Lab7 Solution

YAO ZHAO

Lab7.A: Code

- **Gray Code** is an interesting ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).
- Given N, output the N^{th} Gray Code **WITHOUT** leading zeros (It is guaranteed that $N \geq 1$, so you do not need to worry about $N = 0$).

In the standard encoding the least significant bit follows a repetitive pattern of 2 on, 2 off (… 11001100 …); the next digit a pattern of 4 on, 4 off; the *n*th least significant bit a pattern of 2^n on 2^n off. The four-bit version of this is shown below:

Decimal	Binary	Gray				Decimal of Gray
0	0000	0	0	0	0	0
1	0001	0	0	0	1	1
2	0010	0	0	1	1	3
3	0011	0	0	1	0	2
4	0100	0	1	1	0	6
5	0101	0	1	1	1	7
6	0110	0	1	0	1	5
7	0111	0	1	0	0	4
8	1000	1	1	0	0	12
9	1001	1	1	0	1	13
10	1010	1	1	1	1	15
11	1011	1	1	1	0	14
12	1100	1	0	1	0	10
13	1101	1	0	1	1	11
14	1110	1	0	0	1	9
15	1111	1	0	0	0	8

the least significant bit 2 on 2 off

the 2nd least significant bit 4 on 4 off

the 3rd least significant bit 8 on 8 off

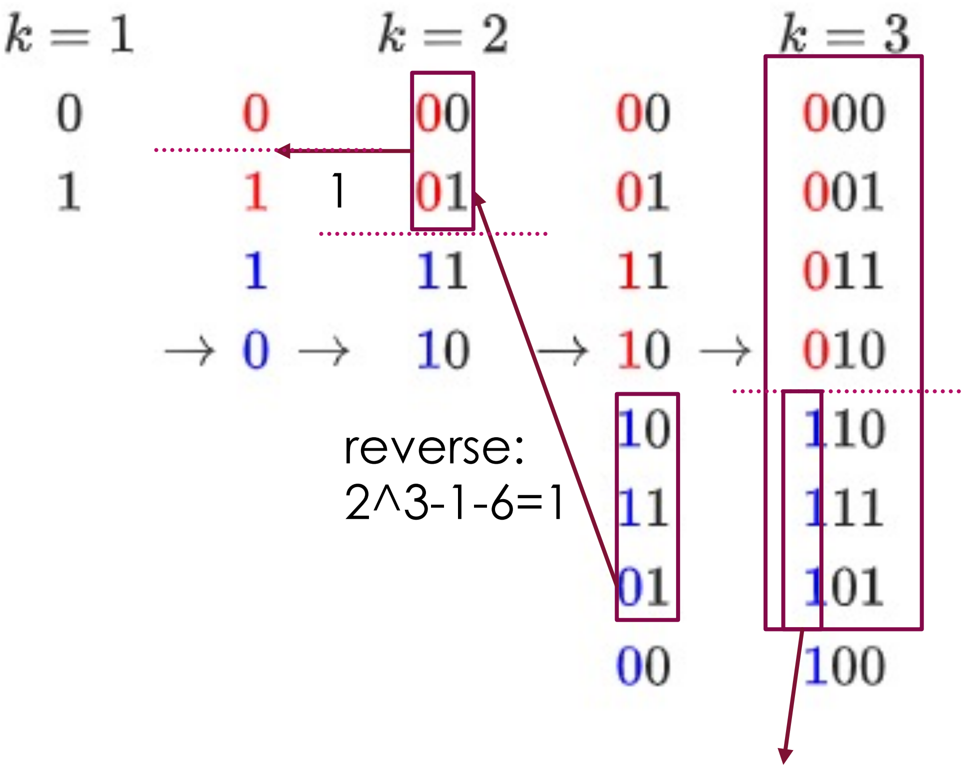
the 4th least significant bit 16 on 16 off

The codes can be constructed through
1.Mirror all existing codes.
2.Add 0 at the left of the original part. Add 1 at the left of the mirrored part.

$$
\begin{array}{cccccccccc}
k=1 & & & & k=2 & & & & k=3 \\
0 & & 0 & & 00 & & 00 & & 000 \\
1 & & 1 & & 01 & & 01 & & 001 \\
 & & 1 & & 11 & & 11 & & 011 \\
 & \rightarrow & 0 & \rightarrow & 10 & \rightarrow & 10 & \rightarrow & 010 \\
 & & & & & & 10 & & 110 \\
 & & & & & & 11 & & 111 \\
 & & & & & & 01 & & 101 \\
 & & & & & & 00 & & 100
\end{array}
$$

Sample 1 Output

101



1≥2^0?yes
print 1

1≥2^1 ? no
print 0

6≥2^2?yes
print 1

Sample 1 Input

6

bit length = 3

Sample 2 Input

100

bit length = 7

100 ≥ 2^6?yes
print 1

reverse:2^7-1-100=27
27 ≥ 2^5?no
print 0

27 ≥ 2^4?yes
print 1

reverse:2^5-1-27=4
4 ≥ 2^3?no
print 0

4 ≥ 2^2?yes
print 1

reverse:2^3-1-4=3
3≥2^1?yes
print 1

reverse:2^2-1-3=0
0≥2^0?no
print 0

Sample 2 Output

1010110

Lab7.B: Hot Spring

- Once there was a magic hot spring. Whoever steeped inside gains happiness ---- or suffering as well.
- The hot spring is only available at moment $1, 2, \dots, N$. The water changes every moment. If someone is enjoying the hot spring at moment i, he or she will gain v_i "happiness". Note that a negative v_i means he or she actually gains suffering.
- A visitor has only one chance to enjoy the hot spring. He or she may start enjoy the hot spring at some moment l, and finishes at some moment r $(l \le r)$. The visitor's final happiness will be $\sum_{i=l}^{r} v_i$. However, if a visitor does not use the hot spring at all, the final happiness will be $-\infty$.
- Now there are Q visitors. The i^{th} visitor arrives at moment L_i and must leave at the end of moment R_i. Within their own time limits, visitors can choose the moments they want to enjoy the hot spring.
- Help each visitor find his or her maximal happiness.

Sample Input

5
1 7 -3 -4 5
3
3 4
2 5
1 5

1	2	3	4	5
1	7	-3	-4	5

3 4

3 3 → -3
3 4 → -7
4 4 → -4

-3

1	2	3	4	5
1	7	-3	-4	5

2 2 →7
2 3 →4
2 4 →0
2 5 →5
3 3 →-3
3 4 →-7
3 5 →-2
4 4 →-4
4 5 →1
5 5 →5

→ **7**

1	2	3	4	5
1	7	-3	-4	5

1 5

2 2 →7	1 1 →1
2 3 →4	1 2 →8
2 4 →0	1 3 →5
2 5 →5	1 4 →1
3 3 →-3	1 5 →6
3 4 →-7	
3 5 →-2	
4 4 →-4	
4 5 →1	
5 5 →5	

→ **8**

Sample Output

-3
7
8

If only 1 query [1, n]:

1. Recursively Divide the given array in two halves
2. Return the maximum of following three

   ① Maximum subarray sum in left half (Make a recursive call)

   ② Maximum subarray sum in right half (Make a recursive call)

   ③ Maximum subarray sum such that the subarray crosses the midpoint

max_cross_mid = ?

