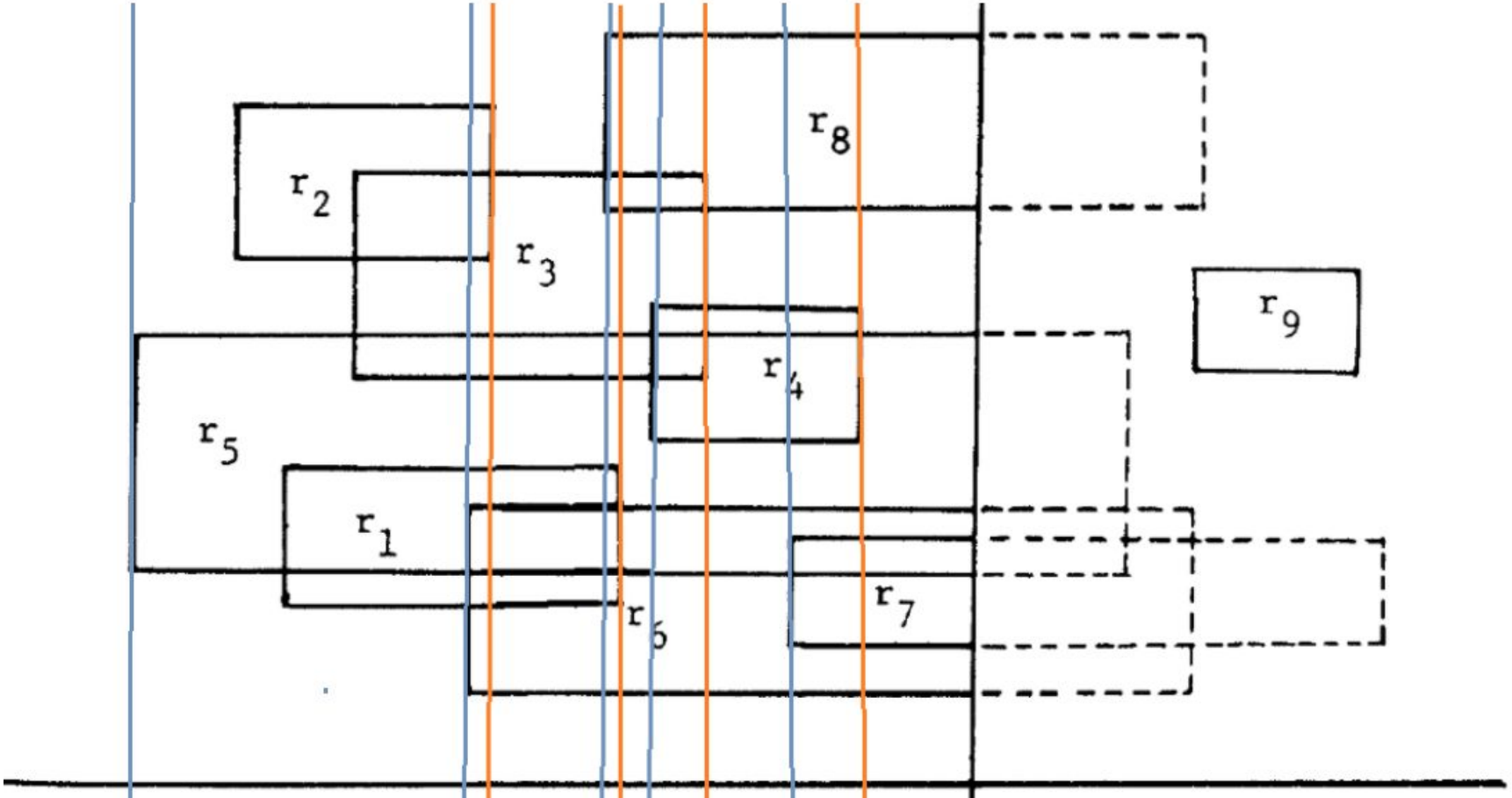
$C=(y_1^4, y_1^7, y_1^3, y_1^1, y_1^8, y_1^2, y_1^6, y_1^5)$ and $P=(y_1^6, y_1^7, y_1^1, y_1^5, y_1^4, y_1^3, y_1^2, y_1^8)$



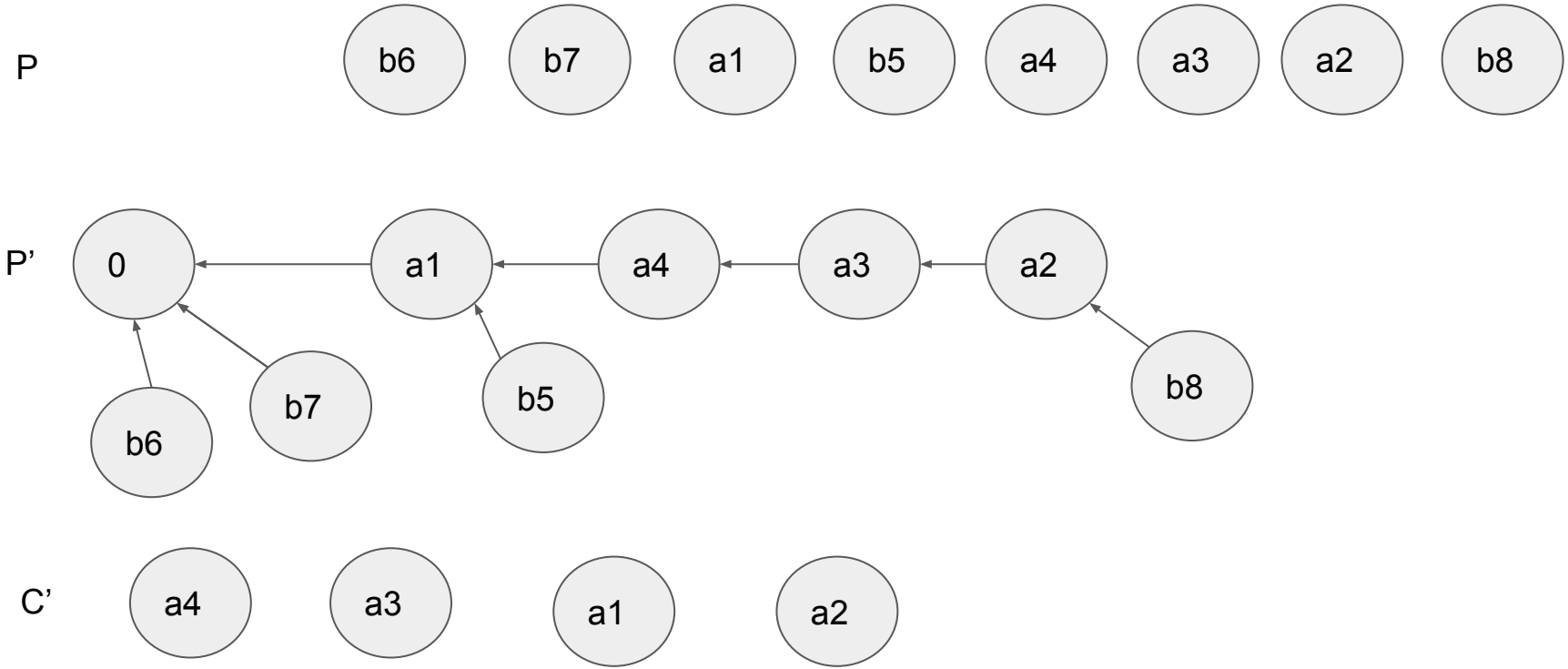
$C = (y_1^4, y_1^7, y_1^3, y_1^1, y_1^8, y_1^2, y_1^6, y_1^5)$ and $P = (y_1^6, y_1^7, y_1^1, y_1^5, y_1^4, y_1^3, y_1^2, y_1^8)$

