EECS3101 notes:: More on graph algorithms

Jerry Wu

2023-03-30

Shortest path algorithms

Abstract

A shortest path is simply the path of minimum weight on a weighted graph. Applications of this idea include, but are not limited to::

- Network routing
- Map/route generation in traffic
- Robotic motion

Types of shortest path problems

There are 3 main types of shortest path problems in graph theory.

Single source, all destinations shortest paths

- BFS on an unweighted graph.
- Dijkstras(LAN)/Bellman-Ford(WAN) for weighted graphs

Single source, single destination

- All known algorithms for this kind of problem have the same worst case running time as single source, all destinations algorithms. In other words, just find all of them!

All pair shortest paths

- A naive approach would be to run single source shortest path algorithm $\forall v \in V$
- FLoyd-Warshall algorithm

- Johnson's algorithm

Properties of shortest path problems

"How many times did we use proof by contradiction? I don't know, 3 times a week?"-Larry YL Zhang 2023

Optimal substructure

- Sub paths of shortest paths are also shortest paths.

- Proof by cut and paste:: Assume (x, y) is not a shortest path, then $\exists$ another (ψ, ϕ) that is even shorter $\implies$ contradiction.

Relaxation

- $\forall v \in V$, we maintain $d[v]$, the estimate of the shortest path from the source node s initialized to ∞ at the start.

- Relaxing an edge (u, v) means testing whether we can improve the shortest path to v found so far by going through u.

```
Relax(u,v,w):
if d[v]>d[u]+w(u,v) then:
    d[v]=d[u]+w(u,v)
    pi[v]=u
```

"Today's lecture is going to be very relaxing. Today is gonna be a medication session"-Larry YL Zhang

Dijkstra's algorithm

- Non negative edge weights, i.e. $\forall \epsilon \in E, w(\epsilon) > 0$

- It's like BFS but uses a priority queue (similar to Prim's algorithm). Key for the min priority queue is $d[v]$.

The main idea

- Maintain a set S containing the solved vertices
- At each step,
  - Select the closest vertex u.
  - add it to S.
  - relax all edges from u.
- Repeat until the queue is empty.

```
Dijkstra(G,w,s):
    for v in V:
        d[v]=infinity
    d[s]=0
    S=null
    Q=V
    while Q != null:
        u=ExtractMin(Q)
        S=union(S, [u])
        for(v in Adj[u]):
            if(d[v]>d[u]+w(u,v)):
                d[v]=d[u]+w(u,v)
```

Proof of correctness

- Loop invariant
  - A the end of each iteration, $\forall v \in S, d[v]$ is the real shortest path distance from the source vertex to v.
- The proof is trivial and is left as an exercise to the reader (24.3). In summary, it is impossible to find a path shorter than the shortest path.

$$\begin{aligned} d[y] = \delta(s,y) \leq \delta(s,u) \leq d[u] &\implies d[u] \leq d[y] \\ &\implies d[u] = d[y] = \delta(s,y) = \delta(s,u) \blacksquare \end{aligned}$$

Why is $d[u] \leq d[y]$? This is because u is selected by $ExtractMin(Q)$, so it will always be the lesser value for $d[v]$. So it is the thing we will be adding to S.

Runtime of Dijkstra's

- We execute $ExtractMin$ exactly $|V|$ times.
- We execute $DecreaseKey$ exactly $|E|$ times.
- Time complexity depends on the priority queue implementation. With a min heap priority queue, we have that the time complexity is::

$$|V|\log(|V|) + |E|\log(|V|)\Theta(|E|\log(|V|)$$

Bellman-Ford algorithm

Dijkstra's algorithm doesn't allow for **negative** weighted edges, so we utilize BF instead.

- BF uses dynamic programming (they're actually the people who invented dynamic programming!)
- It can detect negative cycles (returns false because the "shortest" path distance will be $-\infty$)
- returns shortest path tree otherwise

```
BellmanFord(G,w,s):
    for v in V:
        d[v]=infinity
    d[s]=0
    pi[s]=null
    for i in range |V|-1:
        for (u,v) in E:
            Relax(u,v,w)
    for (u,v) in E:
        if d[v]>d[u]+w(u,v):
            return false
    return true
```

Runtime is $\Theta(|V| \times |E|)$

Loop invariant of BF

- At the end of the k-th iteration, $d[v]$ is the shortest path distance over all paths that contain at most k edges (24.1).