CS483 Analysis of Algorithms
Lecture 04 – Greedy Algorithms

Jyh-Ming Lien

June 15, 2017

Greedy Algorithm

- Greedy algorithm is algorithm that makes the **locally optimal** choice at each stage with the hope of finding the global optimum
- Greedy algorithm never changes the choices that have been made
- Example: How do you compute the minimum number of US coins ($25, $10, $5, $1) to give while making change of 43 cents?

- Advantages
  - Simple and Intuitive
  - Work for problems such as **minimum spanning tree**, **shortest path problem**, and **data compression**.
- Disadvantages
  - Be very careful when use it. May not work for many problems
  - But still provide good approximate solution

Minimum Spanning Tree (MST)

- Given a set of points (cities), how do you use the minimum amount of wire to connect these points?
- Example:

- Given a graph G, a **spanning tree** T of G is a subgraph of G that contains all vertices of G
- The **minimum spanning tree** MST of G is a spanning tree of G of the smallest weight
- MST has many applications including: clustering, good approximation to traveling salesman problem, point connecting problem

Prim's Algorithm

- Algorithm

Algorithm 0.1: PRIM($G = \{V, E\}$)

$x \leftarrow$ a random vertex from G
$V_{MST} \leftarrow \{x\}$
$E_{MST} \leftarrow \emptyset$
for $i \leftarrow \{1 \cdots |V| - 1\}$
do $\begin{cases} \text{find the minimum weight edge } e = \{u, v\} \\ \text{such that } u \in V_{MST} \text{ and } v \in V - V_{MST} \\ V_{MST} \leftarrow V_{MST} \cup v \\ E_{MST} \leftarrow E_{MST} \cup e \end{cases}$

- Example:

Prim's Algorithm

- Why is Prim's algorithm correct?

- How to implement the first statement in Prim's algorithm?

- What is the time complexity of this implementation?

Kruskal's Algorithm

- Algorithm

Algorithm 0.2: KRUSKAL($G = \{V, E\}$)

Sort$E$ from small to large
$E_{MST} \leftarrow \emptyset$
while $|E_{MST}| < |V| - 1$

$$\textbf{do} \begin{cases} \textbf{if } E_{MST} \cup E_i \text{ is acyclic} \\ \quad \textbf{then } E_{MST} \leftarrow E_{MST} \cup E_i \\ i \leftarrow i + 1 \end{cases}$$

- Example:

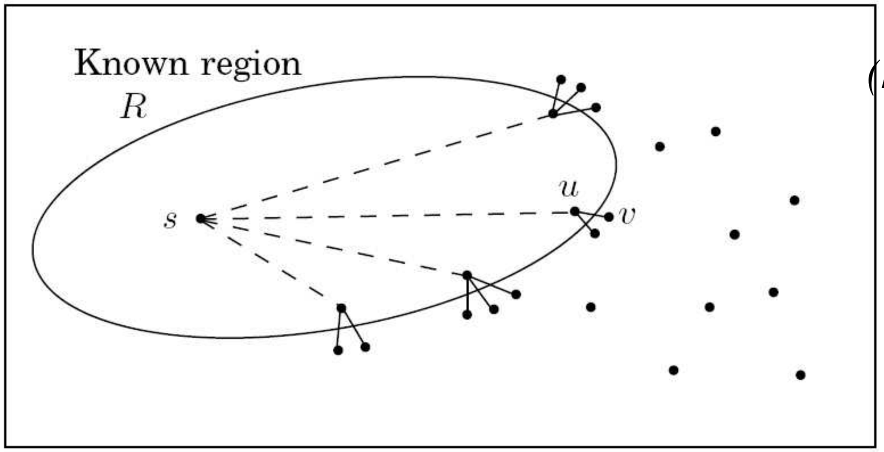
- Why is Kruskal's algorithm correct?

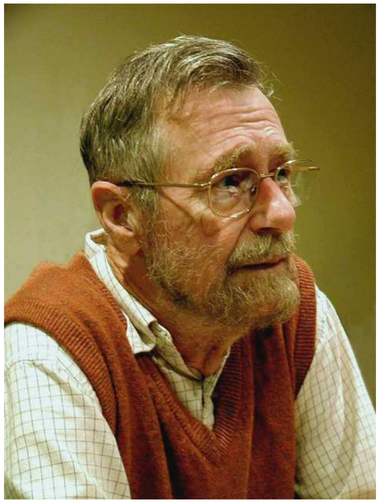
- How to check the acyclic property in Kruskal's algorithm efficiently?

- What is the time complexity of this implementation?

Dijkstra's algorithm

- Edsger Dijkstra (1930-2002): one of the most influential computer scientists
- Dijkstra's algorithm works by extending the current *shortest-paths tree* to the next closest vertex (to the source)
- Example:





(http://www.cs.utexas.edu/users/EWD/)

Dijkstra's algorithm

- Algorithm

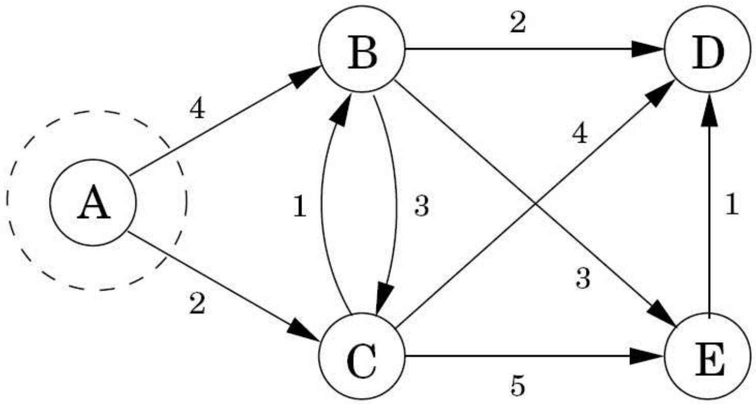
Algorithm 0.3: DIJKSTRA($G = \{V, E\}, s$)

for each $v \in V$
  do $\begin{cases} v.dist \leftarrow \infty \\ v.parent \leftarrow \emptyset \end{cases}$
$s.dist \leftarrow 0$
for $i \leftarrow \{1 \cdots |V| - 1\}$
  do $\begin{cases} v \leftarrow \text{min\_dist}(V) \\ \text{remove } v \text{ from } V \\ \textbf{for each } n \text{ adjacent to } v \\ \quad \textbf{do} \begin{cases} \textbf{if } v.dist + \overline{vn} < n.dist \\ \quad \textbf{then} \begin{cases} n.dist \leftarrow v.dist + \overline{vn} \\ n.parent \leftarrow v \end{cases} \end{cases} \end{cases}$

- What data structure is needed to perform this algorithm?

Dijkstra's algorithm

- Example:



Notes on Dijkstra's algorithm

- Does not work for graphs with **negative weights** (does it work for unweighted graph?)

- Applicable to both undirected and directed graphs

- Efficiency:
  - $O(|V|^2)$ for graphs represented by weight matrix and array implementation of priority queue
  - $O(|E|log|V|)$ for graphs represented by adj. lists and min-heap implementation of priority queue

Will Dijkstra's algorithm still work?

- Example: