Leetcode, $\cdots$

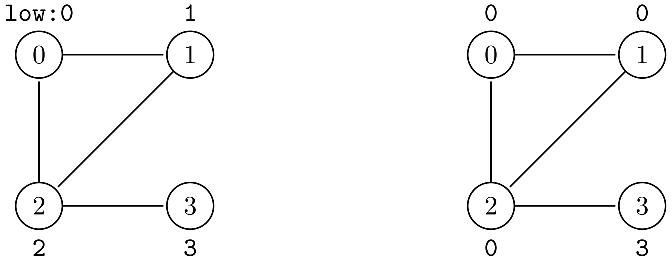
Critical Connections in a Network

id: 1192 tags: graph, dfs

	`ids[node]`	keep tracking ids of nodes in dfs ordering
Par	`low[node]`	smallest id which current node can reach
	`graph[node]`	adjacency list

Alg

1. build graph in form of adjacency list, `graph[node]`
2. tranverse graph dfsly. If neighbor node is not visited, dfs next node, update `low[node]` by `min(low[node], low[neighbor])` by callback.
3. Check if `ids[node] < low[neighbor]` is true, then we find one critical connection.
4. If neighbor node is visited and it is not the node visited right before current node, update `low[node]` by the same as in 2.



You can see that `ids[2] < low[3]`.

Max Sum of Rectangle No Larger Than K

id: 363
tags: binary search

$$\textbf{Par}\begin{cases}\texttt{mat} & \text{the original matrix}\\ \texttt{arr} & \text{array of sums from column } i \text{ to } j \text{ of } \texttt{mat}\\ \texttt{sums} & \text{ordered set of sums from begin to } \texttt{arr[i]}\\ \texttt{cur} & \text{current sum of } \texttt{arr[0}\cdots\texttt{i]}\\ \texttt{k} & \text{threshold for the result}\end{cases}$$

Alg

1. Sub problem: Max sum of subarray of array *arr*
   Init `sums` with 0, *res* with $-\infty$.
   Iterate `arr`: at `i` iteration, `cur += arr[i]`.
   Find the index of lower bound *lb* of `cur-k` in `sums`.
   If *lb* is not the end of `sums`, we have $res = \max(res,$ `cur - sums[lb]`$)$.

2. Then we use

3. todo

Find Peak Element

id: 363
tags: binary search

Par	`nums`	array, where `nums[i] != nums[j]` if `i != j`
	`l`	left pos
	`r`	right pos
	`m`	middle = (left+right) / 2

Alg

1. init left, right `l = 0, r = len(nums)-1`

2.
   ```
   while l < r:
       m = (l+r) // 2
       if (nums[m] > nums[m+1]:  r = m
       else:  l = m+1
   ```

3. the final `l` is the answer.

Sample

id: 363
tags: binary search

$$\textbf{Par}\begin{cases}\texttt{a} & \text{xxx}\\ \texttt{b} & \text{xxx}\\ \texttt{c} & \text{xxx}\end{cases}$$

Alg

1. todo
2. todo
3. todo