How to solve: Maximum subarray sum such that the subarray crosses the midpoint?

1	2	3	4	5
1	7	-3	-4	5

0	1	2	3	4	5
0	1	8	5	1	6

$let\ \ sum(i,j) = \sum_{n=i}^{j} a_n = s[j] - s[i-1]$

mid = 3

max_cross_mid = max (sum(1,4), sum(2,4), sum(3,4), sum(1,5), sum(2,5), sum(3,5))

= max (sum(1,4),sum(1,5))-min(sum(1,2), sum(1,1),sum(0,0)

=max (s[4], s[5])-min(s[0], s[1], s[2])

=6

min=0

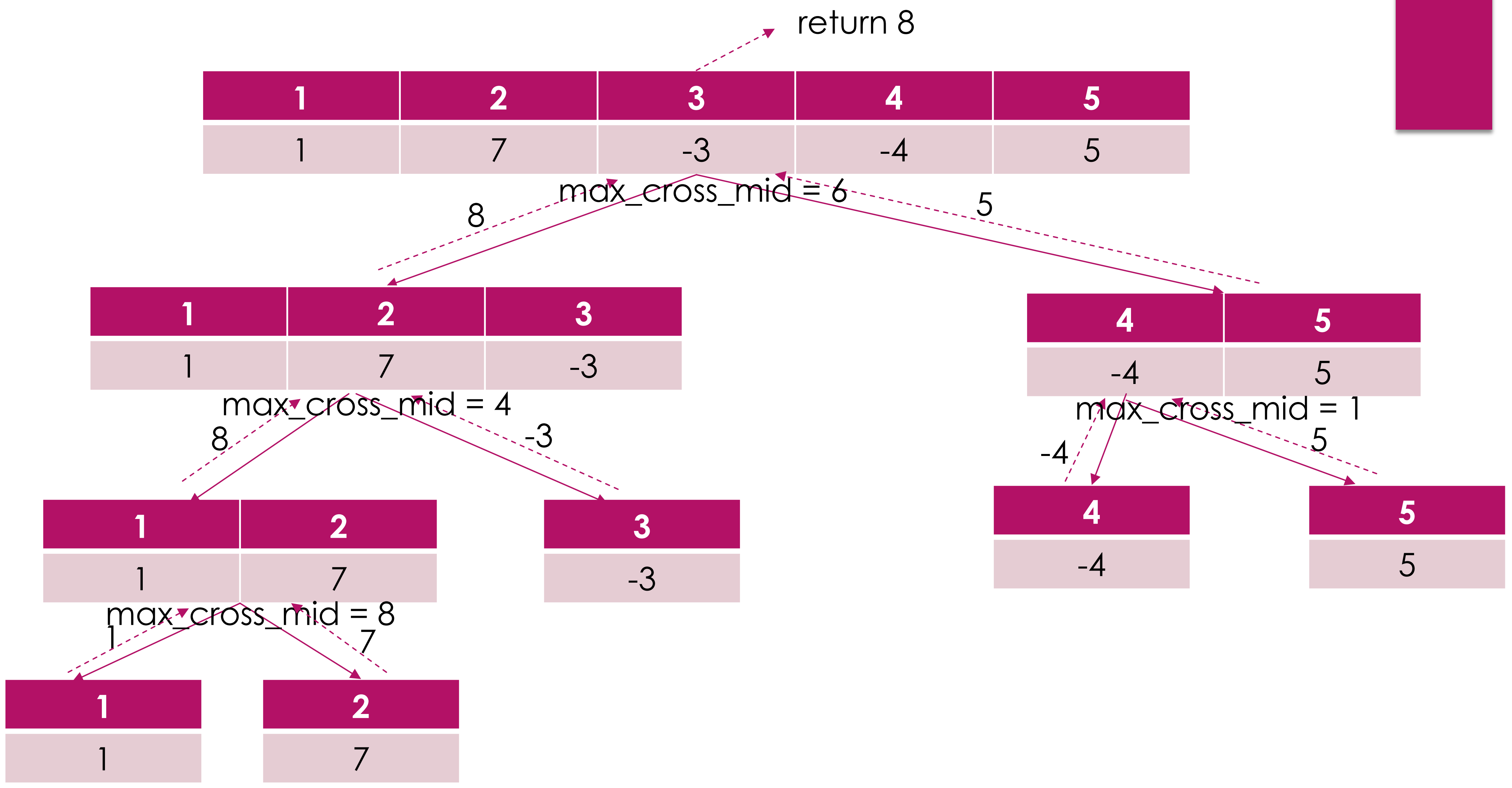
max=6

0	1	2	3	4	5
0	1	8	5	1	6

$let\ \ sum(i,j) = \sum_{n=i}^{j} a_n = s[j] - s[i-1]$
mid = 3
max_cross_mid = max (sum(1,4), sum(2,4), sum(3,4), sum(1,5), sum(2,5), sum(3,5))
= max (sum(1,4),sum(1,5))-min(sum(1,2), sum(1,1),0)
=max (s[4], s[5])-min(s[0], s[1], s[2])
=6

More general representation

left: l right: r mid:m
max_cross_mid =max (s[m+1], s[m+2],…s[r])-min(s[m-1], s[m-2], …, s[l]) **O(n)**

