Exercises for Unit 24.

Problem 1.

We implement Jarnik-Prim to run in $O(m + n \log n)$ with Fibonacci heaps.

We maintain a graph X which contains the edges of the MST we have discovered so far. For each vertex v, we have an edge v_e which has min. weight among all edges connecting v to X. We keep these edges in a MinHeap.

Algorithm:

Initialize v_e = Sentinel, $\forall e \in E$ and cost(sentinel) = ∞.

Insert all v_e in Q

While Q not empty

v_e = DeleteMin(Q). ⇝ find min edge of cut $X, V \setminus X$

add v, v_e to X

(*) for all edges $(v, w) \in E$:

if $w_e \in Q$ and cost(w_e) > cost(v, w) ⇝ updates cost of edges in cut.

$w_e = (v, w)$ ⇝ DecreaseKey(w_e, cost(v, w)) ⇝ $O(1)$ with Fibonacci Heap.

Return X.

Since X has at most $n-1$ edges, we perform $O(n)$ DeleteMin operations. The size of Q is at most n, so our total cost for DeleteMin is $O(n \log n)$.

In step (*) we look at all edges at most once, so overall it will have cost $O(m)$

⇒ Cost of Prim Algo is $O(m + n \log n)$

Average Case Analysis of #DecreaseKey operations.

Consider $v \in V$. $k = \deg(v)$, let $e_1, \dots, e_k$ be the incident edges to v in the order they are scanned. Then cost(e_1), cost(e_2), ..., cost(e_k) is a random permutation of k non-negative numbers.

The edges e_i that cause DecreaseKey must have cost smaller than all the edges scanned before ⇒ cost(e_i) < cost(e_j) $\forall 1 \le j < i$.

$= E[\#\text{DecreaseKey for } v] \le E[\#\text{left-to-right minima of a sequence of length } k] \le \log_e k + 1 = H_k$

(from lecture)

$$\Rightarrow E[\#\text{DecreaseKey}] = \sum_{v \in V} E[\#\text{DecreaseKey for } v] = \sum_{v \in V} \left[\ln\left(\deg(v)\right) + 1 \right] \le n \cdot \ln\left(\sum \deg(v) \cdot \frac{1}{n}\right) + n$$

$$= n \cdot \ln\left(\frac{2m}{n}\right) + n \in O\left(n \ln\left(\frac{m}{n}\right)\right)$$

Exercises for Unit 24

2) For an undirected graph $G=(V,E)$, we are given a minimum spanning tree T^*.

i) edge e, with cost $c(e)$ is added to G.

$T^* \cup e$ has a unique cycle C. We can find cycle C using DFS in $O(n)$ time. Let e' be the edge with the largest cost in C, By cycle property it does not belong to the minimum spanning tree of $G \Rightarrow (T^* \cup e) \setminus e'$ is the spanning tree of G.

ii) edge e is removed from G.

if e does not belong to T^*, T^* will still be a minimum spanning tree of G. if $e \in T^*$, forest $T \setminus e$ has two connected components, with the set of vertices S and $V \setminus S$. Let e' be the edge with the smallest cost in G, such that if $e' = (u, v) \Rightarrow u \in S$ and $v \in V \setminus S$ (e' connects S to $V \setminus S$, thus it belongs to the cut set of $(S, V \setminus S)$) By cut property e' belongs to the minimum spanning tree of $G \Rightarrow (T^* \setminus e) \cup e'$ is a new minimum spanning tree of G. We can find the edge e' in running time $O(m)$ By considering all the edges in a cut-set and we can find two connected components of $T^* \setminus e$ in $O(n)$ time, using DFS. $\Rightarrow$ total running time is $O(m+n)$

Exercises for Unit 24

(3) **Lemma** A minimum spanning tree is a min-max spanning tree.

Proof. assume otherwise: $\exists$ edge e in the minimum spanning tree s.t. $e > \max_{e' \in T'} c(e')$, where T' is the min-max spanning tree

- then, we can remove the edge e from the minimum spanning tree T and add a new edge from T', that joins the 2 connected components that we got by removal of e into a new spanning tree T''

$$w(T'') = w(T) \underbrace{- e + e'}_{<0} < w(T)$$

e: edge we removed; e': edge we added, comes from T'

$\Rightarrow\Leftarrow$ because T is the minimum spanning tree

$\Rightarrow$ assumption was wrong, so the minimum spanning tree is also a min-max spanning tree

□

- we can achieve at least any runtime that we have for minimum spanning tree by simply running some minimum spanning tree alg.
  - → we can be better if we design an alg. specifically for this problem...