Lecture 12 – Greedy Algorithms in Graphs

Reading: KT Sections 4.4 and 4.5

Partial content of these slides have been obtained from the official lecture slides that accompany the textbook. A complete set of slides can be found at: http://www.cs.princeton.edu/~wayne/kleinberg-tardos/

Shortest Path problem
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*

---

![Diagram of a directed graph with weights on the edges.](image-url)
Example

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
    \(d'(v) = \min_{u \in S, e \in \{u, v\}} d(u) + \ell_e\) is as small as possible
  - Add \(v\) to \(S\) and define \(d(v) = d'(v)\)
EndWhile

Why does this work?
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 $O(   )$