Carlos Mc Gregor c.mcgregormuro@mail.utoronto.ca

CSC373S: Algorithm Design, Analysis & Complexity

Lecture 14

Monday February 6, 2017
based on notes by Denis Pankratov

Graph Algorithms

Shortest Path Problem

Single-source version

Input: Weighted graph (directed or undirected):
$G = (V, E)$
$C : E \to \mathbb{R}$
$S \in V$ source

Definition: $G = (V, E)$
$C : E \to \mathbb{R}$
$\forall a, v \in V, d(u, v) =$ length of a shortest path from u to v or ∞ if no such path exists.

Output: $dist[]$ so that $\forall u \in V, dis[u] = d(s, u)$

Observations:

1. If a *negative-weight cycle* is reachable from s, then the problem is *not well-defined*.
   Thus, assume no such cycles in the input. Therefore, we may take shortest paths to be simple - no cycles
2. The problem has optimal substructure.
   Say *shortest path* from s to v is $s = v_0, v_1, \ldots, v_k = v$
   $\to v_0, v_1, \ldots, v_i$ is a shortest path from s to v_i
   $\to$ shortest paths from s can be represented as a tree called shortest path tree

   Special case: $\forall e, c(e) = 1$
   (*Solved by BFS in the* $\mathcal{O}(|V| + |E|)$)
   Less special case: $\forall e, c(e) > 0$

   Solved by Dijkstra's algorithm - like BFS but w/ priority queue instead of regular queue

