CSE 102 11-15-22

- midterm 2 Thur 11-17
- mid 2 review Problems Posted. will Post solns. tonight.
- hw 6 due tonight
- Supplemental lecture: Greedy.

Minimum Weight spanning trees (MWST)

Definitions.

- Graph $G = (V, E)$
- connected
- cycle  ......
- acyclic

- tree : connected & acyclic.
- Tree's theorem
- A subgraph T of G is called a spanning tree iff
  (1) T is a tree, and
  (2) $V(T) = V(G)$
- weighted graph $G = (V, E, w)$

$$w : E \rightarrow \mathbb{R}.$$

- weight of a subgraph H

$$w(H) = \sum_{e \in E(H)} w(e)$$

- min weight sp. tree is a sp. Tree T such that

$$w(T) \leq w(S)$$

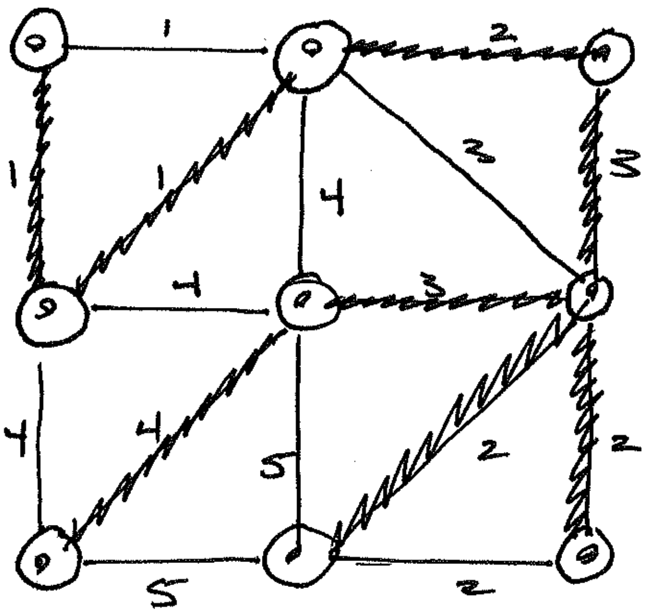
for all sp. trees S in G.

Prim's Algorithm

- choose an init. vertex (which is a tree.)
- amongst all edges incident with the current tree T whose addition to T would not create a cycle, choose one of minimum weight.
- add that edge to T

— repeat and stop after $n-1$ edges were selected.

Ex.



$W(T) = 18$

Thm
When Prim is complete, T is a mwst in G.

Kruskal's Algorithm

- choose an edge of min weight
- Amongst all edges which do not create a cycle with previously selected edges F, choose one of min weight
- add that edge to F
- stop when $n-1$ edges have been selected.

Thm when complete $T = (V(G), F)$ is a MWST in G.

Proof:

let T be the sp. tree in G produced by Kruskal, and let S be any other sp. tree. we must show

$$w(T) \le w(S).$$

Let $\{e_1, e_2, e_3, \dots, e_{n-1}\} = E(T)$ be the edges of T in the order selected by Kruskal.

since $S \neq T$, there is a first edge e_K which is not in S.

i.e.

- $\{e_1, e_2, \dots, e_{k-1}\} \subseteq E(S)$
- $e_k \notin E(S)$

Let H be the subgraph

$$H = S + e_k$$

By the treeness thm, H contains a unique cycle, which includes e_k, call it C. Note C must contain an edge e of S which is not in T, for otherwise C is contained in T

Now remove e from H to get a subgraph R

$$R = H - e = S + e_k - e$$

R is connected since e is a cycle edge. R has $n-1$ edges. $\therefore$ R is a sp. tree.

Kruskal's algorithm guarantees that

$$w(e_k) \leq w(e).$$

<u>Pf</u>: e does not form a cycle with $\{e_1, e_2, \ldots, e_{k-1}\}$ since $\{e_1, \ldots, e_{k-1}, e\}$ $\subseteq E(S)$. Thus if $w(e) < w(e_k)$

Kruskal would have chosen e instead of e_k on k^{th} iteration. ∎

∴ R is a sp. tree of G with one more edge in common with T than S. Also

$$w(R) \le w(S).$$

If $R = T$ we are done, otherwise perform the same construction with R in place of S, to get R_2 having yet another edge in common with T, and $w(R_2) \le w(R)$.

eventually

$$W(T) \leq \dots \leq w(R_2) \leq \underset{\overset{\shortparallel}{R_1}}{w(R)} \leq \underset{\overset{\shortparallel}{R_0}}{w(S)}$$

so $w(T) \leq w(S)$, as required. $\blacksquare$

See

16.3 Huffman Codes

16.4 Matroids

16.5 Scheduling unit-time tasks.

Lower Bounds & Comp. Complexity

Consider P a problem, and all of its instances. we have 2 goals

(1) find an algorithm that solves P, and ~~an find~~ an asymptotic upper bound on its runtime. $O(f(n))$. we try to minimize $f(n)$ as far as possible.

(2) Prove that <u>any</u> algorithm solving P must run in time $\Omega(g(n))$. we try to increase $g(n)$ as much as possible

we're happy when

$$f(n) = \Theta(g(n))$$