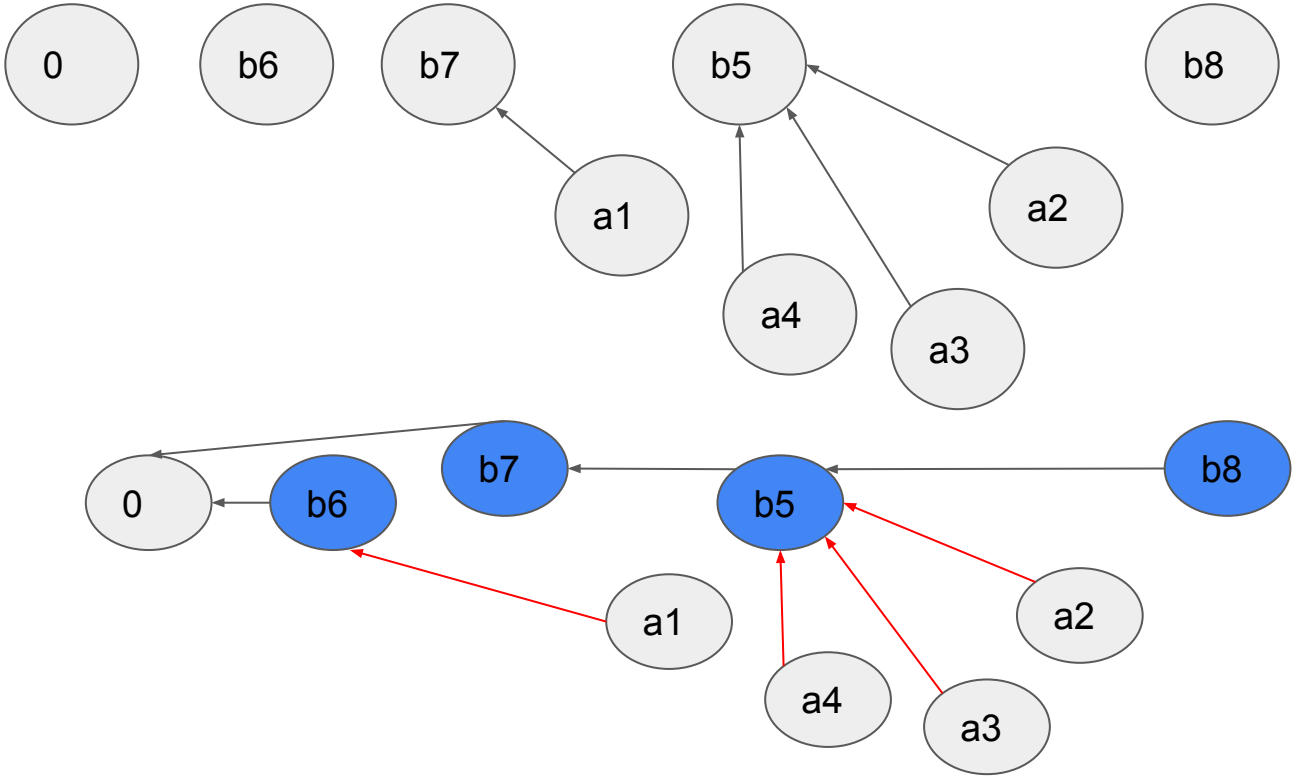


$C = (y_1^4, y_1^7, y_1^3, y_1^1, y_1^8, y_1^2, y_1^6, y_1^5)$ and $P = (y_1^6, y_1^7, y_1^1, y_1^5, y_1^4, y_1^3, y_1^2, y_1^8)$.



