CS 170 Discussion 5

Outline

MT1

Discussion 4 (exam review)

Lecture	Topics
5	FFT, DFS
6	SCC, Dijkstra's
7	Dijkstra's, Bellman-Ford
8	Bellman-Ford, MST (MST Algorithms if time)

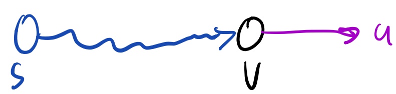
Greedy algorithms!

Update step

We see it both in Dijkstra's and in Bellman-Ford. We have just computed a new dist[v], and want to update the distance of a neighbors.

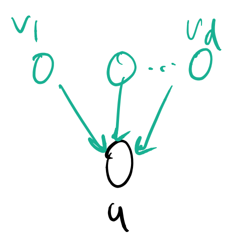
not necessarily the actual distance, just our current estimate

dist[v]



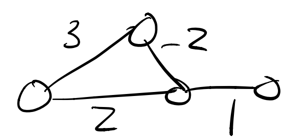
$\text{dist}[u] = \min(\text{dist}[u], \text{dist}[v] + w(v \to u))$

Notice: for a specific u, if the distances of $v_1, \dots, v_d$ are correct, and we run update($v_1 \to u$) ... update($v_d \to u$), the distance to u will be correct.



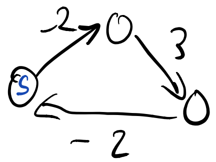
$$\text{dist}[u] = \min_{1 \le i \le d} \{\text{dist}[v_i] + w(v_i \to u)\}$$

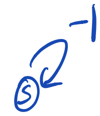
Issue: Dijkstra's update very order-dependent. Can mess up w/ negative edges. So just forget order!



We could just run update() enough times so all distances will be correct! max path length $= |V|-1$, run $|V|$ times on all edges $\Rightarrow O(|V||E|)$. <u>Bellman-Ford</u>

If the distances keep changing, we've hit a negative cycle:



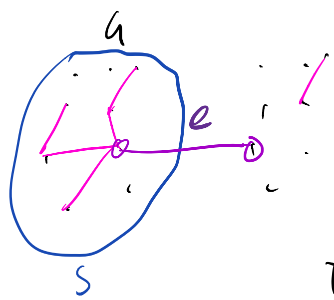


just follow this ∞ times to get small paths

Run once more to check for them.

MST: choosing edges greedily until we get a "small" tree.
We want the smallest tree on G (by weight) that touches all the vertices. minimum tree spanning

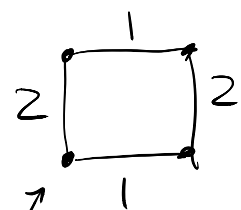
Cut property:



1. G graph (connected, weighted, undirected).
2. $S \subseteq V$ a cut (nonempty)
3. $F \subseteq E$ contained in some MST
4. e is a min weight edge (not unique necessarily) crossing the cut.

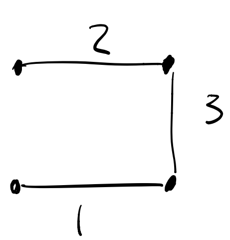
Then $F \cup \{e\}$ is contained in some MST.
(if e unique, it must appear in any MST)

Notice if $F = \emptyset$, then we can show that individual edges show up in some MST with less work needed. But they need not appear in the same MST

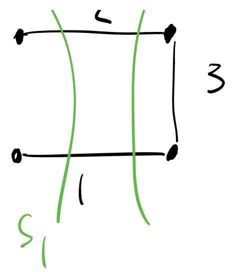


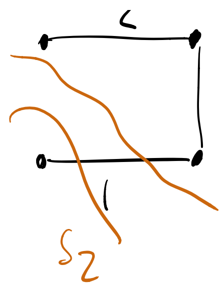
which must appear in an MST?

Warn:

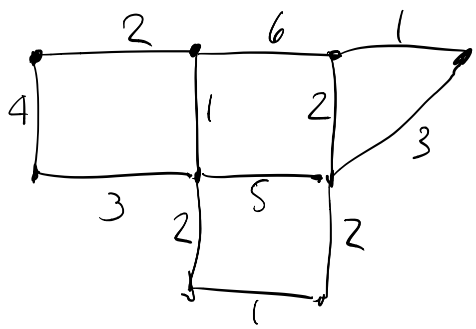


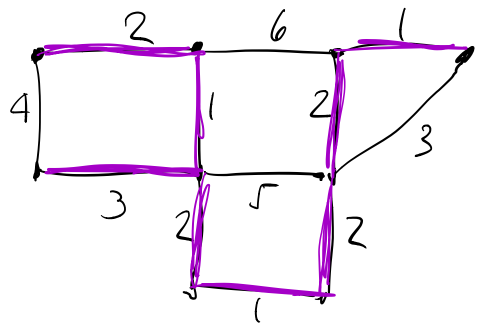
edges of weight 2 and 1 both contained in MST if we look at cut S_1.
But 1 is in MST by cut property, using S_2.



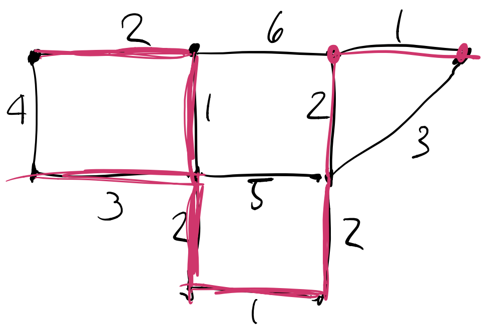


Prim's & Kruskal's



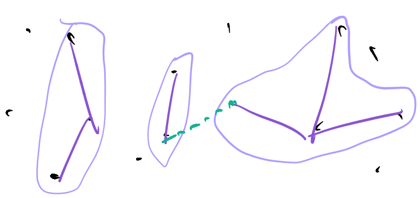


Kruskal



Prim's

Kruskal: sort edges, then add. Cuts look like:



Prim: add the shortest edge that connects to the rest. Cuts look like: