Advanced Algorithms

Lecture 10: MST (contd.), local search

Announcements

- HW 2 due tomorrow!
- HW 1 grading, comments (Vivek Gupta)

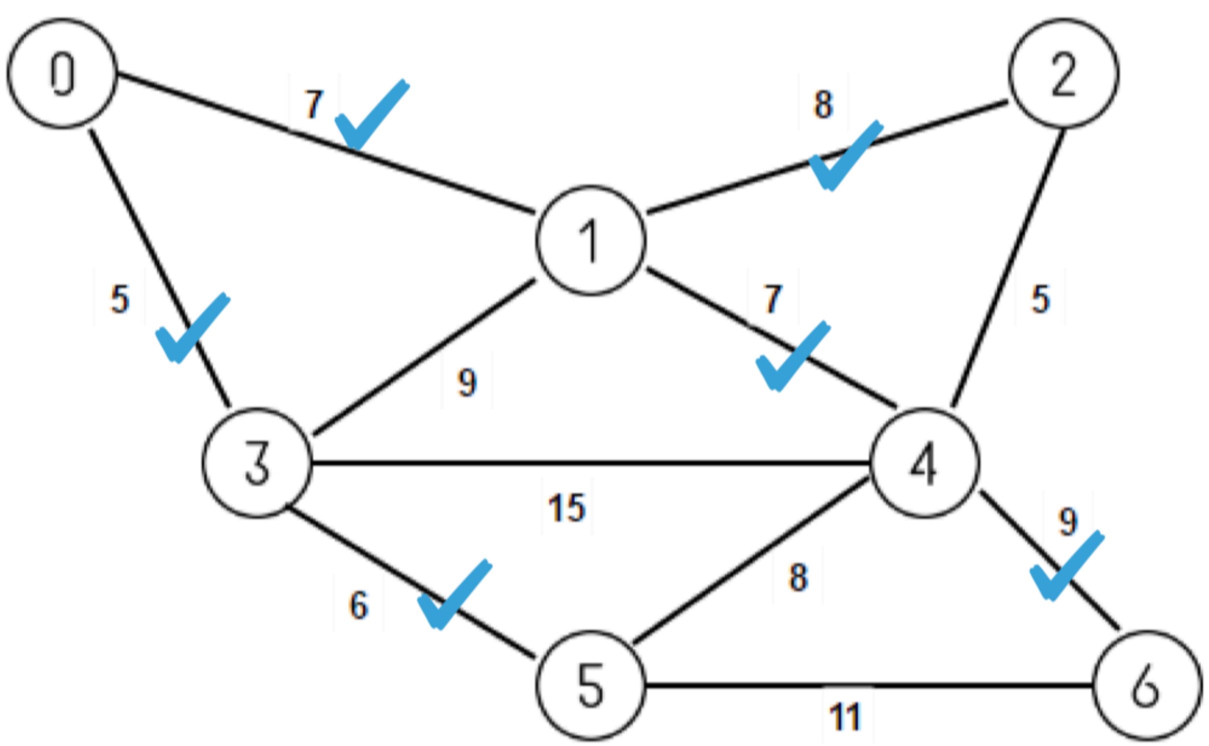
Greedy algorithms – comments

- Usually “easy” to come up with (we are naturally myopic)
- Usually not optimal — examples, Traveling salesman, set cover, …
- (Due to this..) analysis is usually tricky

Example 2: spanning trees

Problem: let G = (V, E) be a (simple, undirected) graph with edge weights $\{w_e\}$ (>0). Pick a subset of the edges, such that (a) all vertices are "connected", (b) total weight of edges is minimized

(Communication backbone in a network)



Greedy strategy

- **Goal:** need to connect all vertices to one another

- Prim: Start with one vertex, add a new vertex to connected set each time

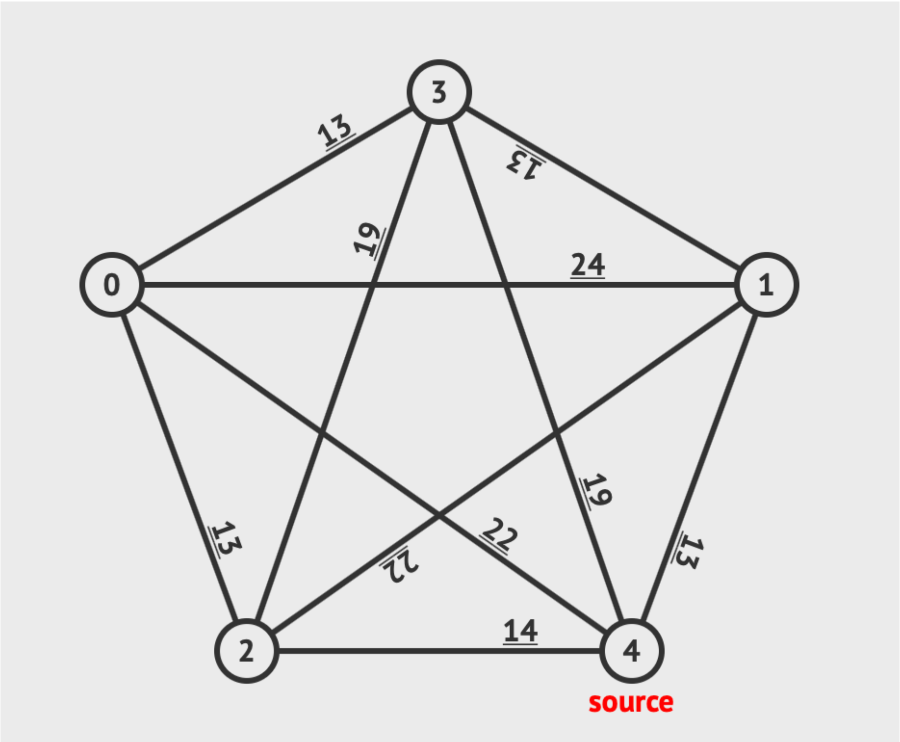
- Kruskal: Add edges one at time, choose min weight edge that isn't "redundant"