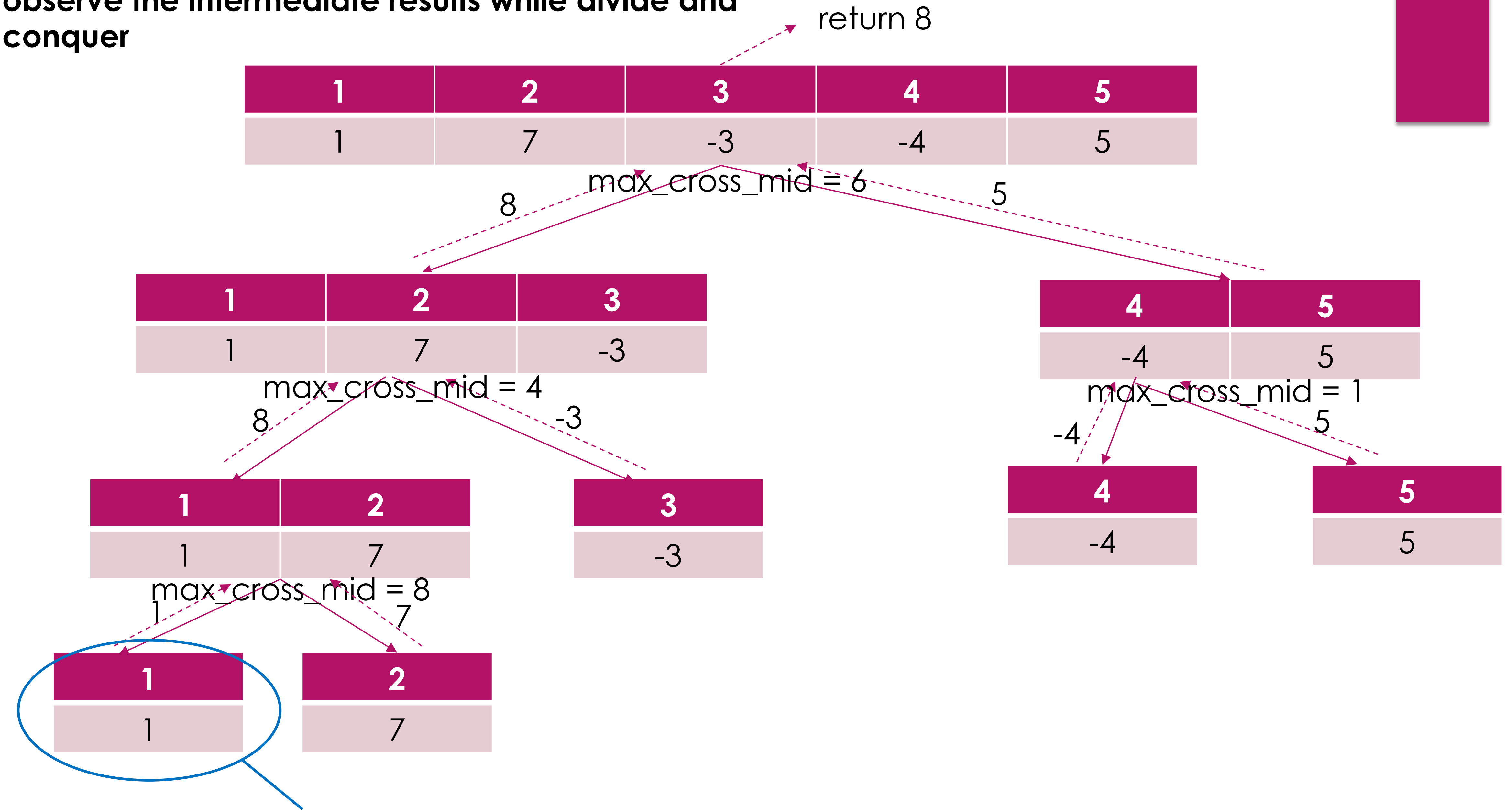


query [1, n]: O(nlogn)

$query\ Q\ times: O(QNlogN)$ $\quad 1 \leq\ N, Q \leq\ 2\times10^5$

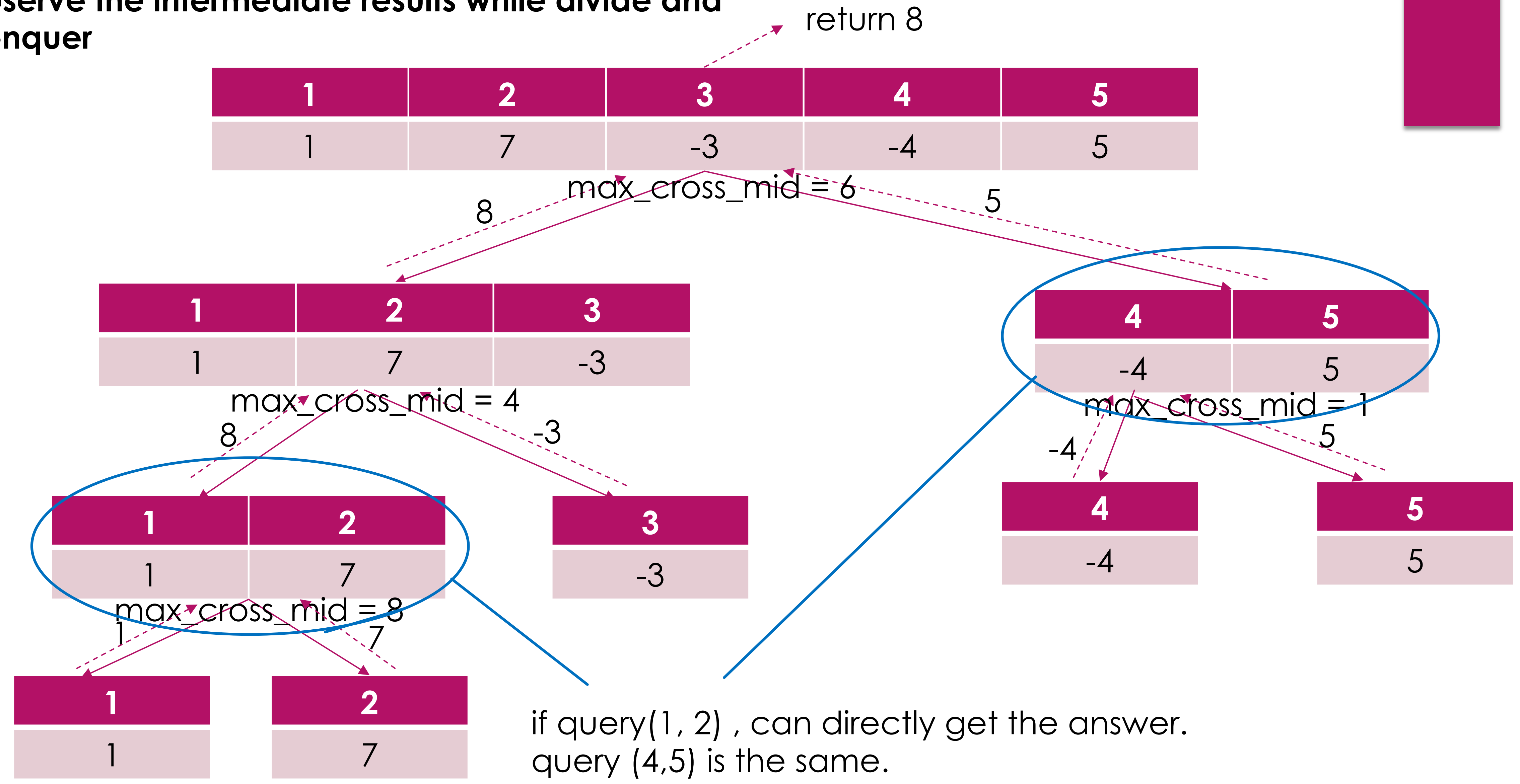
X

observe the intermediate results while divide and conquer

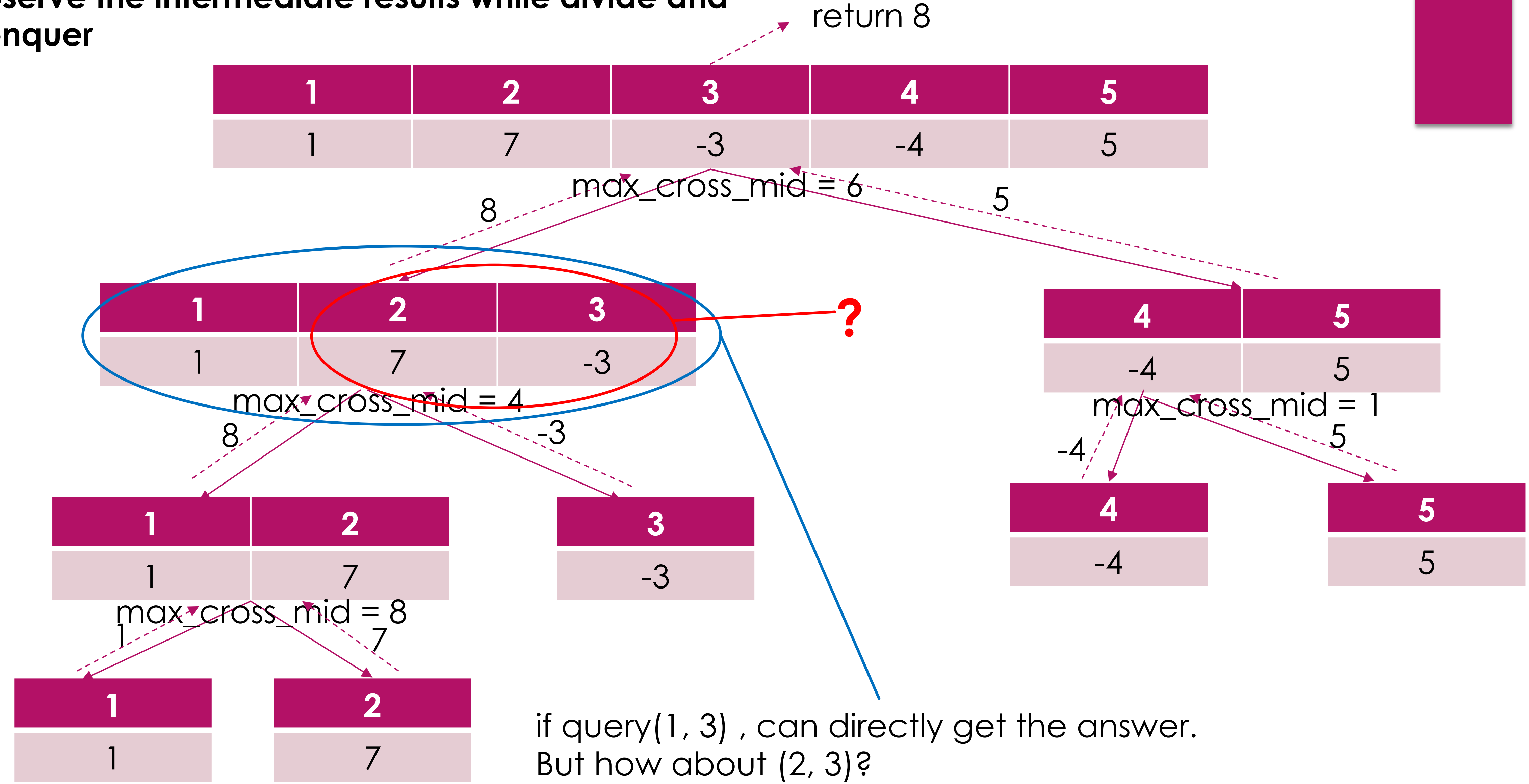


if query(1, 1) , can directly get the answer. query (2,2) (3,3)..(n, n) is the same.

observe the intermediate results while divide and conquer



observe the intermediate results while divide and conquer



1	2	3
1	7	-3

?

l=2 mid=2 r=3

1	2	3
1	8	5

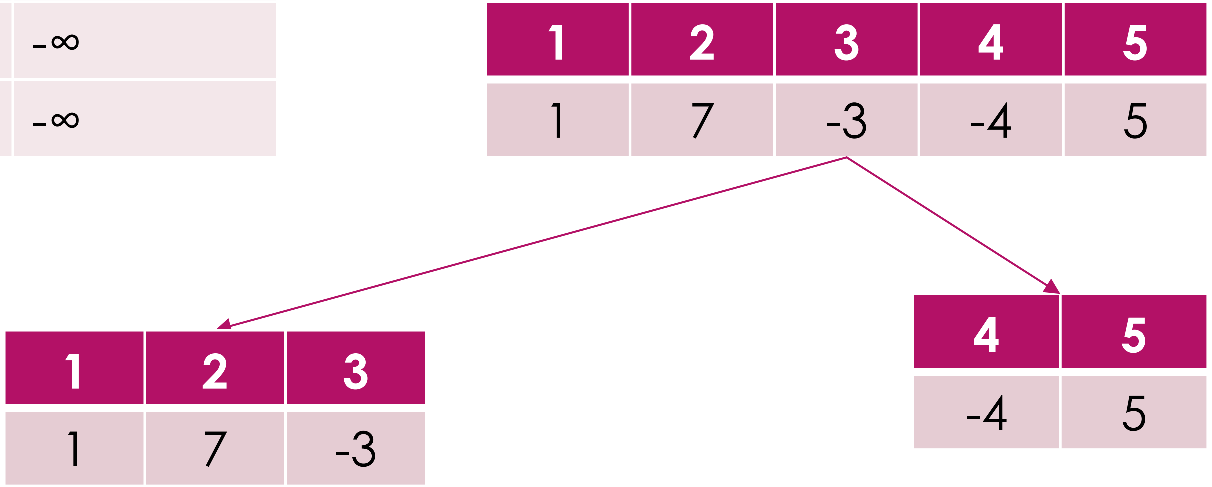
max_cross_mid = max (s[m+1])- min(s[m-1]) = 4 see Page13

max_left = query(2,2)
max_right = query(3,3)

