CS472 Module 6 Part B - Floyd-Warshall's Algorithm

Athens State University

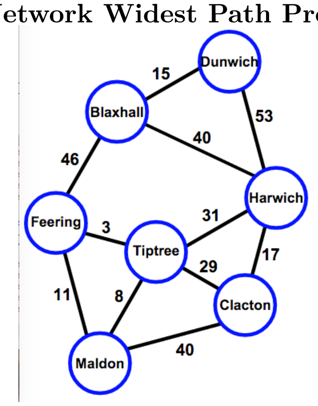
March 31, 2014

Outline

Contents

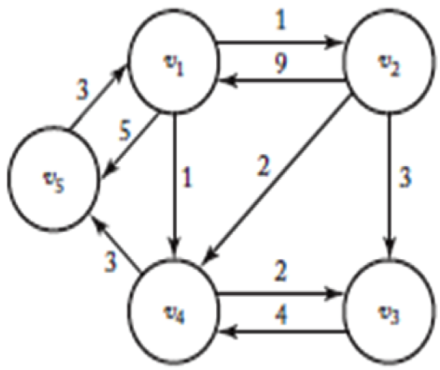
1 Problem statement

Network Widest Path Problem



- The nodes in the graph are wide-area routers on the Internet
- Weight of the bi-directional edges are an estimate of the bandwidth
- Need to find an end-to-end path between two nodes with the maximum possible bandwith

Problem Statement



- A graph $G = (V, E)$ has its vertices numbered from 1 to N.
- Suppose there exists a function $SP(i, j, k)$ that returns the shortest possible path from i to j using only vertices from the set $1, 2, \ldots, k$ as intermediate points.
- We wish to find the shortest path from each node i to each node j using only the vertices from 1 to $k + 1$.

Some important facts

- For each pair of vertices, the true shortest path could be
  - a path that only uses vertices in the set $1, 2, \ldots, k$
  - a path that goes from i to $k + 1$ and then from $k + 1$ to j
- The function $SP(i, j, k)$ gives the best path i to j using $1, \ldots, k$
- if there were a better path from i to $k+1$ to j, then the length of this path would be the concatenation of the shortest path from i to $k + 1$ and the shortest path from $k + 1$ to j
  - In both cases, using the vertices in $1, \ldots, k$

A recurrence relation

If $w(i, j)$ is the weight of the edge between vertices i and j, then we can compute $SP(i, j, k + 1)$ using the formula:

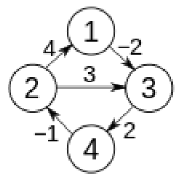
$$SP(i, j, 0) = w(i, j)$$
$$SP(i, j, k + 1) = \min(SP(i, j, k), SP(i, k + 1, k) + SP(k + 1, j, k))$$

2 The Algorithm

Graph Representation

- This is a situation where we should use the adjacency matrix representation of a graph
- Let each entry, in the matrix represent the weight of the edge

$$W(i,j) = \begin{cases} w\{i,j\} & edge(i,j) \\ \infty & \text{not } edge(i,j) \\ 0 & \text{if } i = j \end{cases}$$

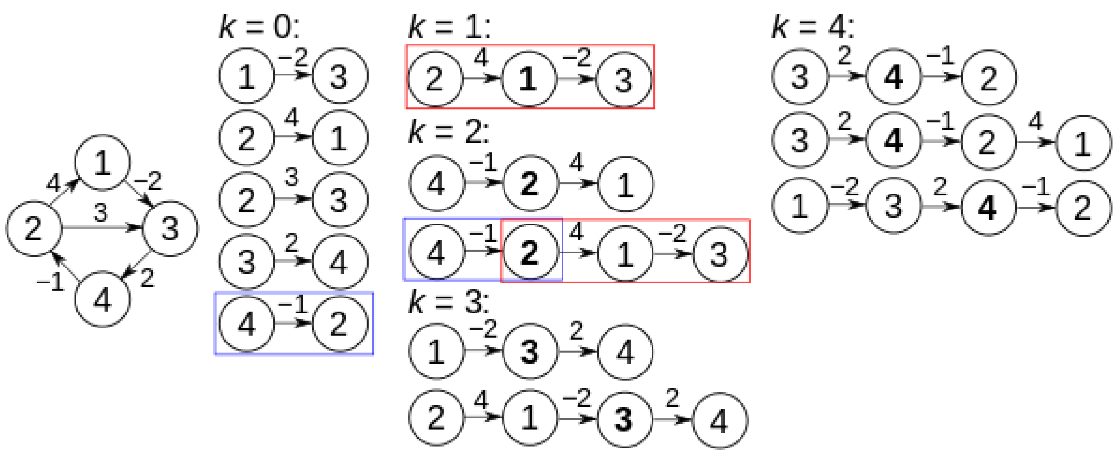


The Floyd-Warshall All-Pairs Shortest Path Algorithm

Algorithm 1: The Floyd-Warshall All-Pairs Shortest Path Algorithm

Input: A graph G=(V,E) with weights w(u,v) (using adj. matrix)
Output: A |V| by |V| array dist with minimum distances
dist[][] $\leftarrow \infty$;
for $v \in V$ **do**
  dist[v][v] $\leftarrow$ 0;
for $e \in E$ **do**
  dist[u][v] $\leftarrow$ w(u,v);
sizeV $\leftarrow$ **for** $k \in$ *[1..sizeV]* **do**
  for *i in [1..sizeV]* **do**
    for *j in [1..sizeV]* **do**
      if *dist[i][k] > dist[i][k] + dist[k][j]* **then**
        dist[i][j] $\leftarrow$ dist[i][k] + dist[k][j];

Floyd-Warshall Algorithm: Example



- At $k = 0$, the only known paths are the single edges in the graph
- At $k = 1$, we look at paths going through node 1
- At $k = 2$, we look at paths going through the nodes $\{1, 2\}$

Floyd-Warshall Algorithm: Negative cycles

- *Negative cycle*: A cycle whose edges sum to a negative value.
- A negative cycle in a graph means that there is no shortest path between any pair of vertices i, j which form part of the cycle.
  - Because the path-lengths from those two vertices can be arbitrarily small.
- One can detect negative cycles with this algorithm by examining the diagonal of the path matrix and checking to see if any value is negative.

Floyd-Warshall Algorithm: Path Reconstruction

- Basic algorithm only gives lengths of paths between all pairs of vertices
- Want to get the actual path between two endpoint vertices
  - Not required to store actual path from vertex to vertex
  - Just store the information about the highest index intermediate vertex you have to pass through to get to any given vertex
  - Store this information in a parallel $|V|$ by $|V|$ matrix *next* where each entry is the highest index vertex you have to travel through to follow the shortest path from i to j

3 Analysis

Analysis of Floyd-Warshall algorithm

- Let $n = |V|$.
- To find all n^2 values of $SP(i, j, k)$ for all i and j from the values of $SP(i, j, k-1)$ returns $2n^2$ operations
- Algorithm sets $SP(i, j, 0) = w(i, j)$ and computes the sequence of n matrices $SP(i, j, 1)$, $SP(i, j, 2)$, $\ldots$, $SP(i, j, n)$, the total number of operations is $n * 2n^2 = 2n^3$
- Therefore, the algorithm has a time efficiency of $\Theta(n^3)$

4 Applications

Applications

- Shortest path in directed graphs
- Transitive closure of directed graphs
- Basis of Kleene's Algorithm for finding a regular expression accepted by finate automaton (important for theortical computer science and compiler design)
- Inversion of real matrices
- Optimal routing in networks
- Maximum bandwidth path in networks

5 Key Points

Key Points

- Algorithm for find all pairs shortest paths in a graph
- Example application of dynamic programming
- Many applications, esp. in computer networking