ICPC Training @ UNITN

Day 2 · 2019-03-20

Soluzioni del contest

Problem A - Hello World

Problem B - Cold-Puter Science

Problem C - Shattered Cake

- Calculate area of the cake; divide by width

Problem D - Dice Cup

- Fill a frequency vector with all possible outcomes
- Return the index(es) with highest frequency

- Smarter solutions are welcome but not necessary, given the problem's constraints

Problem E - Candle Box

- Caveat: on Rita's xth birthday, she adds x candles to her candlebox!
- Simulation is ok:
  - When Theo's age = 3, Theo's candles = 3, Rita's candles = (3+D)(3+D+1)/2 -6
  - Add candles until Theo's candles + Rita's candles = R+T
- Even better: solve a system of equation

$$\begin{cases} C_R = \sum_{i=4}^{a_r} i = \frac{a_r(a_r-1)}{2} - 10 \\ C_T = \sum_{i=4}^{a_t} i = \frac{a_t(a_t-1)}{2} - 10 \\ a_r - a_t = D \\ C_R + C_T = R + T \end{cases}$$

Problem F

- Simulation!
- Caveat: number grows very fast. An int may contain 9 decimal ciphers most
- Hint: $n < a$, $n \mid a \rightarrow n \mid a \pmod{n}$
- Proof: $n \mid a \rightarrow n \mid r + kn \rightarrow n \mid a$ iff $r = 0 \leftrightarrow a \pmod{n} = 0$, $a \pmod{n} < n \leftrightarrow n \mid a \pmod{n}$

Problem G

- Greedy: local optimum is global optimum
- i.e., if I have to assign 1 extra ballot box, I will assign it to the city with the most populous districts
- Priority queue

Graphs

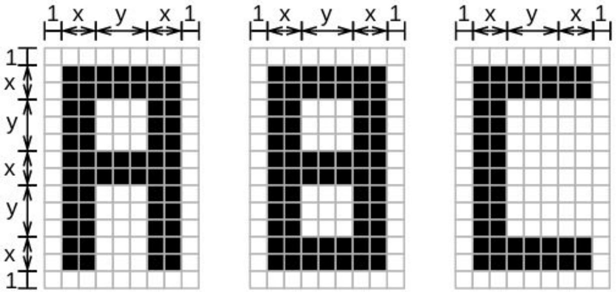
Flood Fill CC · Image Detection

- Connected Components (CC) can be found (and colored) using a DFS/BFS.
- Some problems may ask to recognize shapes, figures, etc. in a picture given a bitmap.
- Solution: each pixel is a vertex connected to adjacent pixels

Example: “Mason’s Mark” SWERC 2018

[...] He makes a black and white photo and observes :

- The picture shows stones, and every stone contains exactly one mark.
- All marks have one of the following shape with x and y being arbitrary strictly positive integers,and possibly different for each mark. Note that marks are surrounded by white pixels, and that marks cannot be rotated.



- The picture contains some noise, which are black pixels surrounded by 8 white pixels
- There are 3 kinds of black pixels, corresponding respectively to the noise, the mason’s marks,and the region around the stones
- Every white pixel belongs to the surface of a stone and some of them also belong to the interior of a mark.
- The white pixels belonging to the surface of the same stone but not belonging to the interior of the mark are all connected with respect to vertical and horizontal adjacency.
- The black pixels of the region around the stones are connected with respect to vertical, horizontal,diagonal, and anti-diagonal adjacency. All pixels of the border of the picture are black and belong to this region

```
##########################
##........######......#..#
#...###....#####..#......#
#...#.#....####.........##
#...###.....##....#####..#
#...#.#.....#.....#####..#
#...###.....#.....##.##..#
#........#..#.#...#####..#
#..###......#.....#####..#
#..#........#...#.##.##..#
#..#........#.....##.##..#
#..#...#.#..#...#.##.##..#
#..###......#............#
###....#....##....##.....#
##########################
```

1. Preprocess image, clear noise
2. Flood color the external border
3. Flood color each internal region
4. Flood color each letter
5. For each letter, flood color the internal pixels and count the number of CCs: 1 = A, 2 = B, 0 = C

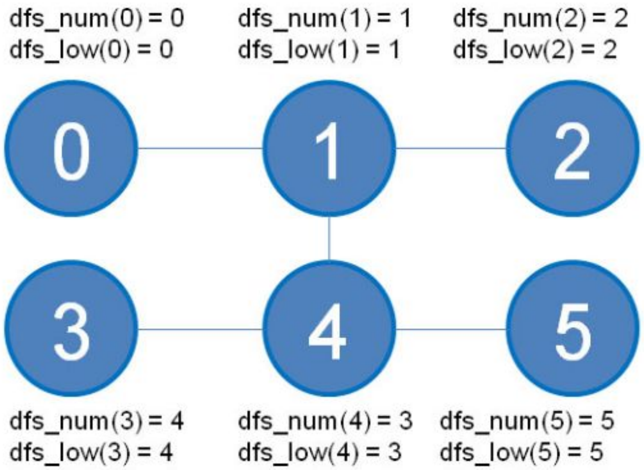
Bipartite check

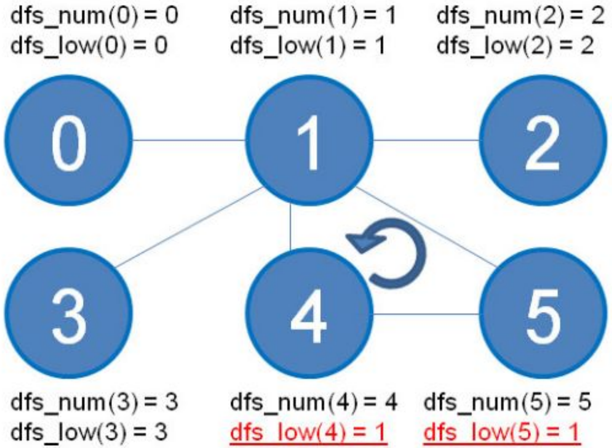
- Use graph coloring techniques to assert whether a graph is bipartite or not
- Remember that a tree is always bipartite (the opposite does not hold)
- More on bipartite graphs in the following lectures...

Cycles in graphs

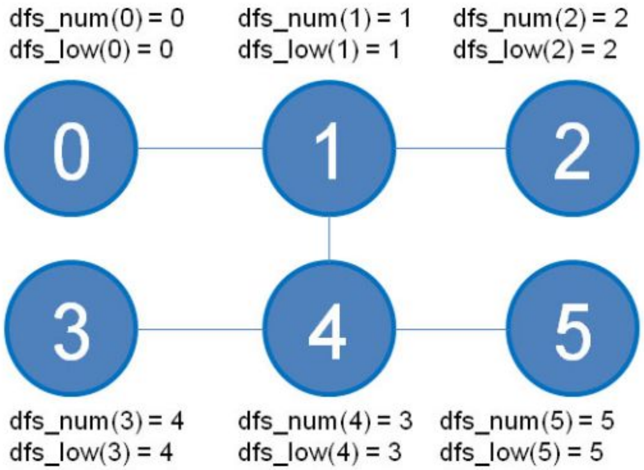
- Thou shalt not try to enumerate all possible cycles in a graph! V!
- But
  - in a directed graph, we may search for Strongly Connected Components
  - in an undirected graph, articulation points / bridges
- Hopcroft - Tarjan: Articulation points, Bridges, SCC
- Kosaraju: SCC
- Both run in $O(V + E)$
- Both are just a tweak of the DFS

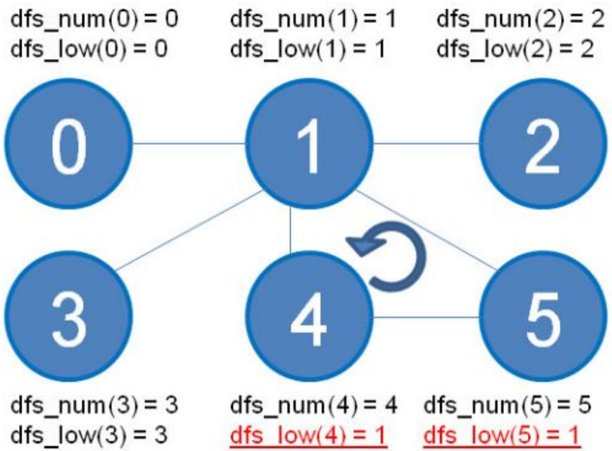
Tarjan





1. Run the DFS on the graph (V, E).
2. For each u in V, set dfs_num(u) the iteration counter when u is visited for the first time
3. For each u in V, set dfs_low(u) the minimum dfs_num(v) such that exists (u, v) in E and (u, v) is a back edge (not part of the DFS recursive subtree)



