CMPT307: SCC & MST

Week 11-2

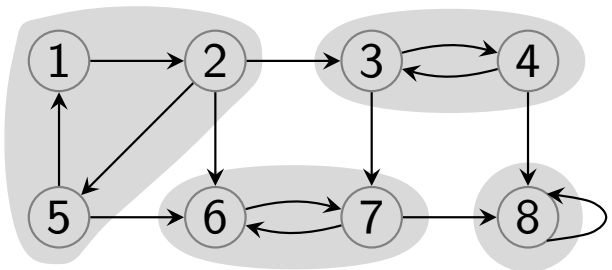
Xian Qiu

Simon Fraser University

xianq@sfu.ca

Strongly Connected Components

- given directed graph $G = (V, E)$ and $C \in V$
- C is a strongly connected component (SCC) if C is a maximal set s.t. any pair of vertices are reachable from each other



- many graph algorithms do the following:
  - decompose G into strongly connected components
  - run the algorithm separately for each component
  - combine the solutions

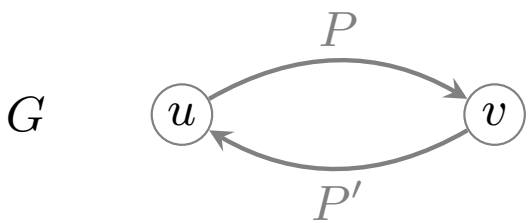
Transpose Graph

given directed G, the transpose G^T of G is the directed graph by reversing all edges of G

Proposition

G^T and G have the same strongly connected components

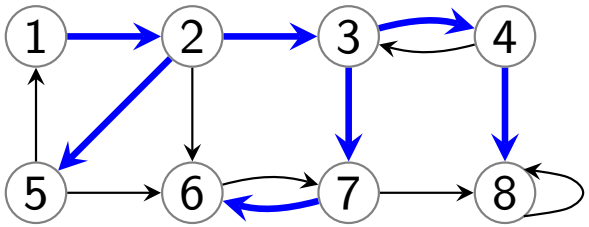
- ▷ assume $\exists$ u-v path in G, then consider G^T



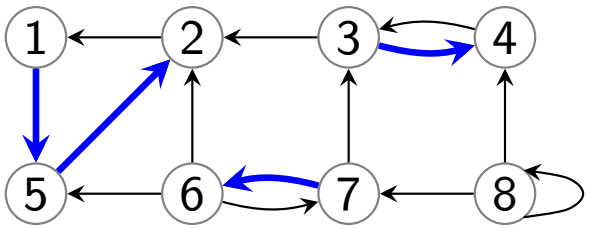
- ▷ if an u-v path exists in G^T, then $\exists$ v-u path in G
- ▷ the cycle formed by P and P' is an SCC

Algorithm

- run DFS: get a forest, then consider G^T
- run DFS again



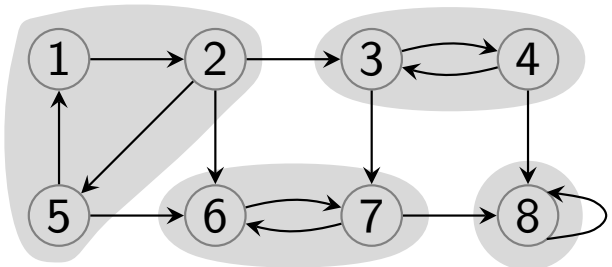
increasing order w.r.t. $v.f$: 8 4 6 7 3 5 2 1



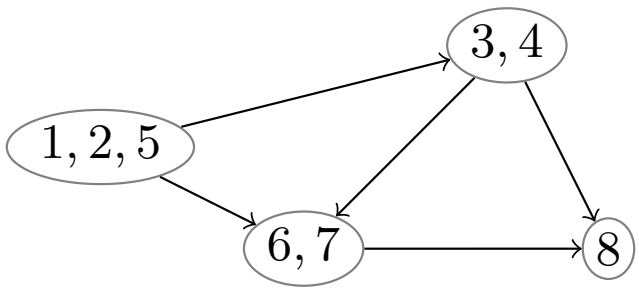
run DFS for G^T w.r.t. $v.f$ in decreasing order

Component Digraph

SCC of digraph G



component graph G^{SCC}



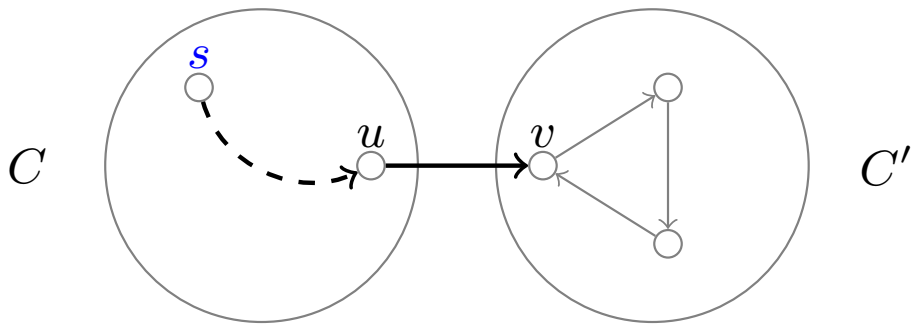
Lemma. The component digraph of is acyclic.

Component Finishing Lemma

Lemma

Let C and C' be distinct SCC of digraph G. If there is an edge $(u, v) \in G$ with $u \in C$ and $v \in C'$, then $f(C) > f(C')$.

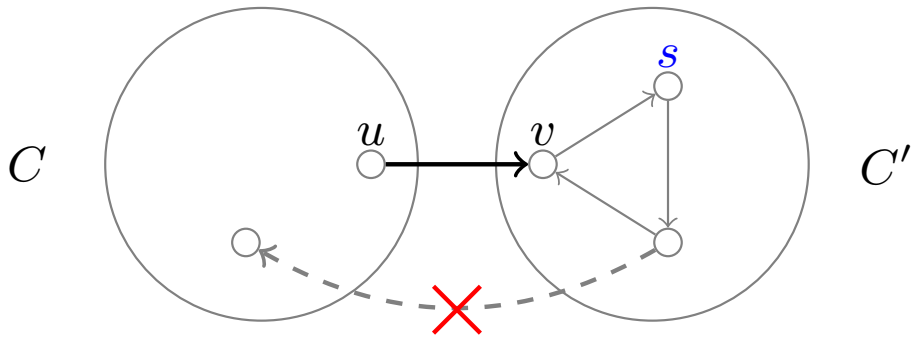
- define $d(C) = \min_{u \in C} \{u.\text{d}\}$ and $f(C) = \max_{u \in C} \{u.\text{f}\}$
- case 1: $d(C) < d(C')$ and let $s \in C$ be firstly discovered



$\exists$ white path from s to any $v \in C'$, hence any $v \in C'$ is a descendant of s

Component Finishing Lemma

- case 2: $d(C) > d(C')$ and let $s \in C'$ be firstly discovered



no edge from C' to C, hence C' finishes before C

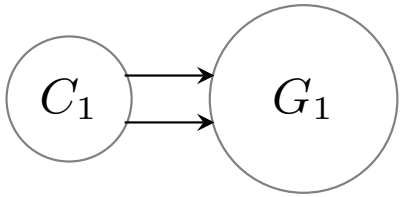
Corollary

Let C and C' be distinct SCC of digraph G. If there is an edge $(u, v) \in G^T$, with $u \in C$ and $v \in C'$, then $f(C) < f(C')$.

- $f(C) > f(C')$, then no (u, v) in G^T

Correctness of the Algorithm

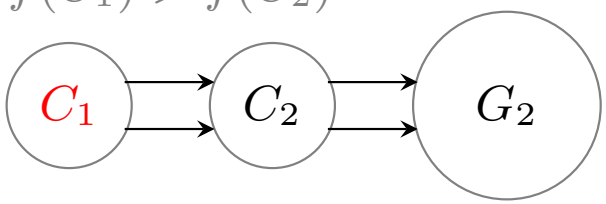
start with u of maximum u.f, and get C_1 which is an SCC



$G_1 = G_1 - C_1$

then proceeds to u with maximum u.f in $G - C_1$, and get SCC C_2

$f(C_1) > f(C_2)$



$G_2 = G - G_1 - G_2$

already been searched

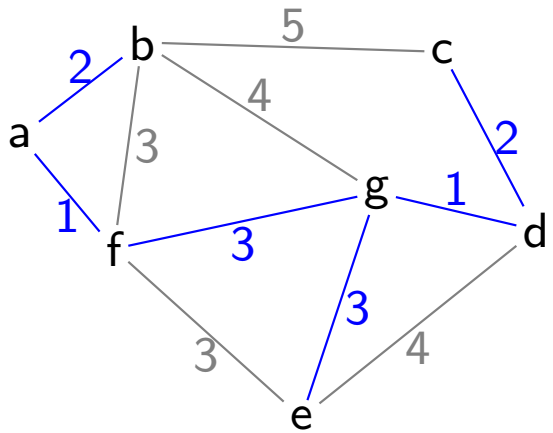
by corollary, no edges from C_1 to C_2 in G^T, so C_1 cannot be extended (correctness by induction)