Merge: one-pass, second-pass algorith

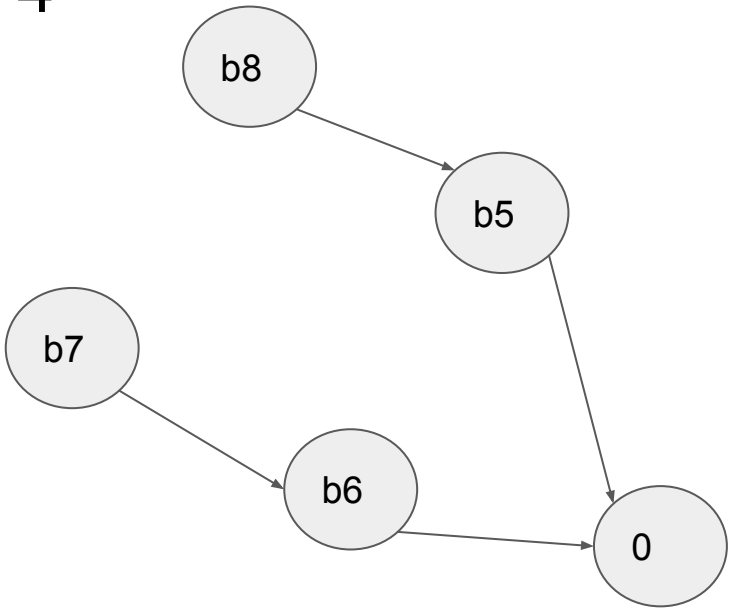




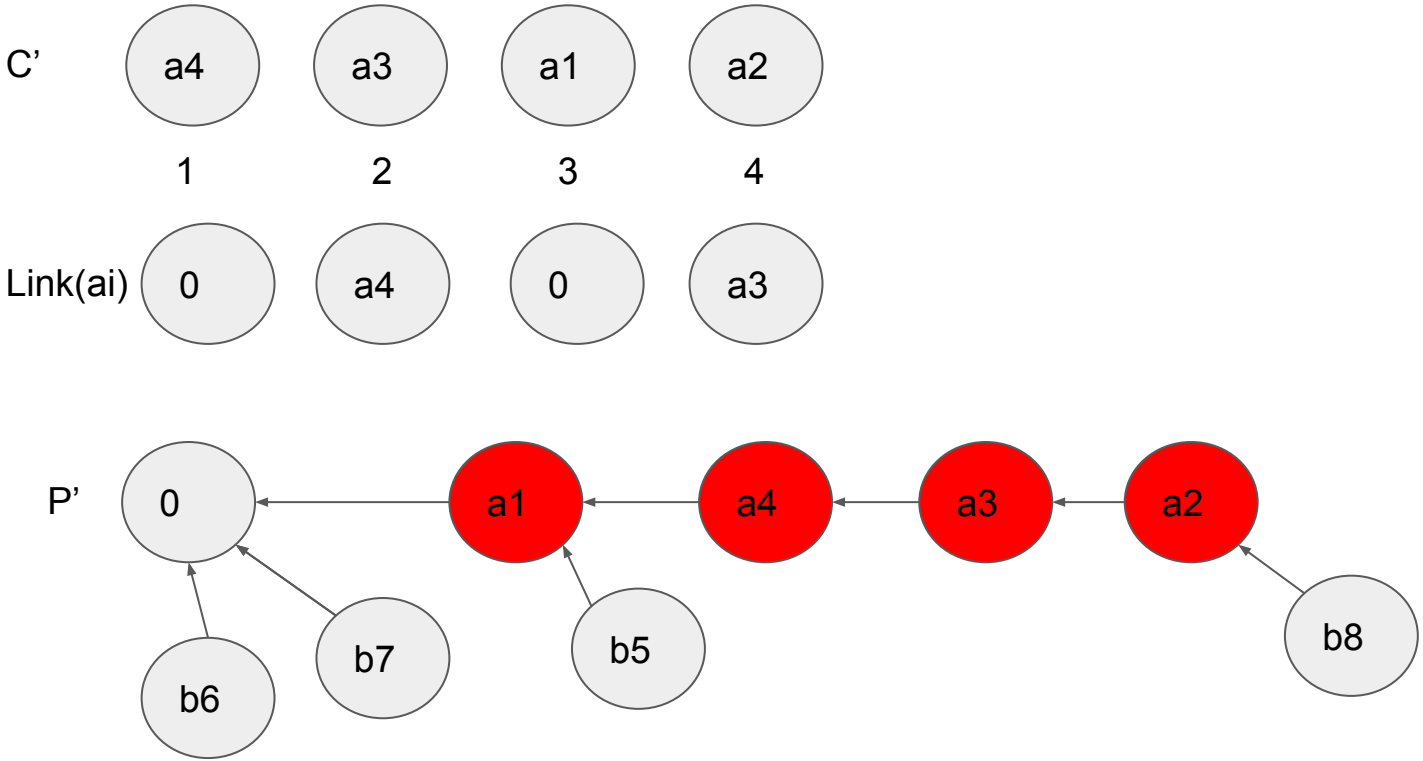
4

5 6 2 8 1 3 7 4

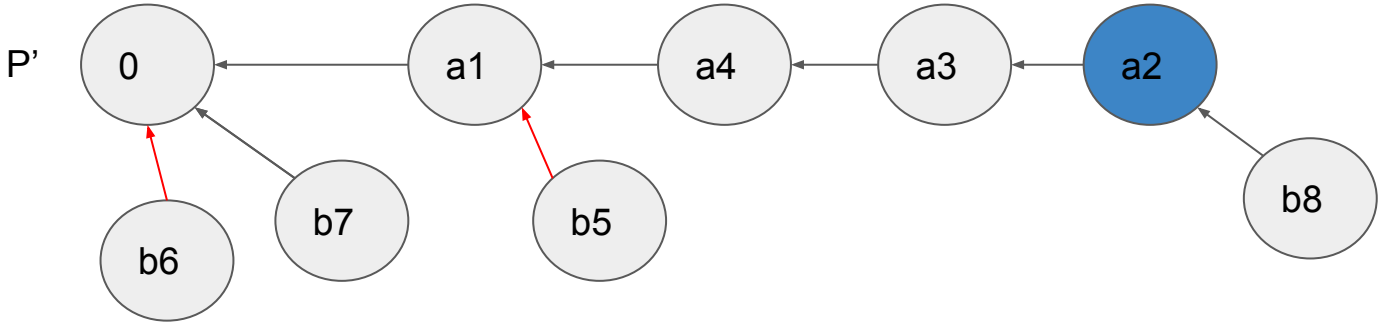
5 6 8 7



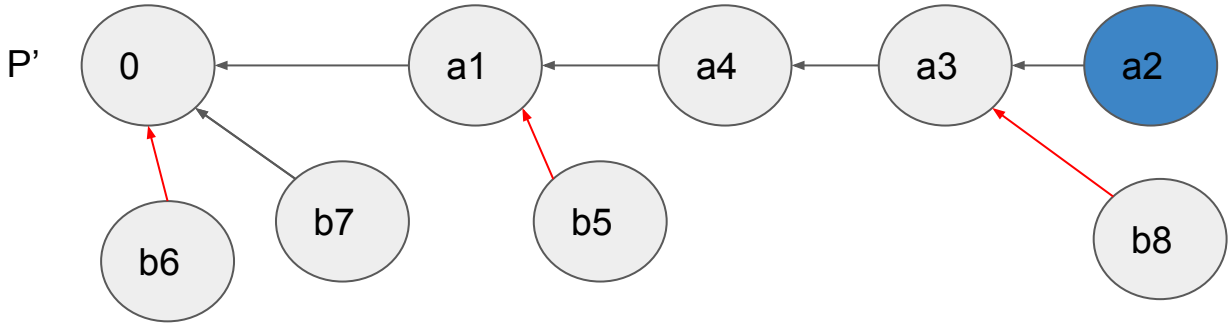
