CMPSC 465
Data Structures and Algorithms
Spring 2022

Instructor: Chunhao Wang

Greedy algorithms

Greedy algorithms

Minimum Spanning Tree

Running time of Kruskal's algorithm (I)

Depends on how we implement make_set, find_set, and union

Using linked list:

$$\{a, b, c\} \quad \text{head} \to a \to b \to c \qquad \text{find\_set}(b)\text{: } O(1)$$

$$\text{make\_set}(v)\text{: } O(1)$$

$$\{d, e\} \quad \text{head} \to d \to e$$

$$\text{union}(a, b) \qquad \text{head} \to a \to b \to c \to d \to e$$

Cost of union: O(length of the shorter list)

Using an array to implement it:

$$\begin{array}{r|c|c|c|c|c|} \hline \text{vertex} & 1 & 2 & 3 & 4 & 5 \\ \hline \text{head} & 1 & 1 & 1 & 4 & 4 \\ \hline \end{array} \xrightarrow{\text{union}} \begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

Running time of Kruskal's algorithm (II)

```
1 def KRUSKAL_MST(undirected G = (V, E), weights w = (w_e)_{e∈E}):
2     Set A := { };
3     for v ∈ V:
4         make_set(v) ;                                   // O(|V|)
5     Sort E in increasing order of edge weights ;        // O(|E| log |V|)
6     for (u, v) ∈ E:
7         if find_set(u) ≠ find_set(v):
8             A := A ∪ {(u, v)};
9             union(u, v);
```

Worst-case cost for union: $O(|V|)$. What about the cost for lines 6-9?

Consider a single $v \in V$. Once it's touched in some union operation, the size of the set at least doubles. Since the maximum size of a set can be $|V|$, each v is touched at most $O(\log |V|)$ times

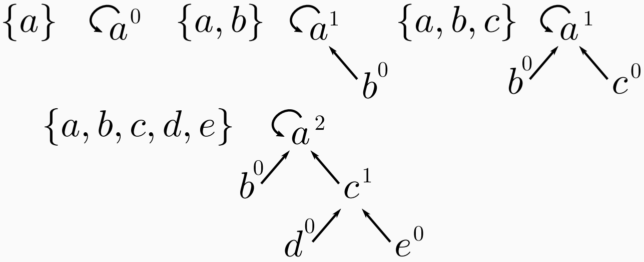
At most $|V|$ vertices are involved in union operations, so the total cost of lines 6-9: $O(|V| \log |V|)$

Total cost of the algorithm: $O(|E| \log |V|)$

Alternative data structure

The linked-list implementation is good enough, but there exist better data structures to improve the worst-case cost for union

Directed tree disjoint set:



Definition

$\pi(x)$: parent of x

root node: x s.t. $\pi(x) = x$

$\text{rank}(x)$: number of the edges in the longest simple path from x to a leaf

Operations of direct tree disjoint set (I)

- $\mathrm{make\_set}(v)$

  def $\mathrm{make\_set}(v)$**:**

  $\pi(v) := v$;

  $\mathrm{rank}(v) = 0$;

  Cost: $O(1)$

- $\mathrm{find\_set}(v)$

  def $\mathrm{find\_set}(v)$**:**

  while $v \neq \pi(v)$**:**

  $v := \pi(v)$;

  return v;

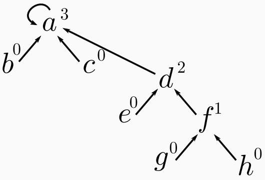
  Cost: O(depth of the node in the tree)

- what about union?

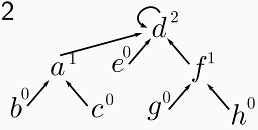
Operations of direct tree disjoint set (II)

- union:

Option 1



Option 2



better!

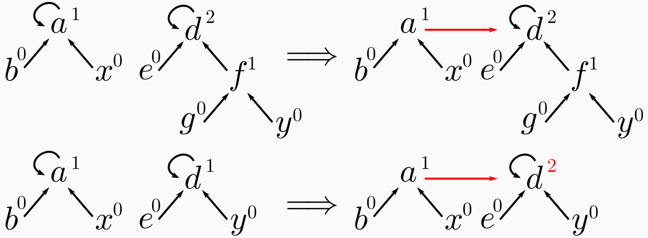
Basic idea: attach the smaller ranked tree to a larger one

Operations of direct tree disjoint set (II)

def union(x, y)**:**

- $r_x := \text{find\_set}(x)$, $r_y := \text{find\_set}(y)$;
- **if** $\text{rank}(r_x) > \text{rank}(r_y)$**:**
  - $\pi(r_y) := r_x$;
- **else:**
  - $\pi(r_x) := r_y$;
  - **if** $\text{rank}(r_x) == \text{rank}(r_y)$**:**
    - $\text{rank}(r_y) := \text{rank}(r_y) + 1$;

Cost: dominated by find_set



Cost of find_set using directed tree disjoint set

Observation

Root note with rank k is formed by the merge of two rank $k-1$ trees

Lemma

Any root node of rank k has at least 2^k nodes in it

Proof.

