Lecture 5: Some Odds and Ends concerning Weighted Graphs

Ian Pratt-Hartmann

Room KB2.38: email: ipratt@cs.man.ac.uk

2016–17
Outline

Weighted graphs: Dijkstra’s algorithm

All-pairs shortest paths: dynamic programming
• A (non-negative) **weighted graph** is a triple \((V, E, f)\), where \((V, E)\) is a graph and \(f\) is a function \(f : E \to \mathbb{R}^+\).

• The length of a path in \((V, E, f)\) is simply the sum of weights on its edges.

• If \((u, v) \in E\), we write \(f(u, v)\) rather than \(f(((u, v)))\), and if \((u, v) \not\in E\), we write \(f(u, v) = \infty\).

• Consider the computation task:

**UNDIRECTED SHORTEST PATH**

Given: A weighted graph \(G = (V, E, f)\) and nodes \(s, t \in V\)

Return: The shortest path from \(s\) to \(t\), or \(\infty\) if \(t\) is not reachable from \(s\).
The following algorithm, known as Dijkstra’s algorithm, computes UNDIRECTED SHORTEST PATH.

begin Dijsktra($V, E, f, s$)
    set $D(s) = 0$.
    for all $u \in V$
        if $(s, u) \in E$,
            set $D(u) = f(s, u)$
        else
            set $D(u) = \infty$
    until $D(u) = \infty$ for all $u \in V \setminus V^*$ do
        choose a $u \in V \setminus V^*$ with $D(u)$ smallest
        add $u$ to $V^*$
        for all (other) $v \in V \setminus V^*$ do
            set $D(v) := \min(D(v), D(u) + f(u, v))$
        return $D(t)$.
end
To see that this works:

- let length of shortest path from $x$ to $x'$ be $d(x, x')$
- suppose $u$ is first node added to $V^*$ with $D(u) > d(s, u)$.
- Consider a shortest path $s$ to $u$ first leaving $V^*$ on the edge $(y, z)$.

\[
V \\
V^* \\
\quad s \\
\quad D(y) = d(s, y) \\
\quad y \\
\quad f(y, z) \\
\quad z \\
\quad d(z, u) \geq 0 \\
\quad u
\]
Weighted graphs: Dijkstra’s algorithm

\[ D(y) = d(s, y) \]

\[ D(z) = D(y) + f(y) = d(s, z) \]

\[ u \]

\[ d(z, u) \geq 0 \]

- \[ D(y) = d(s, y) \], since \( u \) is first ‘incorrect vertex’ added to \( V^* \).
- \[ D(z) = D(y) + f(y) = d(s, z) \], since drawn path from \( s \) to \( u \) is shortest, whence drawn path from \( s \) to \( z \) is shortest.
- But then \( D(u) \leq D(z) = d(s, z) \leq d(s, u) < D(u) \), a contradiction.
Outline

Weighted graphs: Dijkstra’s algorithm

All-pairs shortest paths: dynamic programming
• The following algorithm computes, for a weighted graph 
\((V, E, f)\), the shortest distance between any pair of vertices \(u\) and \(v\).

• The idea is simple:
  • first record the distance between every pair of vertices \textit{via no intermediate nodes}, taking that distance to be \(\infty\) if there is no connecting edge;
  • now, supposing that we have correctly recorded the distances between all pairs of node \textit{with intermediate nodes limited to} \(v_1, \ldots, v_k\), do the same \textit{with intermediate nodes limited to} \(v_1, \ldots, v_k, v_{k+1}\).

• In the following algorithm, we assume that the graph is presented as a pair of arrays:
  • \texttt{connectionArray[i][j]}: \(\top\) if there is an edge from \(v_i\) to \(v_j\);
  • \texttt{distanceArray[i][j]}: weight of edge from \(v_i\) to \(v_j\) in this case.
First, we initialize some arrays:

```java
for(i= 0;i< size;i++)
    for(j= 0;j< size;j++)
        if(i == j)
            distanceArray[i][j]= 0;
        else if(!connectionArray[i][j])
            distanceArray[i][j]= Integer.MAX_VALUE;
        else
            distanceArray[i][j]= inputArray[i][j];
for(i= 0;i< size;i++)
    for(j= 0;j< size;j++)
        if((i != j) && connectionArray[i][j])
            routeArray[i][j]= j; // No detours
        else
            routeArray[i][j]= -1; //Never used
for(i= 0;i< size;i++)
    for(j= 0;j< size;j++)
        accessArray[i][j]= connectionArray[i][j];
```
Now for the actual algorithm:

```c
for(k= 0;k< size;k++)
    for(i= 0;i< size;i++)
        if(i != k)
            for(j= 0;j< size;j++)
                if(i != j ){
                    tempAccess= accessArray[i][k] && accessArray[k][j];
                    tempDistance= distanceArray[i][k]+distanceArray[k][j];
                    if(tempAccess &&
                        tempDistance < distanceArray[i][j]){  
                        distanceArray[i][j]= tempDistance;
                        accessArray[i][j]= true;
                        routeArray[i][j]= routeArray[i][k];
                    }
                }
```
• Reading:
  • G+T: Ch 14, sec. 14.5.1, pp. 412–413.