計算2.2 A的link



5 6 2 8 1 3 7 4

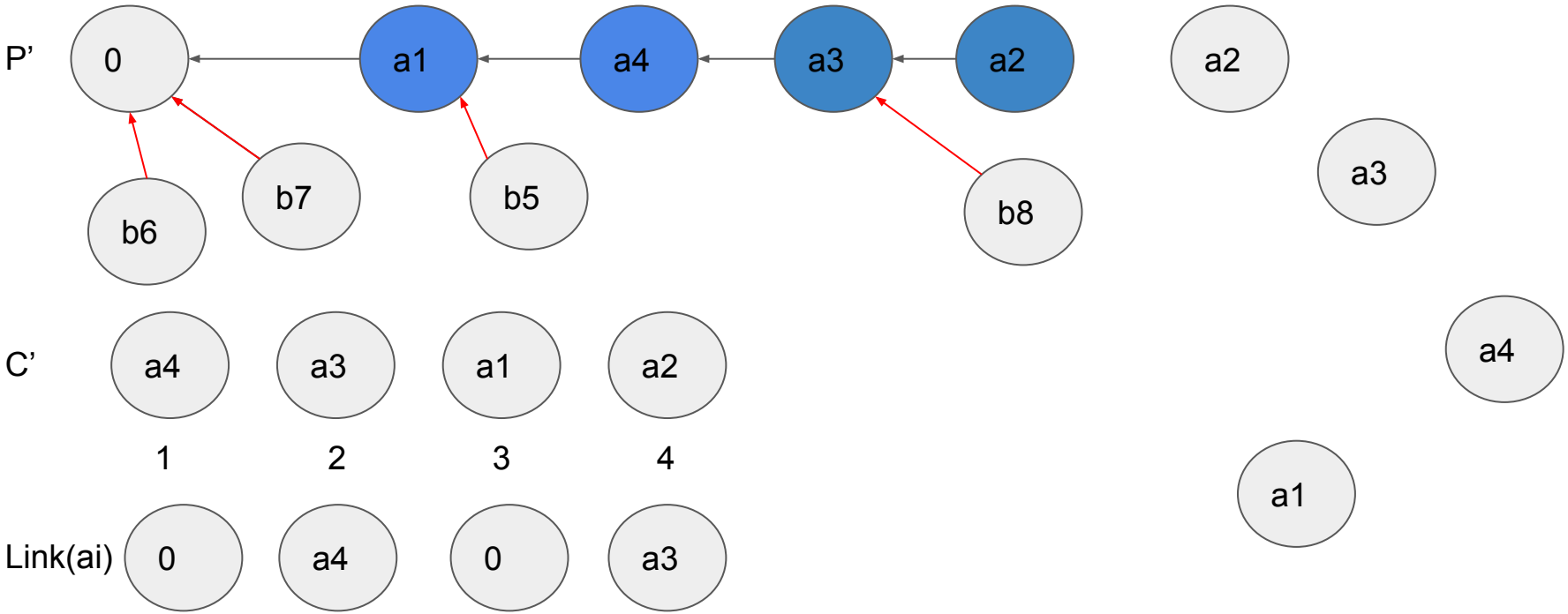


Q = [a2]
c = a3

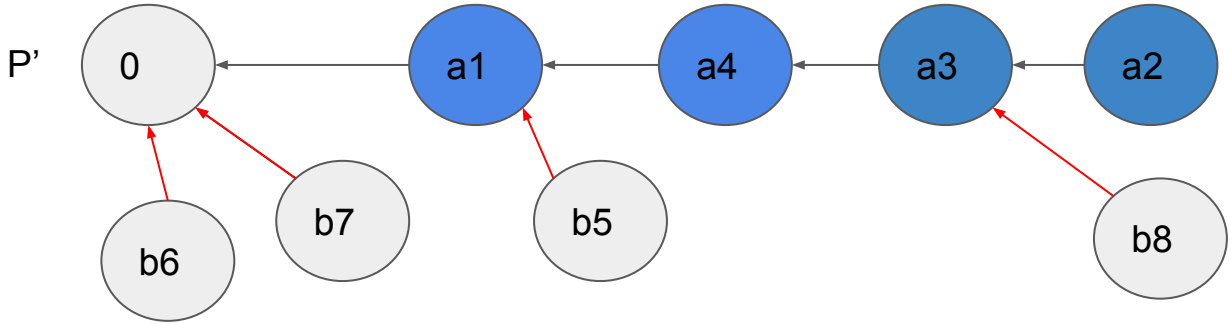


Q = [a2]
c = a3

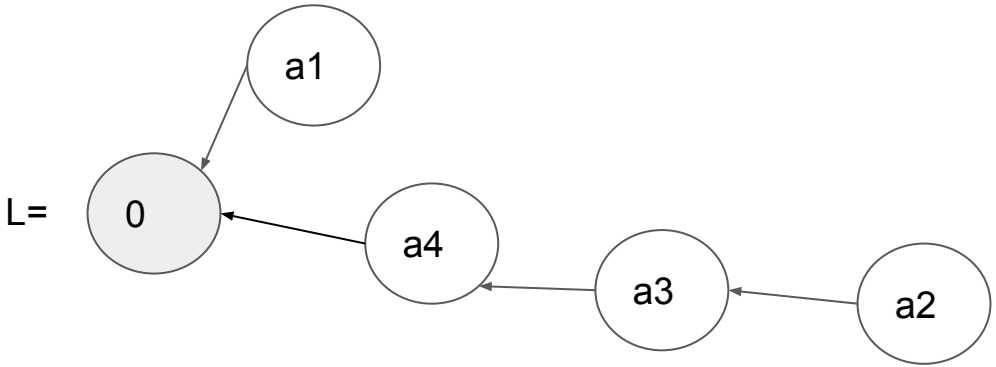
4 7 3 1 8 2 6 5