By induction: base case has $k = 0$ and $2^0 = 1$.

Assume the statement is true for $k-1$. By observation: after merging, the number of nodes is $\geq 2^{k-1} + 2^{k-1} = 2^k$ $\square$

By the lemma, if we have $|V|$ nodes, the maximum rank is $\log|V|$. So

- the cost of find_set: $O(\log|V|)$
- the cost of union: $O(\log|V|)$

Total running time of Kruskal using directed tree disjoint set

1 **def** KRUSKAL_MST(*undirected* $G = (V, E)$, *weights* $w = (w_e)_{e \in E}$):
2   Set $A := \{\}$;
3   **for** $v \in V$:
4   make_set(v) ;   // $O(|V|)$
5   Sort E in increasing order of edge weights ;   // $O(|E| \log |V|)$
6   **for** $(u, v) \in E$:
7   **if** $\text{find\_set}(u) \neq \text{find\_set}(v)$:
8             $A := A \cup \{(u, v)\}$;
9             $\text{union}(u, v)$;

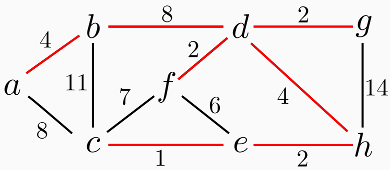
Lines 6-9: $O(|E| \log |V|)$

Total cost: $O(|E| \log |V|)$

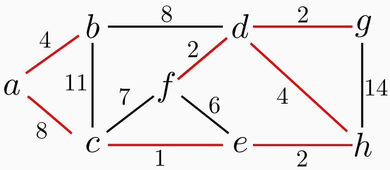
Prim's algorithm

Intuition: iteratively grows the tree

Kruskal



Prim
starting with c



Prim's algorithm: pseudocode

Let S be the set included in the tree so far

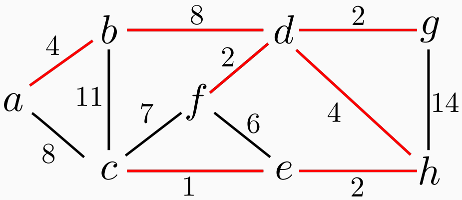
$\mathrm{cost}(v) := \min_{e=(u,v) \text{ s.t. } u \in S} w_e$ and $\mathrm{prev}(\cdot)$ is used to keep track of the tree

def $\textsc{Prim\_MST}$*(undirected $G = (V, E)$, weights $w = (w_e)_{e \in E}$)*:

- **for** $v \in V$:
  - $\mathrm{cost}(v) := \infty$;
  - $\mathrm{prev}(v) := \mathrm{nil}$;
- Pick any initial vertex u_0;
- $\mathrm{cost}(u_0) := 0$;
- $H := \mathrm{make\_queue}(V)$; // keys are $\mathrm{cost}(v)$
- **while** *H is not empty*:
  - $v = \mathrm{delete\_min}(H)$;
  - **for** $e := (v, z) \in E$:
    - **if** $\mathrm{cost}(z) > w_e$:
      - $\mathrm{cost}(z) := w_e$;
      - $\mathrm{prev}(z) := v$;

Prim's algorithm: a running example

Starting with f



Set S	a	b	c	d	e	f	g	h
$\{\}$	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	0/nil	∞/nil	∞/nil
f	∞/nil	∞/nil	$7/f$	$2/f$	$6/f$		∞/nil	∞/nil
f, d	∞/nil	$8/d$	$7/f$		$6/f$		$2/d$	$4/d$
f, d, g	∞/nil	$8/d$	$7/f$		$6/f$			$4/d$
f, d, g, h	∞/nil	$8/d$	$7/f$		$2/h$			
f, d, g, h, e	∞/nil	$8/d$	$1/e$					
f, d, g, h, e, c	$8/c$	$8/d$						
f, d, g, h, e, c, b	$4/b$							
f, d, g, h, e, c, b, a								