5 Feb 2024 Fast Implementations of MST

Announcement:

Challenge Problems are mandatory for 5820, optional (+ ungraded) for 4820.

Two algorithms (among many) for MST:

1. PRIM's.

   Start from any vertex. Build a tree by growing out from there, one edge at a time, always adding the min-weight edge from the tree to its complement.

2. KRUSKAL's

   Sort edges in increasing weight order.
   Add edges from the list in this order, omitting the ones that form a cycle.

Space complexity measured in words, not bits.

Running Times. The assumption is always

(1) if a problem has input size B bits, then one "word" of data is log(B) bits.

(2) Operations whose inputs and outputs are O(1) words are assumed to run in O(1) time.

E.g. adding two integers each of size log(B) bits takes O(1) time.

How many bits in the input to MST?

Graph has n vertices
m edges
m edge weights (integers)

$\log(n)$ bits to identify a vertex

$\log(m) \le 2\log(n)$ bits to identify an edge.

List of n vertices: $O(n \log n)$.

List of neighbors of vertex v:

if v has d_v neighbors,

$O(d_v \log n)$ bits.

$\log B \ge \log(m+n) + \log\log n$

$\ge \log(n)$

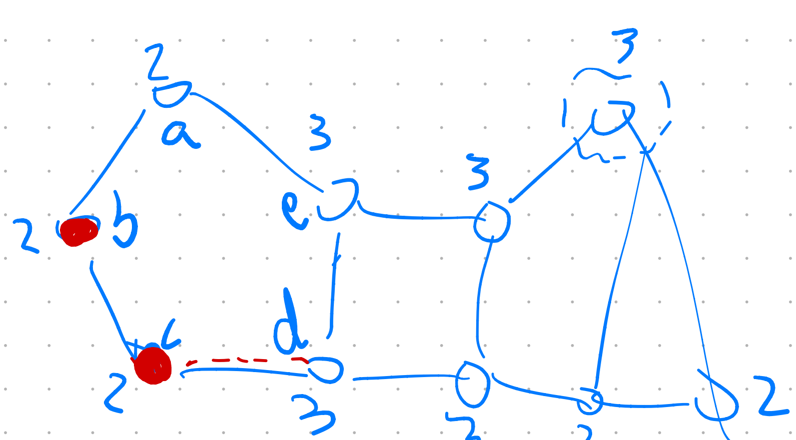
B

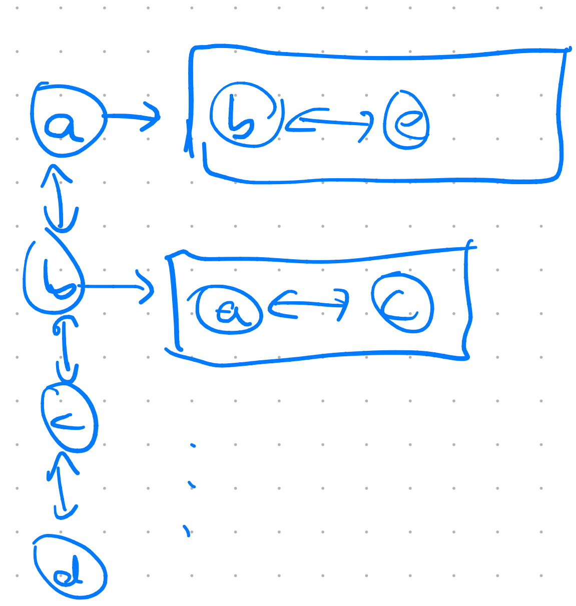
Total bits in adj list is

$$O(n \log n) + \sum_v O(d_v \log n) = O((m+n) \log n).$$

Total <u>words</u> in adj list: $O(m+n)$.

In MST problem the standard assumption is that edge weights are in the range $[0, 2^{O(\log n)}]$ so writing an edge weight in binary takes $O(\log n)$ bits, i.e. $O(1)$ words.





Implementing Prim's algorithm:

Using priority queue — a data struct that stores elements ("values") with priorities ("keys").

Operations:

- insert an element
- delete an element
- change priority of element
- extract element of min priority

"heap" $O(\log n)$

Elements of the PQ are vertices not yet in T.

Priority of element v is the min weight of an edge from T to v. (Or ∞ if no such edge has been found)

Extra data struct mapping $v \notin T$ to cheapest edge. (e.g. hash map)

n loop iterations.

Each starts with ExtractMin to find the vertex with min-weight connection to T.

Use hash map to find that min weight edge.

Update priority of all neighbors.