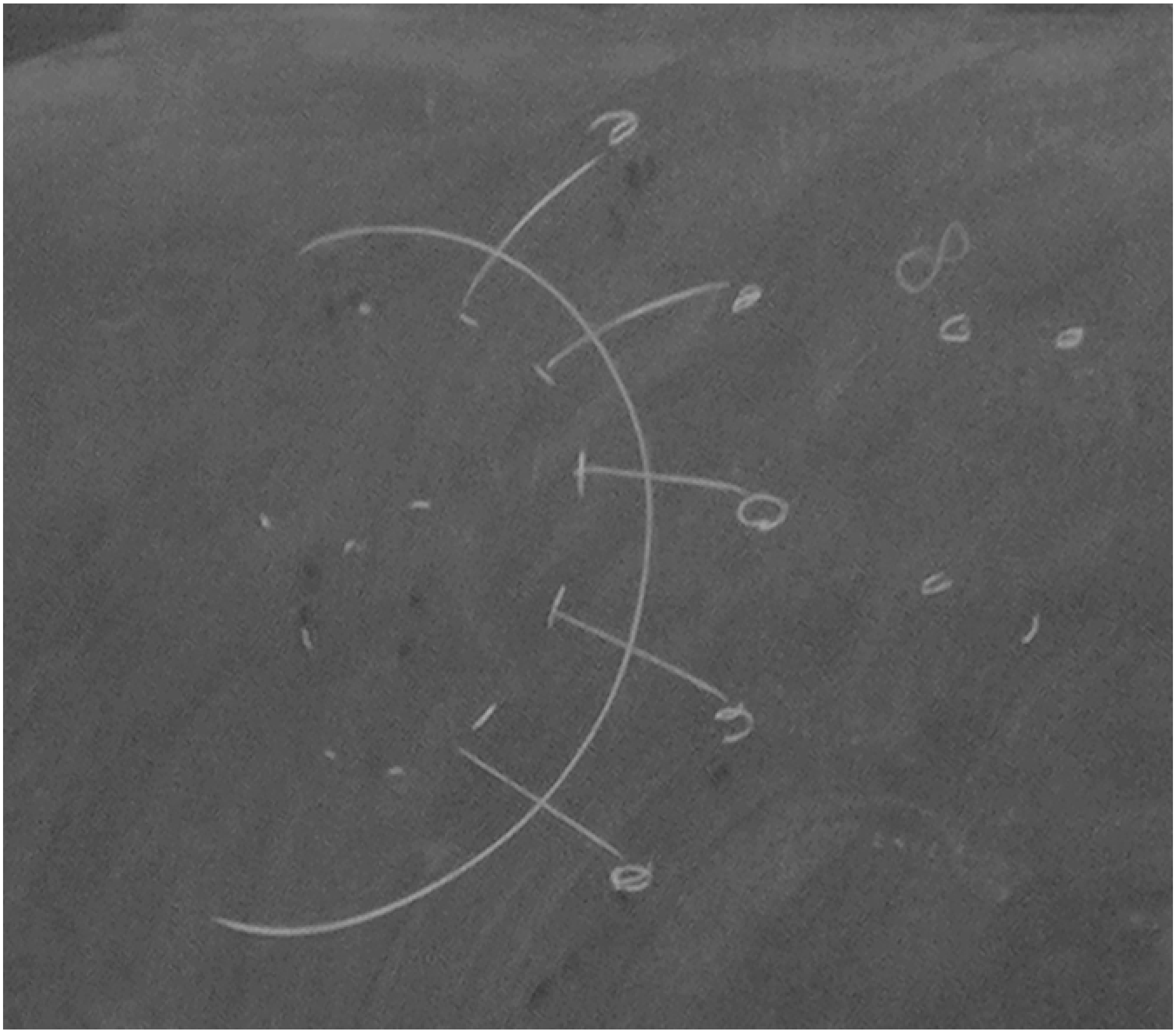

Figure 1: *Dijkstra Visualization*

```
def Dijkstra(Adj, c, s):
  init arrays dist[], prev[] of size |V| each

    for v in V:
      dist[v] = float('inf')
      prev[v] = None

    dist[] = 0

    Q = MinPriorityQueue(V) /* by dist[] */

```

```
    while Q is not empty:
        v = Q.ExtractMin()
        for u in Adj[v]:
            if dist[v] + c[v,u] <= dist[u]:
                dist[u] = dist[v] + c[v,u] // Decrease key
                prev[u] = v

    return dist
```

Runtime:
$1\times$ Build PQ
$|V|\times$ ExtractMin
$|E|\times$ Decrease Key

Example: if PQ is implemented w/ binary heaps, $\mathcal{O}((|V|+|E|)\log|V|)$.
Dijkstra is *greedy*.
Let $T_i = (V_i, E_i)$ be the tree constructed by Dijkstra by i^{th} step.

Loop Invariant:

(a) T_i can be extended to a *shortest path tree*
(b) $\forall u \in V_i, \forall v \notin V_i$, we have $dist[u] = d(s,u) \leq d(s,v) \leq dist[v]$