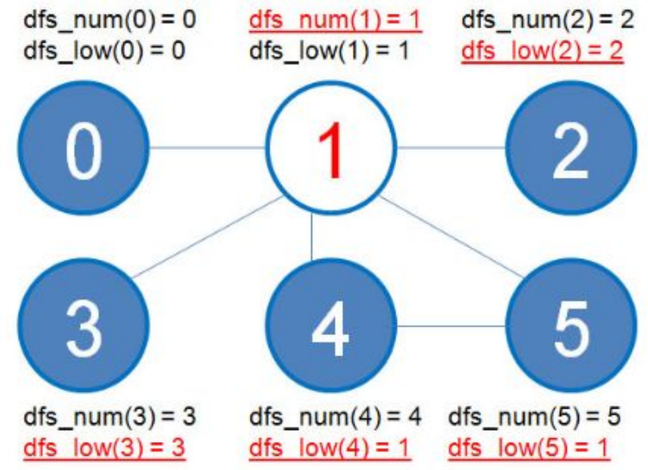


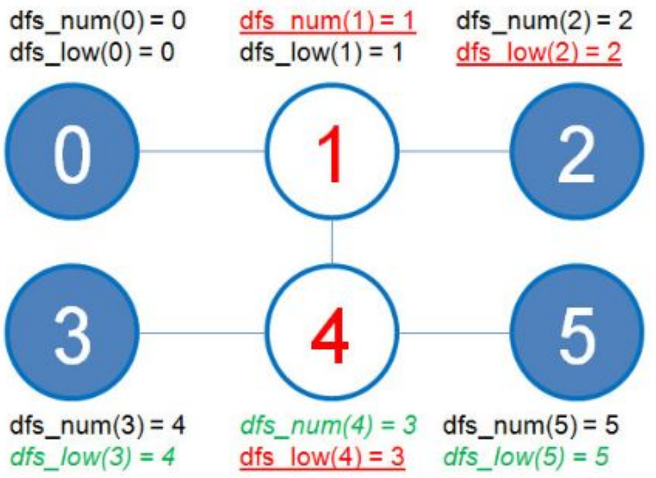
Note: if dfs_num = dfs_low there are no back edges

(vertex has been visited only once)

An edge (u, v) is a BRIDGE if

dfs_low(v) > dfs_num(u)

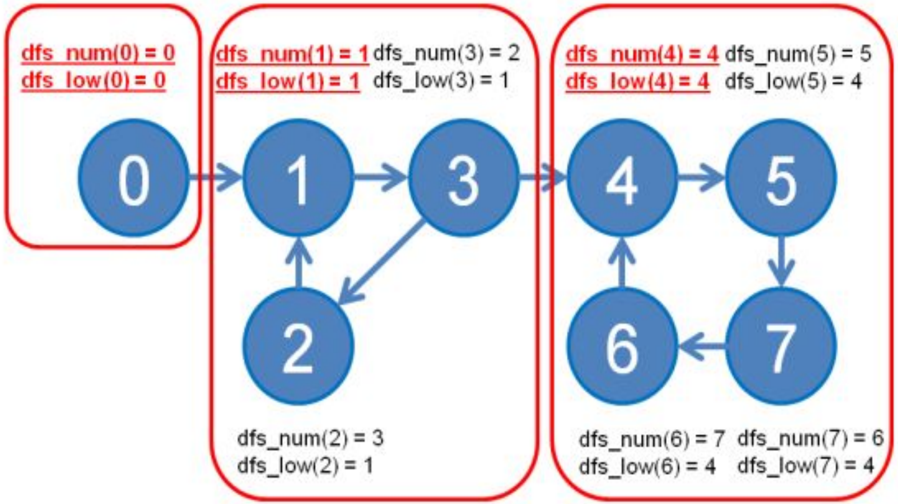




A vertex u is an ARTICULATION POINT if for any neighbour v

dfs_low(v) >= dfs_num(u)

Special case: the root of the DFS is an articulation point if the property holds AND it has more than one neighbour

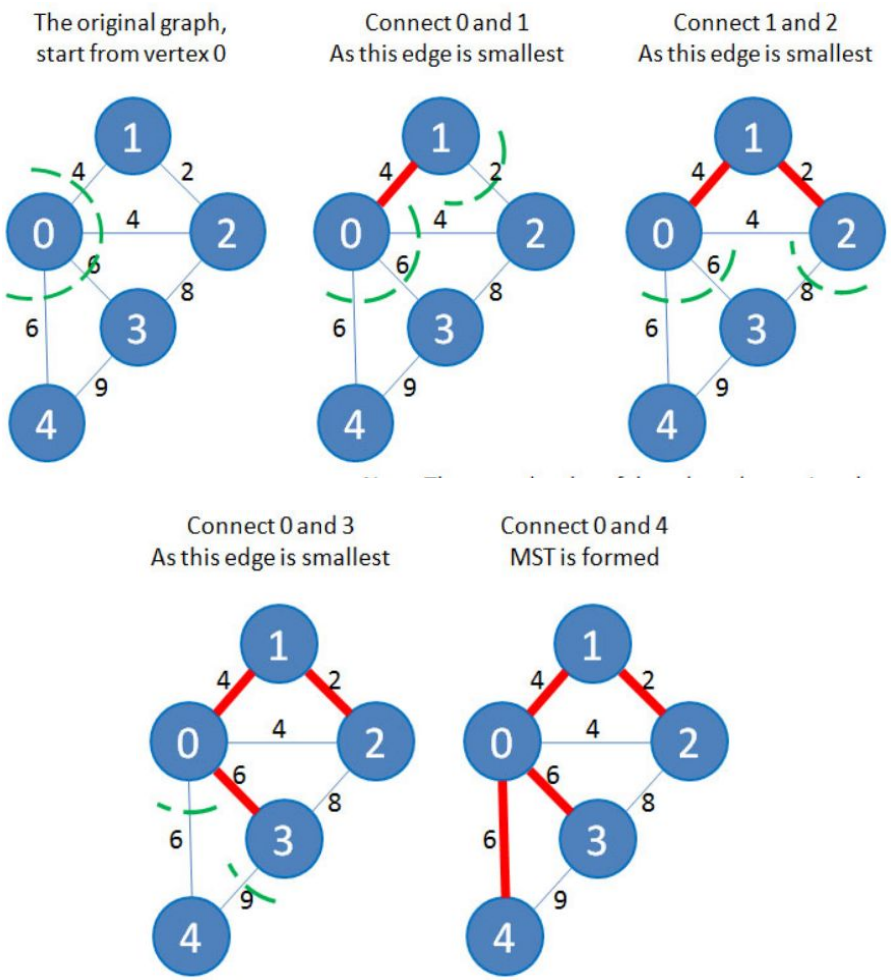


Tweak Tarjan to get SCCs

1. When we first visit a vertex, we push it into a stack
2. We update dfs_low only of vertices already visited
3. After the recursive call, if dfs_num(u) == dfs_low(u) we are in the root of a SCC. Pop the stack until we pop u.

Minimum Spanning Tree

- Given a weighted graph (V, E), w:V ➔ **R+**, find a subset E' of E such that (V, E') is a tree and sum forall w(e'), e' in E' is minimal.
- This problem can be solved using greedy techniques.
- Hint: let T a subtree of the MST, we want to augment it connecting a new vertex. We'll choose the vertex v which is connected to u in V(T) such that w((u,v)) is the minimum for every possible choice of u and v.
- Kruskal's algorithm, Prim's algorithm
- We'll go with Prim's, as Kruskal asks for a data structure you may not have seen yet (Union-Find Disjoint Set)



1. Select a root, add all edges in a prioq sorted by increasing weight
2. Pop from the queue, add edge to the MST if it connects a vertex which has not been connected yet