Lecture 14 – Greedy Algorithms in Graphs

Reading: KT Sections 4.4 and 4.5

Shortest path problem

• Input:
  • Directed graph G=(V,E)
  • Weight function w

• What's the weight of a path?
• What's a shortest path?

• What's the output of that problem?

Example*
What would happen if we have negative edges?
Dijkstra’s “water-ripple” idea

- Start with initial vertex in visited set
- Maintain sets of visited vs. unvisited vertices
- Expand visited set one vertex at a time
- Look at all arcs from visited set to any vertex in unvisited set.
- Choose the vertex that is cheapest to include and visit it

Dijkstra’s algorithm

Dijkstra’s Algorithm \((G, \ell)\)

Let \(S\) be the set of explored nodes

For each \(u \in S\), we store a distance \(d(u)\)

Initially \(S = \{s\}\) and \(d(s) = 0\)

While \(S \neq V\)

Select a node \(v \notin S\) with at least one edge from \(S\) for which

\[d'(v) = \min_{u \in S \setminus v} d(u) + \ell_v\]

is as small as possible

Add \(v\) to \(S\) and define \(d(v) = d'(v)\)

EndWhile

A bit of analysis

- This algorithm starts at a start node \(s\).
- For every iteration of the algorithm, the closest node to \(s\) is added to the set of results.
- You can prove by induction that for node \(u\) added to the result set \(S\),
  - The distance from \(s\) to \(u\) is always the shortest distance
- Complexity of this algorithm is \(O(\quad )\)

Minimum Spanning Tree problem
Repairs on a budget (Formally)

- Input:
  - Undirected graph $G = (V,E)$
  - With edge weights $w(u,v)^*$

- Output:
  - Find a subset of the edges $T \subseteq E$ that connect all the vertices**
  - Such that:
    $$w(T) = \sum_{(u,v) \in T} w(u, v)$$

* Can weights be negative?
** What are the properties of $T$?
Prim in action

A bit of analysis

• Note that both algorithms greedily add edges to the MST

• How to prove the optimality of such algorithms?
  • The main idea is that both algorithms try to find the “best” edge that connects parts of the spanning tree together
  • That edge will always have the minimum cost
  • That edge will never create cycles

• Complexity of this algorithm completely depends on the details of its implementation.

Proof sketch

• Assume the MST include e

If we’re lucky

• If T contains e, then we are done!
Otherwise...

- We need to find $e'$ that can be exchanged with $e$
  - How can we find $e'$?