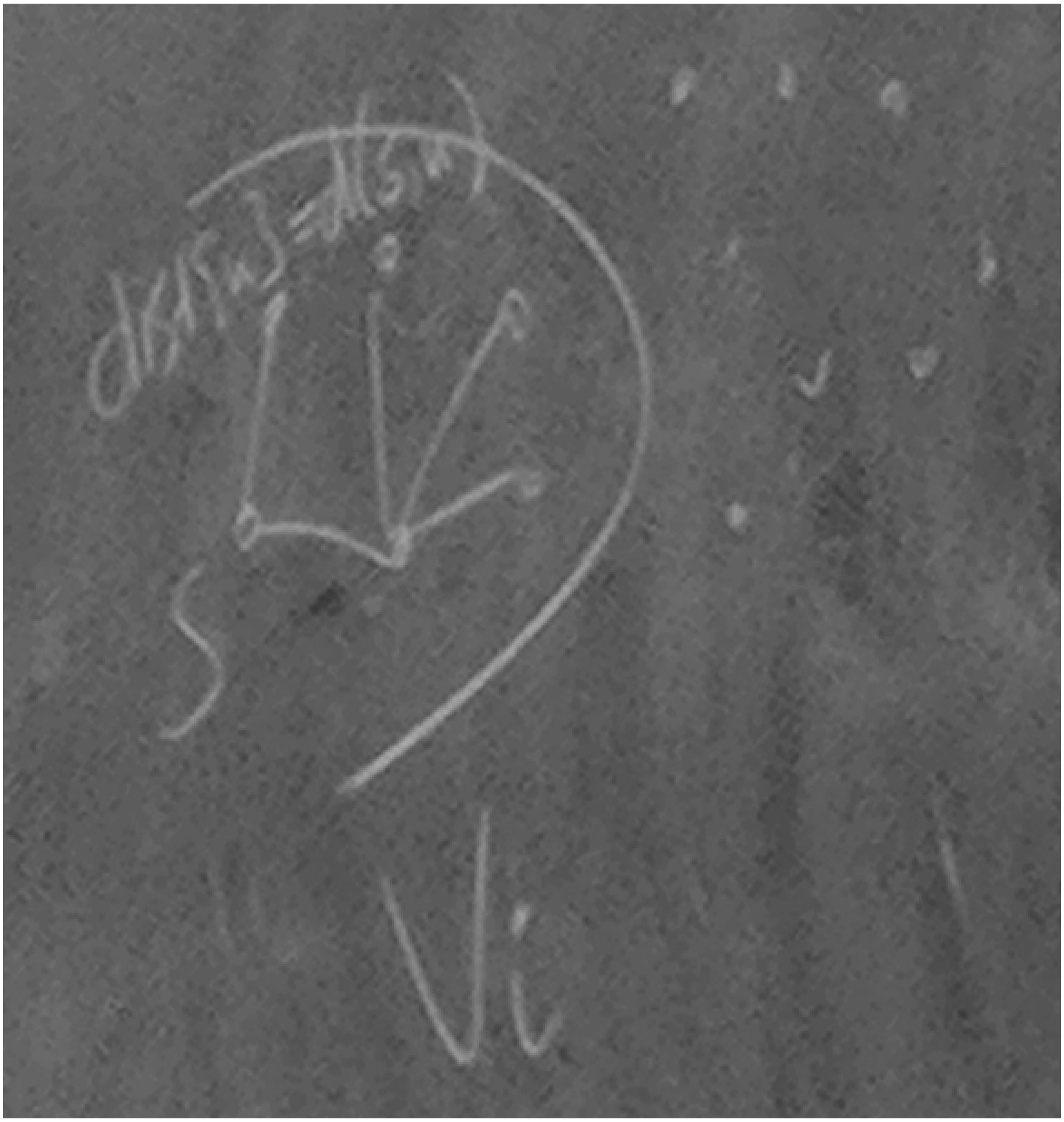

Figure 2: *Visualization of what is going on*

<u>Proof by Induction on i</u>
Base case: $i = 0$
$T_0 = (s, \varnothing)$, clearly extends to some shortest path tree

$dist[s] = 0 = d(s, s)$
$dist[v] = \infty$ if $v \neq s$ so (2) is also satisfied

Induction Assumption: Assume that T_i extends to shortest path tree $T*$ (some $i \geq 0$) & (2)

Inductive Step: Suppose the algorithm selects $edge(u, v)$ in step $i + 1$:

$$T_{i+1} = (V_i \cup \{v\}, E_i \cup \{(u, v)\}$$

Case 1: $(u, v) \in E(T*) \to (1)$ is satisfied

$$\begin{aligned} d(s, v) &= d(s, u) + c(u, v) \text{ (by defn of } T*) \\ &= dist[u] + c(u, v) \text{ (by IA)} \\ &= dist[v] \text{ (by Dijkstra's)} \end{aligned}$$

Since $dist[v]$ was smallest, this completes part (2) of LI.

Case 2: $(u, v) \notin E(T*)$
Let $(w, v) \in E(T*)$. Assume for contradiction $w \notin V_i$
Let (x, y) be the edge w/ $x \in V_i$ & $y \notin V_i$

Figure 3: See *computation below*

$$\begin{aligned}
d(s,v) &= d(s,x) + c(x,y) + d(y,v) \text{ (by definition of } T*) \\
&= (dist[x] + c(x,y)) + d(y,v) \\
&\geq (dist[u] + c(u,v)) + d(y,v) \\
&> dist[u] + c(u,v) \geq d(s,v)
\end{aligned}$$

$\rightarrow w \in V_i$

$$\begin{aligned} d(s,v) &= dist[w] + c(w,v) \\ &\geq dist[u] + c(u,v) \text{ (by the greedy choice) (\textit{Note:} in fact, must have equality!)} \end{aligned}$$

To get (1), consider $(T * \{(w,v)\}) \cup \{(u,v)\}$.
(2) follows from (*) + observing how edges are updated by Dijkstra

Like BFS, but with priority queue!