Surprise: both turn out to be optimal!

Prim’s algorithm

- start with $S_1 = \{u\}$
- for $t = 1, \ldots, n-1$:
    - add least wt edge out of S_t

https://visualgo.net/en/mst



Correctness

- **Observation:** at each iteration, we have a *set of connected vertices* — S_t
- Will show: There exists a min spanning tree for the *full graph* that contains all edges chosen so far — **structural assumption**

Inductive proof: assuming there's an MST for the full graph containing edges added until *t*, prove that there's an MST for the full graph containing edge added at *t+1*

Proof of “opt prefix” property

Proof of “opt prefix” property

Running time

Minimum spanning tree

- Simple algorithms — analysis slightly tricky
- Common inductive approach for greedy algorithms: show that there's an optimal solution that agrees with all choices so far
- Can be solved in O((m + n) log n) time
- Procedure closely related to shortest paths — Dijkstra's algorithm

Local search

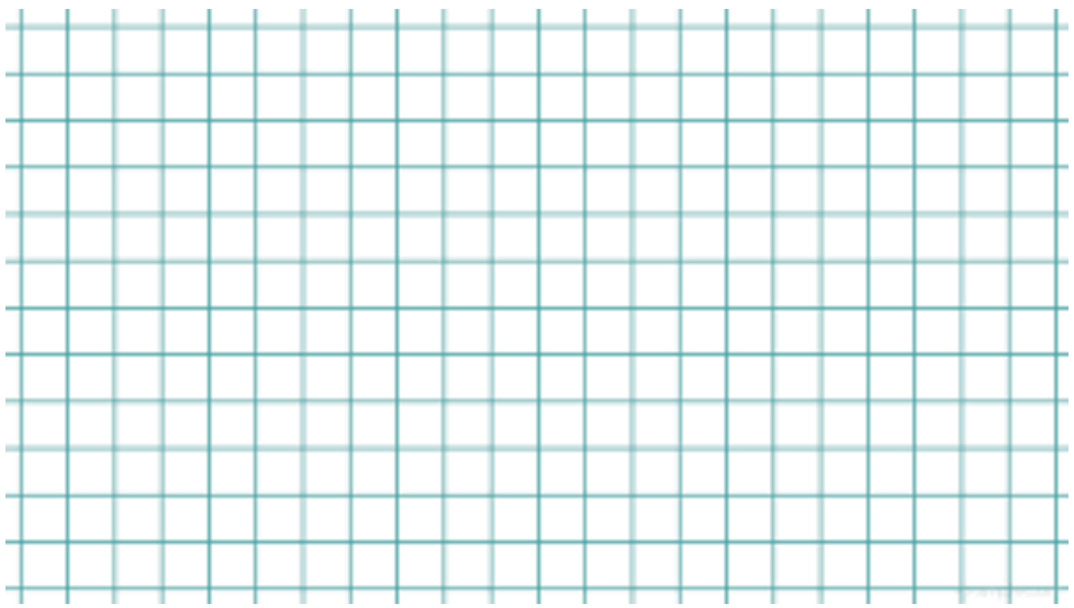
Main idea

- Start with *any* solution, try improving by moving to “nearby” solution
- Stop if no nearby solution is better

Classic example – function opt

Problem: Let f(x) be a function defined on domain D. Find argmin_x f(x)

Multi-variate functions



When is it optimal?

- Any *local optimum* is actually “global” optimum (opt over domain)
- Does this property hold for some natural class?

Minimizing a convex function

(over all of R^n)

Gradient descent

(all of modern ML)

- What is a direction in which function value drops?
- General algorithm:

Matching problem

Problem: suppose we have *n* children and *n* gifts. Each child has some "happiness value" (V_{ij}) for each gift. Find an allocation (one gift per child) so that total happiness is maximized.

Matching – greedy?

Local search

Local search

Claim: take any solution S in which swaps do not increase value. Then total happiness of S >= (1/2) total happiness of OPT solution

2 approximation – proof

Claim: take any solution S in which swaps do not increase value. Then total happiness of S >= (1/2) total happiness of OPT solution