Do we need to store all **intermediate results? No**
N^2 = 40000M

The results can be updated gradually during the divide-and-conquer process

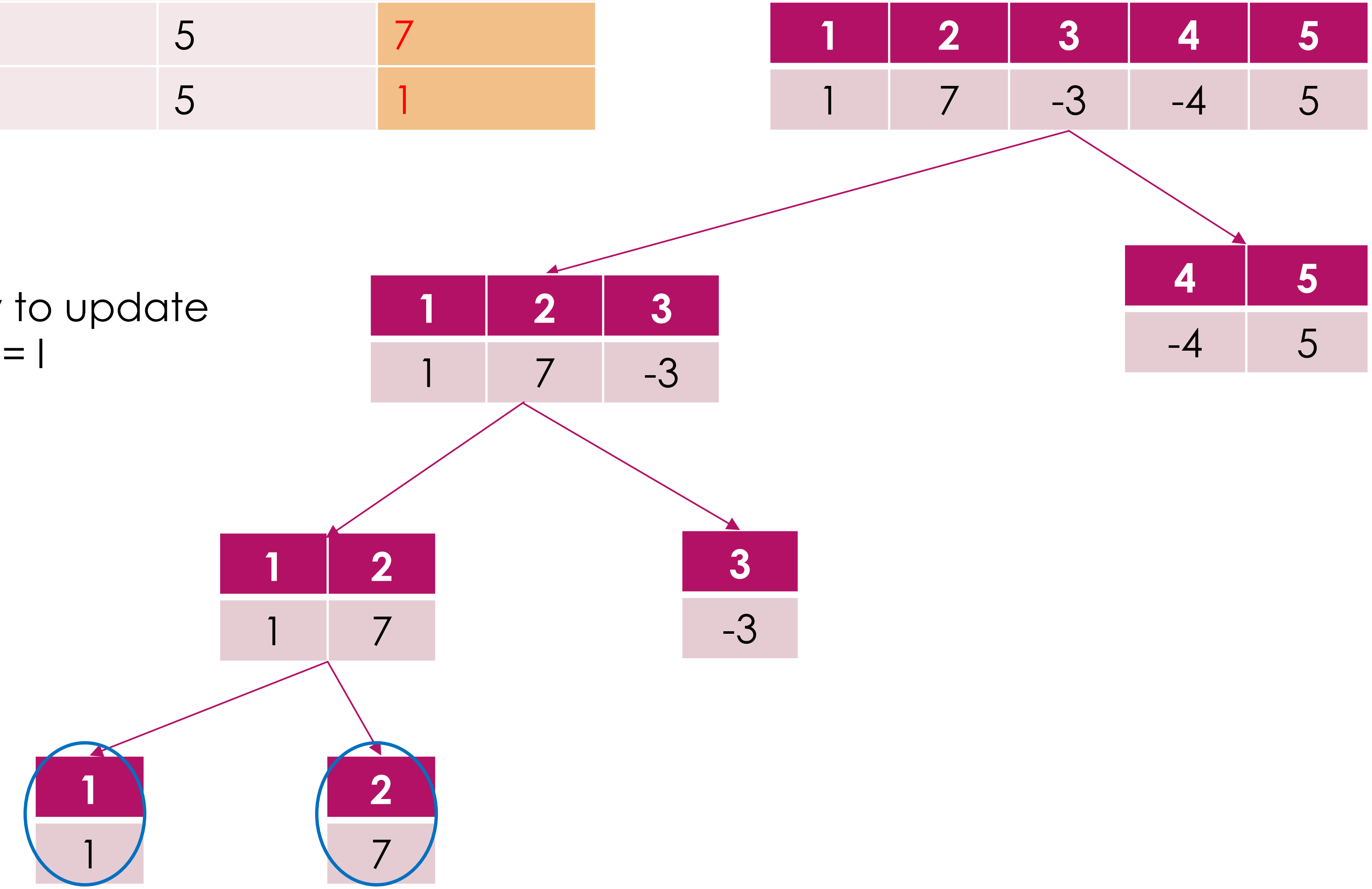
Query	l	r	answer
3 4	3	4	-∞
2 5	2	5	-∞
1 5	1	5	-∞



Query	l	r	answer
3 4	3	4	-∞
2 5	2	5	7
1 5	1	5	1

try to update
ql = l

l = r



Query	ql	qr	answer
3 4	3	4	-∞
2 5	2	5	7
1 5	1	5	8

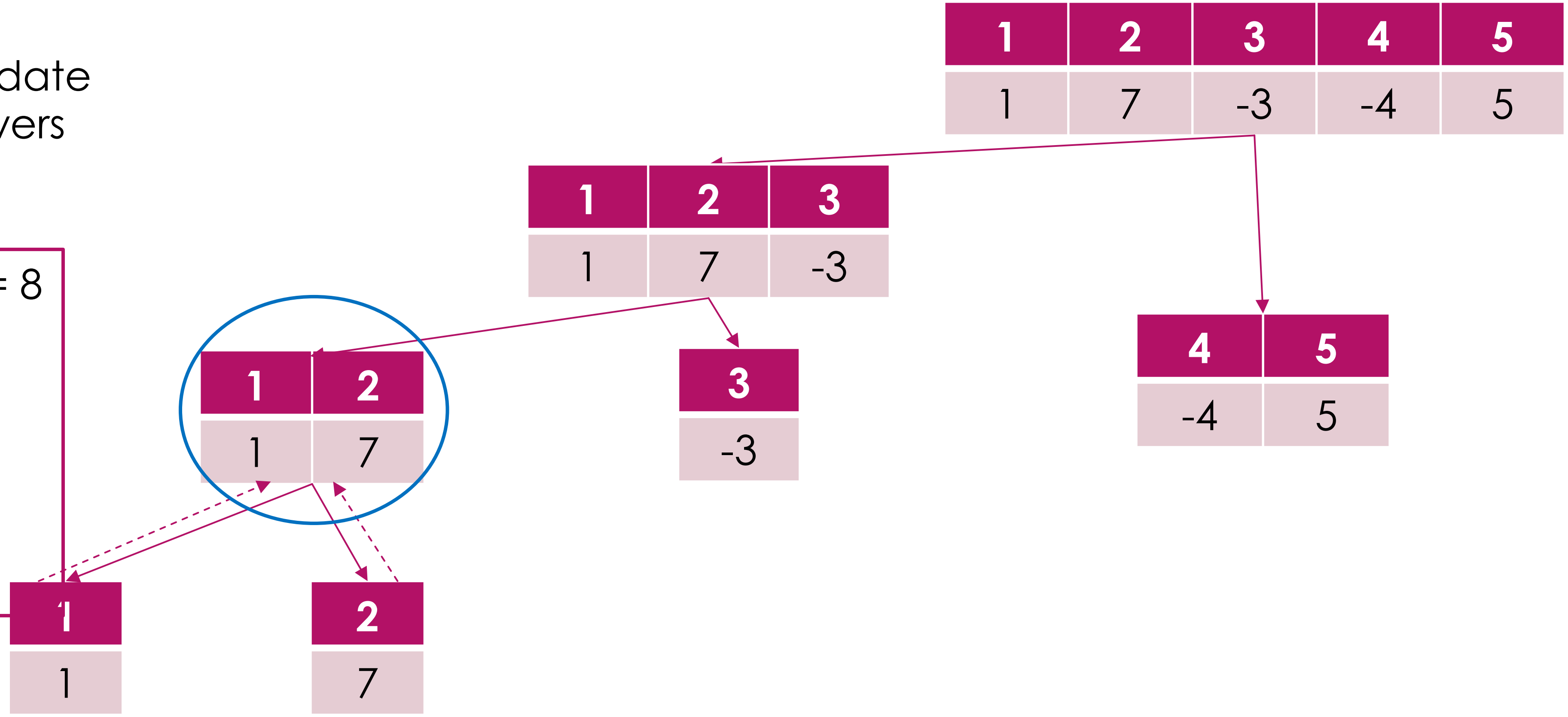
ql>mid and qr > r no update

ql>mid and qr > r no update

$[1,2] \in [1,5]$ $best = 8 > 1$

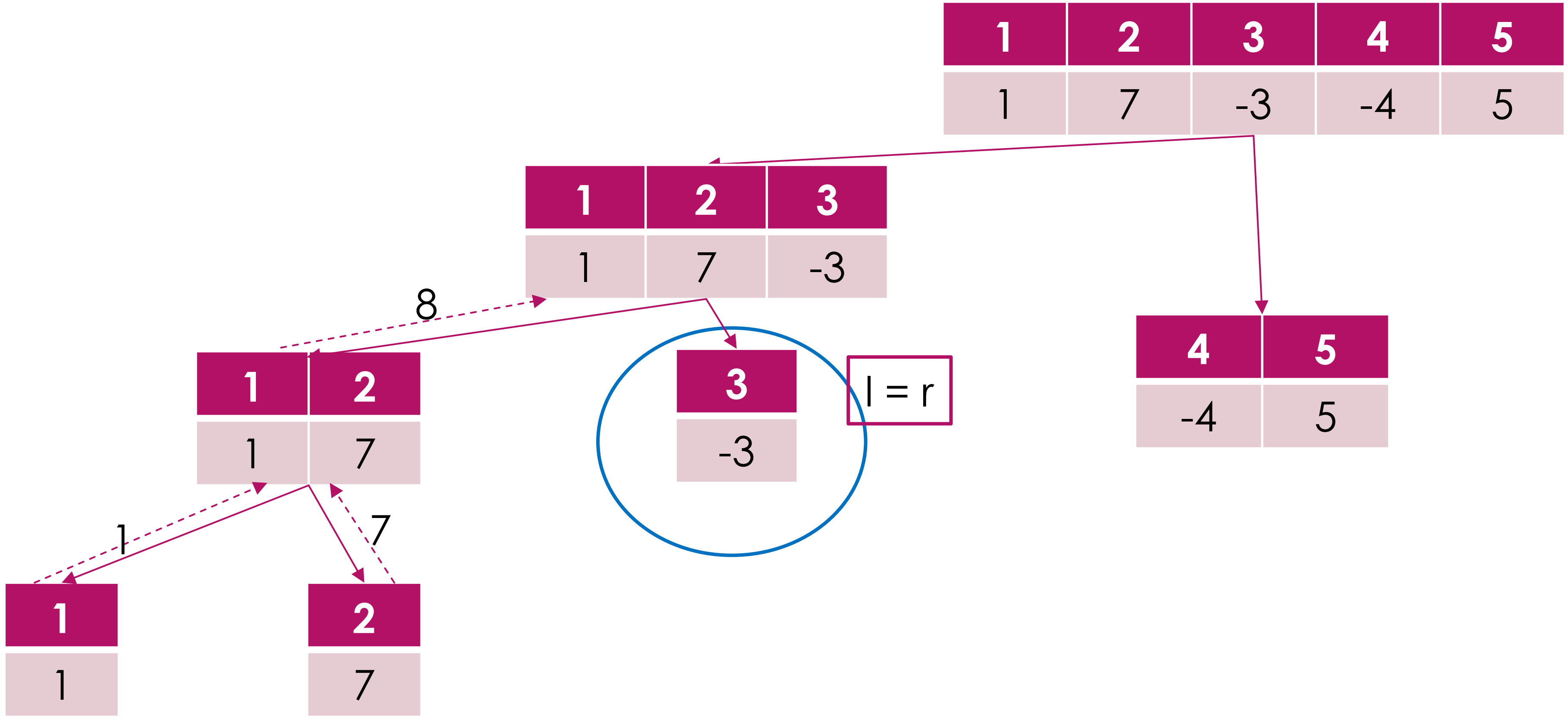
try to update the answers

max_cross_mid = 8
left_result = 1
rignt_result =7
best = 8
l = 1
r = 2
m = 1



Query	ql	qr	answer
3 4	3	4	-3
2 5	2	5	7
1 5	1	5	8

ql=l=3, update



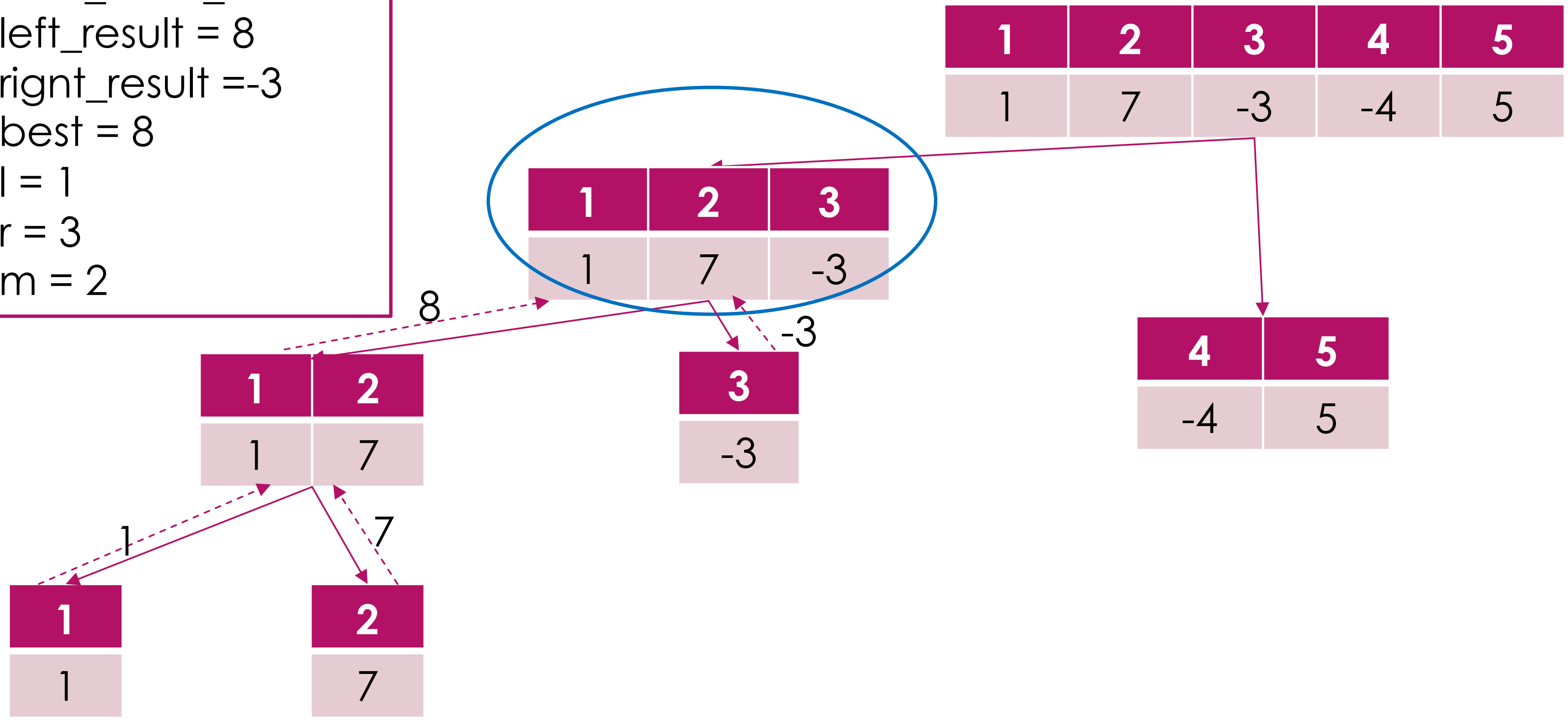
Query	ql	qr	answer
3 4	3	4	-3
2 5	2	5	7
1 5	1	5	8

ql>mid and qr > r no update

ql≤mid and qr > r compare rignt_result and max_cross_mid [2,3]

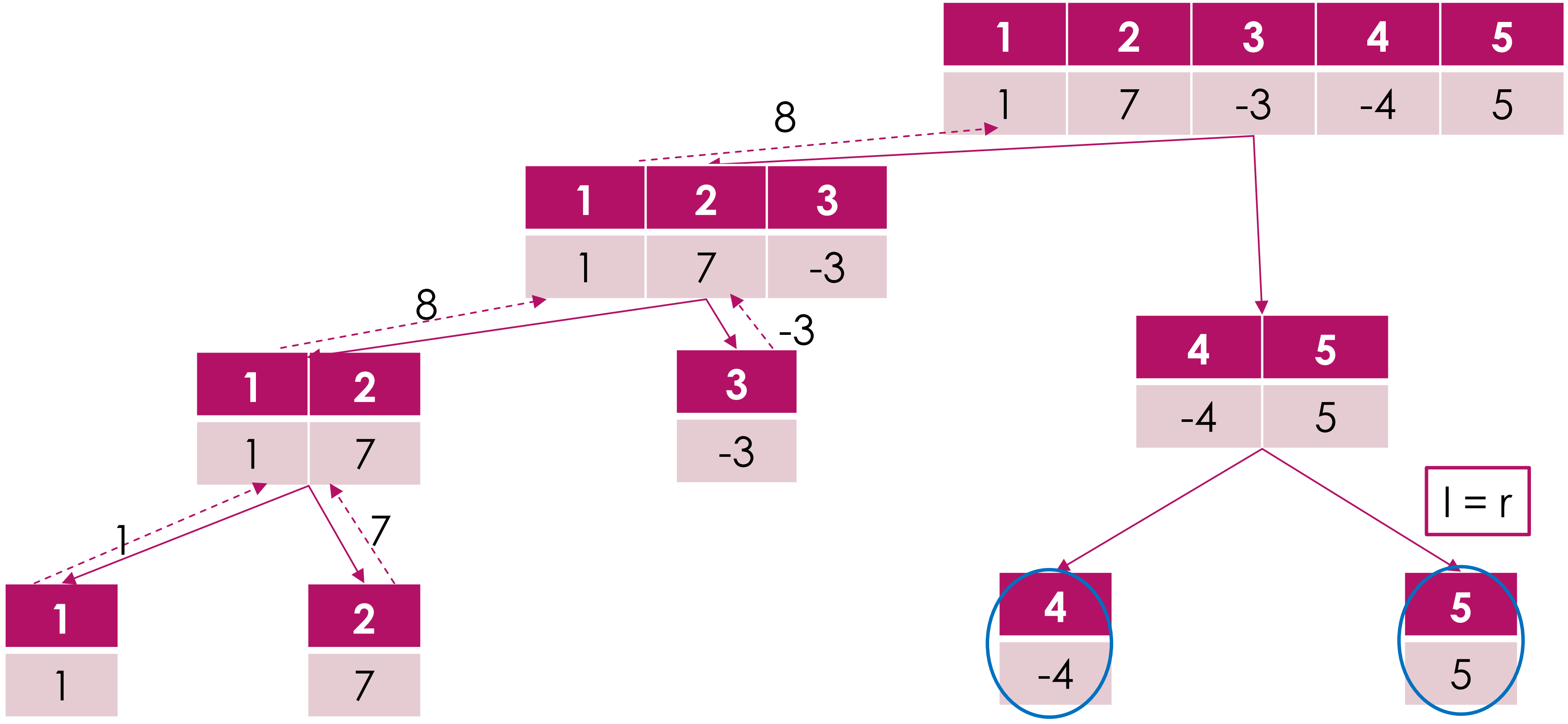
$[1,3] \in [1,5]$ $best = 8 = 8$ no update

max_cross_mid = 5
left_result = 8
rignt_result =-3
best = 8
l = 1
r = 3
m = 2



Query	ql	qr	answer
3 4	3	4	-3
2 5	2	5	7
1 5	1	5	8

no update



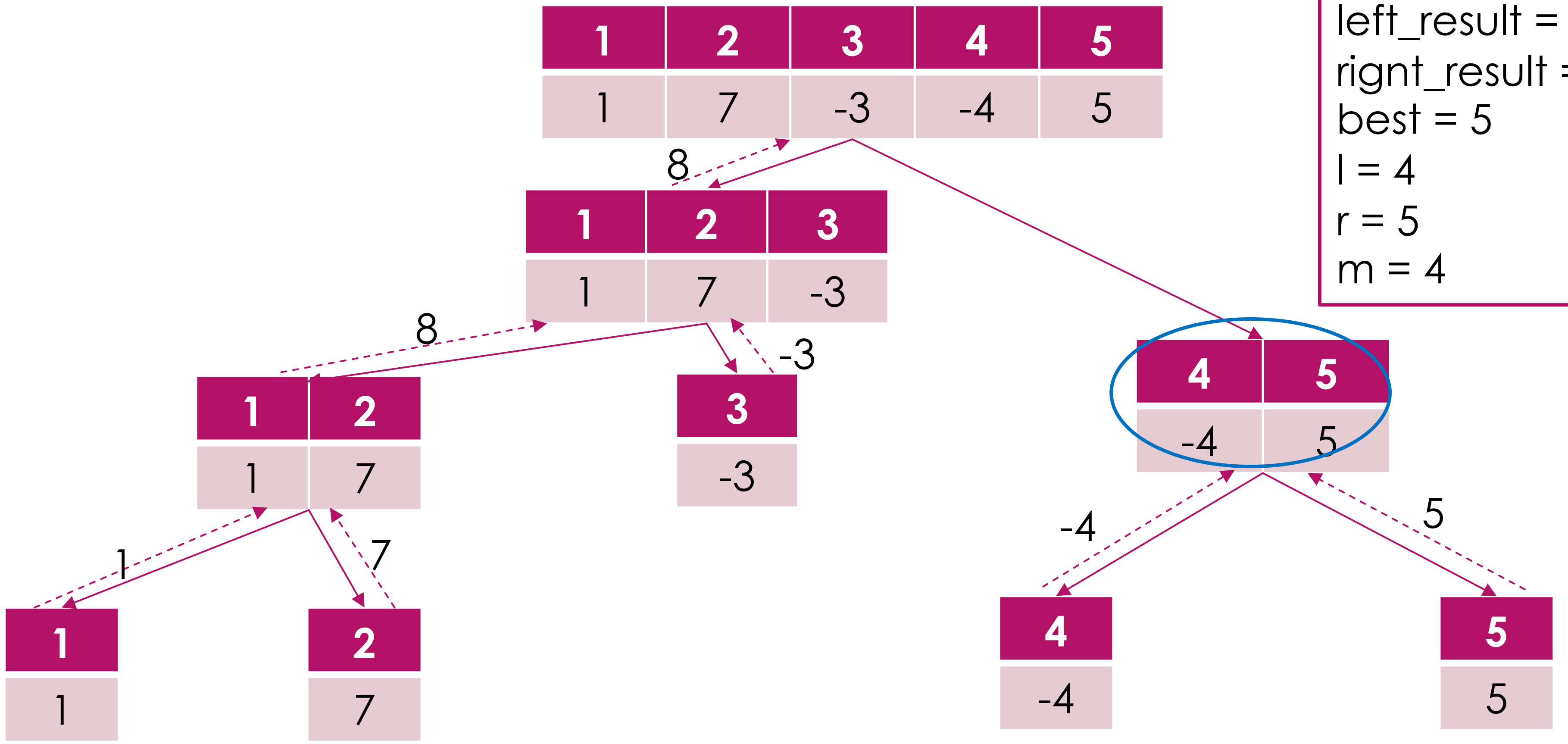
Query	ql	qr	answer
3 4	3	4	-3
2 5	2	5	7
1 5	1	5	8

qr<mid no update

$[4,5] \in [2,5]$ $best = 5 < 7$ no update

$[4,5] \in [1,5]$ $best = 5 < 8$ no update

max_cross_mid = 1
left_result = -4
rignt_result =5
best = 5
l = 4
r = 5
m = 4



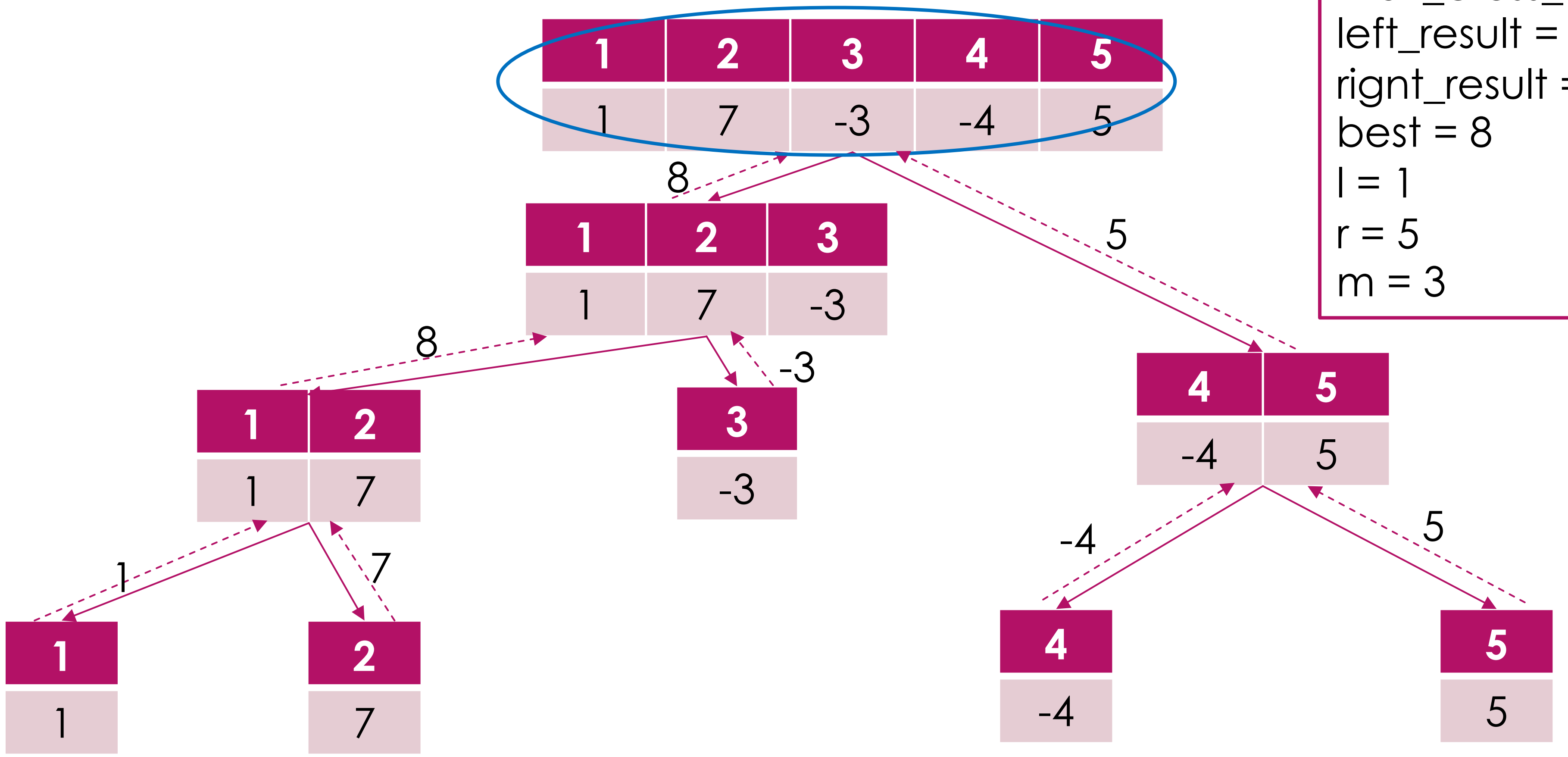
Query	ql	qr	answer
3 4	3	4	-3
2 5	2	5	7
1 5	1	5	8

l<ql ≤ mid <qr <r update with calc max_cross_mid(ql,qr)

l<ql ≤ mid <qr <r update with calc max_cross_mid(ql,qr)

$[1,5] = [1,5]$ $best = 8 = 8$ no update

max_cross_mid = 6
left_result = 8
rignt_result =5
best = 8
l = 1
r = 5
m = 3



1	2	3	4	5
1	7	-3	-4	5

$$\left.\begin{array}{ll} \text{max\_cross\_mid(1,5)} & O(n) \\ \text{max\_cross\_mid(2,5)} & O(n) \\ \text{max\_cross\_mid(3,4)} & O(n) \end{array}\right\} O(qn)$$

1	2	3	4	5
1	7	-3	-4	5

S:

0	1	2	3	4	5
0	1	8	5	1	6

min:

0	1	2	3	4	5
0	1	8	5	1	6

min(0~2) min(1~2) min(2~2)

max:

0	1	2	3	4	5
0	1	8	5	1	6

max(4~4) max(4~5)

O(n)

min:

0	1	2	3	4	5
0	1	8	5	1	6

min(0~2) min(1~2) min(2~2)

max:

0	1	2	3	4	5
0	1	8	5	1	6

max(4~4) max(4~5)

max_cross_mid(1,5) = max[5] – min[0] = 6
max_cross_mid(2,5) = max[5] – min[1] = 5
max_cross_mid(3,4) = max[4] – min[2] = -7

O(1)

Query	ql	qr	answer
3 4	3	4	-3
2 5	2	5	7
1 5	1	5	8

final answer **Total: O(q+n)logn**

l<ql ≤ mid <qr <r update with max_cross_mid(3,4) = -7 no update

l<ql ≤ mid <qr <r update with max_cross_mid(2,5) = 5 no update

l≤ql ≤ mid <qr ≤r update with max_cross_mid(1,5) = 8 no update

query [1, n] no need O(nlogn):

Kadane's Algorithm:

```
Initialize:
    max_so_far = INT_MIN
    max_ending_here = 0


Loop for each element of the array
  (a) max_ending_here = max_ending_here + a[i]
  (b) if(max_so_far < max_ending_here)
            max_so_far = max_ending_here
  (c) if(max_ending_here < 0)
            max_ending_here = 0
return max_so_far
```

O(n)

$query\ Q\ times: O(QN)$ $1 \leq N, Q \leq 2\times10^5$

https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/