1 Feb 2019

Announcements

[1] Problem Set 2 is out. 3 questions, due Thurs 11:59 pm.
[2] Homework partner finding
- Use Google sheet. (See Piazza post for instrux.)
- Or, attend CS homework partner finding event
  Monday, Feb. 4, 6–7pm
  Gates 3rd floor lounge.

Today's lecture: The Minimum Spanning Tree Problem

Given an undirected graph $G=(V,E)$ with non-negative edge costs $c(e)$ for all edges in E.
(I'll also use $c(u,v)$ for $c(e)$ when $e=(u,v)$.)

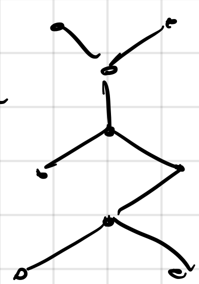
Input is the "adjacency list" representation of G.
A (doubly) linked list of vertices.
For each vertex, a (doubly) linked list of the edges it belongs to.
For each edge, a pointer to its two endpoints.

In this lecture and throughout CS 4820,
n = # of vertices
m = # of edges

Goal: Output a connected subgraph joining together <u>all</u> vertices with minimum total edge cost.

Note: Among the min-cost connected spanning subgraphs, at least one of them is a tree.

(Any conn. spanning subgraph can be transformed to a tree by repeatedly finding a cycle and removing one of its edges.)



Boruvka's Algorithm.

$E_T \leftarrow \emptyset$

while E_T does not connect all vertices together

{

one Boruvka phase

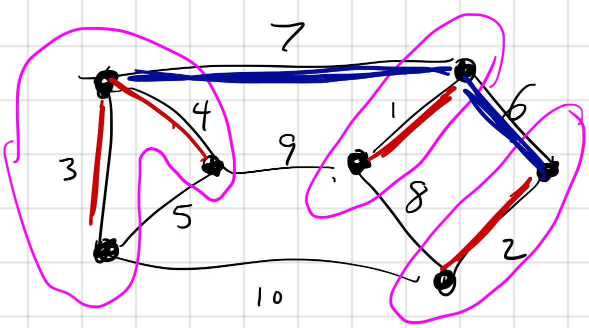
for each connected component of T select the min-cost edge joining this component to another one.

let $S = \{$edges selected in the line above$\}$

$E_T \leftarrow E_T \cup S$

}

output $T = (V, E_T)$



In this case 2 Boruvka phases suffice to connect the entire graph together.

Running time of depth-first search?

(a) $O(\log n)$

(b) $O(m \log n)$

(c) $O(m+n)$ ←

Maximum # of Boruvka phases?

(a) $O(1)$

(b) $O(\log n)$ ←

(c) $O(n)$

(d) $O(m)$

Boruvka's Alg:

$O(\log n)$ phases

$O(m)$ running time for each.

$\therefore$ $O(m \log n)$ total time.

Kruskal and Prim's Algorithms

Kruskal:

Sort edges of G by increasing cost: $e_1, e_2, \ldots, e_m$.
$E_T \leftarrow \emptyset$
for $i = 1$ to m:
  insert e_i into E_T if it doesn't
    create a cycle
endfor
output $T = (V, E_T)$

Prim:

choose a root node, r.
$T = (\{r\}, \emptyset)$

while $V(T) \neq V(G)$
  find min-cost edge from $V(T)$ to $V(G) \setminus V(T)$
  call this edge $e = (u,v)$ where $u \in V(T)$,
    $v \notin V(T)$.
  $V(T) \leftarrow V(T) \cup \{v\}$
  $E(T) \leftarrow E(T) \cup \{e\}$
endwhile
output T