Minimum Spanning Trees

The MST problem

- given undirected $G = (V, E)$ and edge costs $c : E \to \mathbb{R}$
- find a spanning tree T of G of minimum total cost

- T is a spanning tree of G if T is a tree with $V(T) = V$
- total cost: $c(T) = \sum_{e \in T} c(e)$



techniques involved in the MST problem

- greedy algorithm
- data structures: disjoint sets, min-priority queue
- amortized analysis

Coloring Process

Edge-coloring process

- initially, all edges are uncolored
- for all uncolored edges, we color it either blue (accepted) or red (rejected)
- blue edges form MST

Color invariant

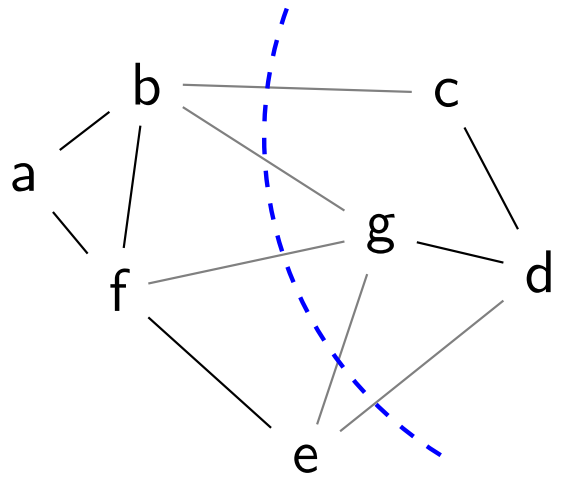
There is an MST containing all the blue edges and non of the red edges.

- the set of blue edges can be extended to an MST
- the final set of blue edges is an MST

Cut

- a cut of $G = (V, E)$ is a partition of V into two sets:

$$(X, \bar{X}), \text{ where } \bar{X} = V \backslash X$$



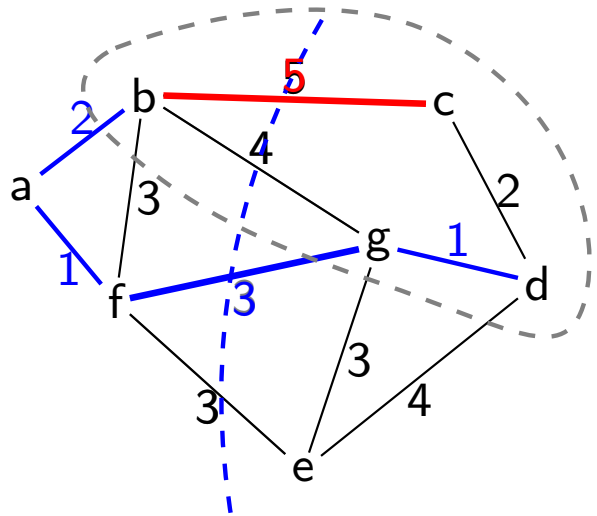
- $e = (u, v)$ crosses $(X, \bar{X})$ if its endpoints are in different parts of the cut, *i.e.*, $u \in X$ and $v \in \bar{X}$
- $\delta(X)$: set of edges that cross $(X, \bar{X})$, *i.e.*,

$$\delta(X) = \{(u, v) \in E \mid u \in X, v \in V \backslash X\}$$

Coloring Rules

blue rule: best-in

- ▷ select a cut $(X, \bar{X})$ that is not crossed by any blue edge
- ▷ color the min cost edge blue among uncolored edges in $\delta(X)$



red rule: worst-out

- ▷ select a simple cycle C that does not contain any red edge
- ▷ color the max cost edge red among uncolored edges in C

Greedy Algorithm

Generic-MST

apply the two coloring rules arbitrarily until all edges are colored;

Theorem

The above algorithm maintains the color invariant in each step and eventually colors all edges.