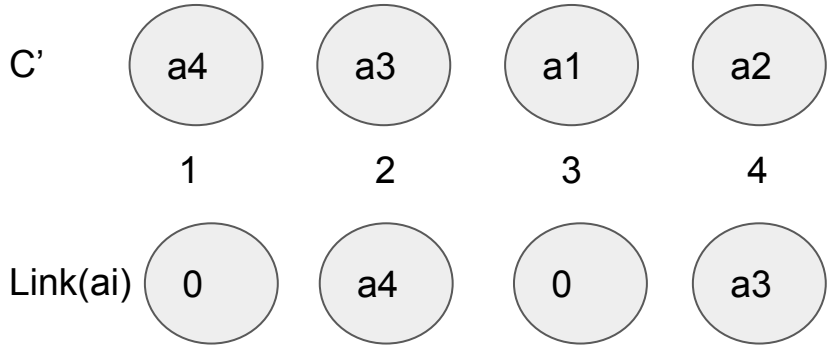

P’
0
a1
a4
a3
a2
b6
b7
b5
b8
C’
a4
a3
a1
a2
1
2
3
4
Link(ai)
0
a4
0
a3
a2
a3
a4
a1

2.4 . Step2 4 7 3 1 8 2 6 5



(6, 1)
(5, 4)
(5, 3)





b8: retrieve: a3
b5: retrieve a1