Basic Algorithms Rec 11

given a graph $G$, weighted, directed. Let a source $s \in V$
and sink $t \in V$, find the shortest path from $s$ to $t$.

**E.g.**

$$
\begin{array}{c}
A & 10 \\
B & 15 & 4 & C \\
D & 14 & 1 & E & 1 \\
\end{array}
$$

min path from

A $\rightarrow$ D

d $A \rightarrow B \rightarrow E \rightarrow D$
or $A \rightarrow C \rightarrow D$

of $wt = 14$.

---

**Algorithm:**

1) Start with the source $s$, set distance = 0, parent = -1

2) Set the distance of every other point = $\infty$

3) We grow a "tree" Dijkstra tree, similar to a Prim's tree except that we look at total distance rather than current edge weight.
Pseudo code

for i = 1 to n
  d[v] = \infty
  p[v] = -1
  H[v] = \infty
  d[e] = 0, H[e] = 0

while ( H is not empty )
  v = H.deleteMin()
  for u vh of v
    if d[u vh] > d[v] + \omega(v, u vh)
      d[u vh] = d[v] + \omega(v, u vh)
      H[u vh] = d[u vh]
      p[u vh] = v


Complexity analysis:

Each edge is examined once to get a decrease-key operation.
& each vertex is extracted once using a deleteMin.

\Rightarrow T(dk) = O(n \cdot \text{tem} + m \cdot \text{tdk})

where \text{tem} = \text{time to extract-min of n-elements}

\text{tdk} = \text{time to decrease-key of n-elements}.
with binary heap

\[ \tau(\mathcal{G}) = O(n + m \log n) \]

with fibonacci heap

\[ \tau(\mathcal{G}) = O(n + u \log n) \]

---

Proof of correctness:

prove by induction that the distance to all visited nodes (nodes removed from the queue) is correct. If to all unvisited nodes the dist is using only visited.

Base case for source is trivial.

Now assume that the first \( k-1 \) vertices visited have correct distance. \& the same for unvisited ones.

Say that Dijkstra chooses some vertex \( v \) as the \( k \)th to visit. \& \( d[v] \) is smaller greater than it's correct value. i.e. \( \exists \) a smaller path to \( v \).

Dijkstra chooses \( v \) as \( u \rightarrow v \) gives smallest weight to \( v \).

Suppose the smallest actual path to \( v \) has some unvisited vertices, say \( w \) is the first one.

\[ \Rightarrow \ d[v] > d[w] \Rightarrow \text{to induction hypothesis.} \]
So there are no unvisited vertices on actual path to \( v \).

\[ \Rightarrow \text{the last vertex on path to } v, \text{ before } v, \]

say \( x \), has been visited

\[ \Rightarrow d_{E}^{*} = d_{E}^{*} + \omega(x, v) \]

but \text{Dijkstra chose } u \text{ because}

\[ d_{E}^{*} + \omega(u, v) \leq d_{E}^{*} + \omega(x, v) \]

\[ \Rightarrow \square \]

\[ \Rightarrow \text{It is true for steps} \]

---

Application for finding colored paths.

Given directed graph \( G = (V, E) \) & coloring of edges \( c: E \rightarrow \{ \text{red, blue, white} \} \).

Find a path from \( s \) to \( t \) using at least one red & one white edge but minimize blue edges.

---

Reduction to \text{Dijkstra}:

we will construct a "super" graph.
Make 4 copies of the graph $G$.

1) if $u \rightarrow v$ was a white edge in the original $G$ add edge of weight $= 0$ between $u_1 \rightarrow v_3$, $u_2 \rightarrow v_4$

2) -- " red " -- $u_1 \rightarrow v_2$, $u_3 \rightarrow v_4$.

3) -- " blue edge, in original $G$ add edge of weight $= 1$, between $u_1 \rightarrow v_1$, $u_2 \rightarrow v_2$, $u_3 \rightarrow v_3$, $u_4 \rightarrow v_4$

Run Dijkstra on this graph & return the path & remove subscripts in the path.
Proof of correctness.

Because we go from $s$, to $t$, we go through at least one red and one white edge.

Because blue edges have weight $\leq 1$, we minimize the number of blue edges.

Any path here is a valid path.

Say the actual path with minimum number of blue edges is $s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k \rightarrow t$.

Then this also is a path in the supergraph. => we will find something equivalent.

Programming Assignment is for only 2 colors.

Notes on Dijkstra:

+ Does not work on negative weight edges.
Floyd-Warshall algorithm

All pairs shortest paths.

find shortest path betw all pairs of points in a directed weighted graph.

Run dijkstra from all points.

$\Rightarrow O(n^3 \log n)$

does not work on "-ve edges.

Floyd-Warshall

- $O(n^3)$
- finds negative cycles / works with "-ve edges.

Very good for dense graphs.

Dynamic Programming.
$DP[i][j][k] = \text{distance of } j \text{ from } i$ using only the first $k$ vertices $1 \ldots k$.

$DP[i][j][0] = \begin{cases} 0 & i = j \\ w(i,j) & (i,j) \in E \\ \infty & \text{otherwise} \end{cases}$

$DP[i][j][k] = \min \left\{ DP[i][j][k-1], \quad \begin{array}{c} DP[i][j][k-1] + \\ DP[k][j][k-1] \end{array} \right\}$

either use the $k$th vertex in which case we use it only once.

we will go from $i \rightarrow k \rightarrow j$

we will go from $i \rightarrow k$, using only $1, \ldots, k-1$ and from $k \rightarrow j$, using only $1 \ldots k-1$ vertices.
for $k = 1$ to $n$
  for $i = 1$ to $n$
    for $j = 1$ to $n$
      \[ \text{DP}[i][j][k] = \min( \text{DP}[i][j][k-1], \text{DP}[i][j][k-1] + \text{DP}[k][j][k-1]) \]

\[ \Rightarrow T(d) = O(n^3). \]

Space needed = $O(n^3)$. 

less space:
change the array for distances by getting rid of last index.
\[ \text{DP}[i][j][k] \quad (k \text{ is implicit}) \]
\[ d = \text{distance from } i \text{ to } j \text{ using first } k \text{ vertices.} \]
\[ (k \text{ is defined by the outer loop}) \]
update the array by getting rid of old values.