Proof of the Theorem

by induction: assume true for $(k-1)$-th iteration and consider k:

blue rule

- ▷ assume $e \in \delta(X)$ is colored blue and $e \notin$ MST
- ▷ there exists $e' \in \delta(X)$, $e' \in$ MST, with $c(e') \geq c(e)$
- ▷ replace e' with e in MST (feasible?)

red rule

- ▷ assume $e \in C$ is colored red and $e \in$ MST
- ▷ there exists $e' \in C$, $e' \notin$ MST, with $c(e') \leq c(e)$
- ▷ replace e' with e in MST (feasible?)

hence, color invariant holds

- ▷ at termination, are all edges colored? yes

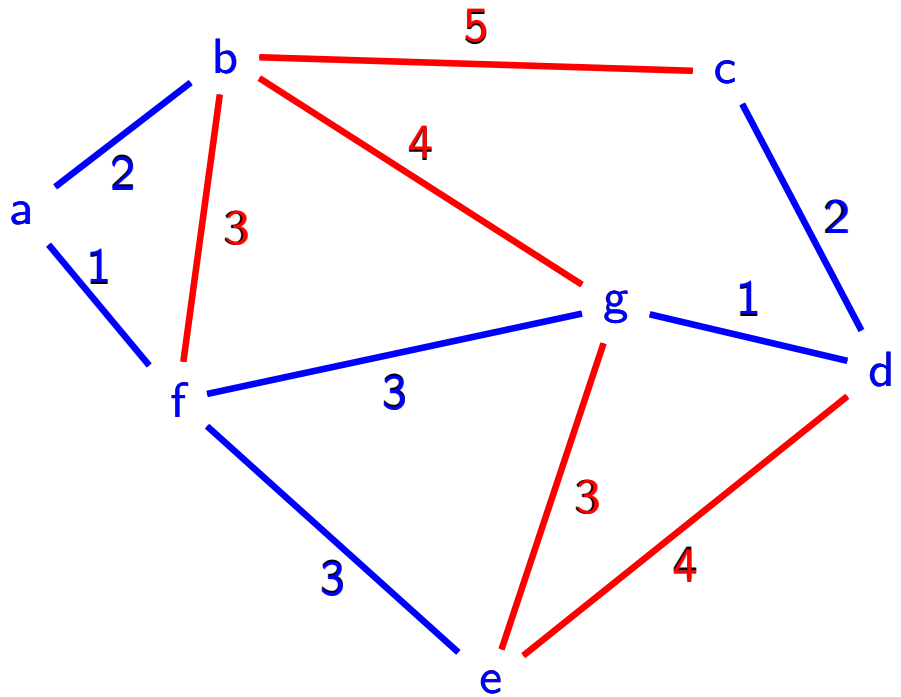
Kruskal's Algorithm

MST-Kruskal(G, c)

```
   // initialization and preprocessing
1  A = ∅;
2  for v ∈ G.V do
3      MAKE-SET(v)
4  sort edges of G into non-decreasing order w.r.t. edge costs;
   // greedy
5  for each edge (u, v) from the sorted list do
6      if FIND-SET(u) ≠ FIND-SET(v) then
7          A = A ∪ {(u, v)}; // add (u, v) to A if A ∪ {(u, v)} is acyclic
8          UNION(u, v);
9  return A;
```

Example

$e_1 = (a, f), c(e_1) = 1$
$e_2 = (d, g), c(e_2) = 1$
$e_3 = (a, b), c(e_3) = 2$
$e_4 = (c, d), c(e_4) = 2$
$e_5 = (b, f), c(e_5) = 3$
$e_6 = (f, g), c(e_6) = 3$
$e_7 = (e, f), c(e_7) = 3$
$e_8 = (e, g), c(e_8) = 3$
$e_9 = (b, g), c(e_9) = 4$
$e_{10} = (d, e), c(e_{10}) = 4$
$e_{11} = (b, c), c(e_{11}) = 5$



Running Time

- sorting – $O(m \log m) = O(m \log n)$
- $m + n$ times of disjoint set operations, in which n operations are $\textsc{Make-Set}$
- using union by rank and path compression for disjoint set operations, the running time is $O((m + n)\alpha(n)) = O(m\alpha(n))$
- as $\alpha(n) = O(\log n)$, the total running time is $O(m \log n)$

Prim's Algorithm

MST-Prim(G, c)

```
1 let A be an empty tree;
2 choose an arbitrary node r ∈ G.V as the root of A;
3 for i = 1 to n − 1 do
4     find a min-cost edge e_i ∈ δ(V(A));
5     A = A + {e_i};
```

- use priority queue to extract min-cost edge, yielding running time $O(m \log n)$ (*cf.* textbook)
- using Fibonacci heap to implement the priority queue can improve the running time to $O(m + n \log n)$

Example