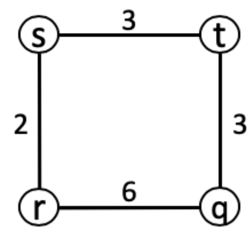
Ruolin Li
31764160

4 Weighted Shortest Paths (WSP)

Finally, we'll turn to weighted, undirected graphs. Finding shortest paths in such graphs has numerous applications: such as finding the cheapest airfare between two cities, or finding the shortest route from where we are to where we want to get to in a maze of hiking trails.



Step 1: Build intuition through examples.

$SP_{s \to q} = 6$

Step 2: Develop a formal problem specification

Step 2: WSP Instance: $G=(V,E)$, $n=|V|\geq 2$, connected. A designated source node $s \in V$
Each edge (u,v) has a weight. $w(u,v)$ $w: E \to \mathbb{R}^+$

Potential solution: Tree which is rooted at s, has n nodes.
Good solution: If path from s to any node $v \in V$, in the tree, is a shortest path from s to v in G
minimum total cost

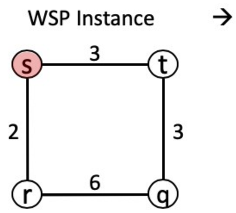
Step 3: Identify similar problems. What are the similarities?

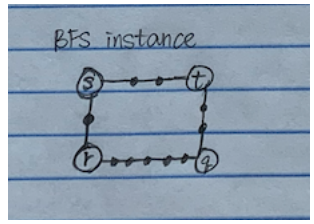
Step 3: ~~Find the~~ BFS finds shortest paths in unweighted graphs &
~~It~~ It produces a tree, rooted at source s.

Step 4: Evaluate simple algorithmic approaches.

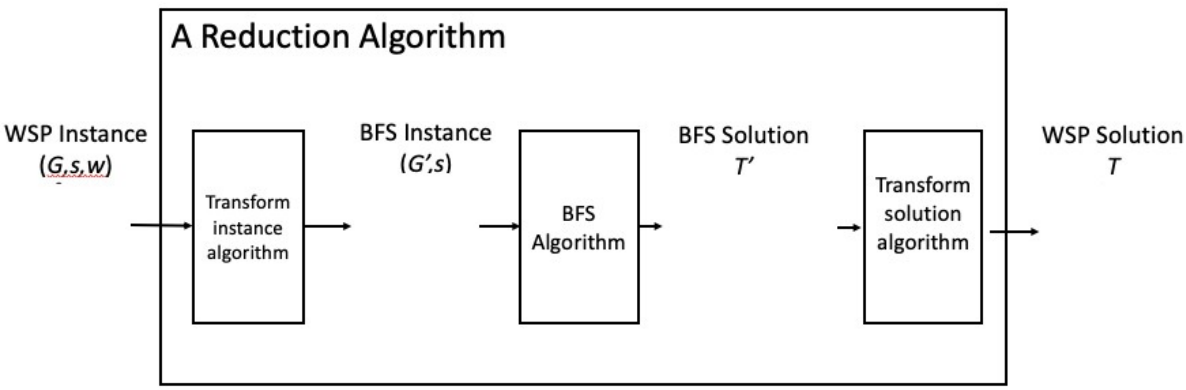
Here we'll explore a way to solve WSP by "reducing" it to BFS. For this, we'll assume that each weight is a positive integer.

1. How can we transform (i.e., "reduce") our WSP example instance into an *unweighted* bfs input, while preserving distances? **Fill in the corresponding bfs instance on the right.**





2. Now, **describe the more general "Transform instance algorithm" and "Transform solution algorithm" in the picture below, for arbitrary instances of WSP.**



Transform instance algorithm:

→ Transform instance algorithm:

for each edge $e \in (u,v)$ of G, with weight $w(u,v)$. add $w-1$ intermediate nodes.

remove edge (u,v)

add edges $(u, i_{e,1}), (i_{e,1}, i_{e,2}), (i_{e,2}, i_{e,3}), \cdots, (i_{e,w-1}, v)$

return the resulting graph(G', s)

$P'[i]$

Transform solution algorithm:

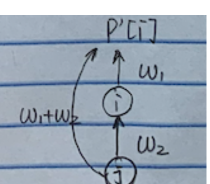
→ Transform solution algorithm: $(T', P'[1, \cdots])$

While T' has intermediate nodes

Let i be such a node, let j be i's child

set $P'[j] = P'[i]$; $P'[i] = \text{null}$

$w(P'[j], j) = w(P'[i], i) + w(i, j)$



3. Explain why the reduction algorithm is correct.

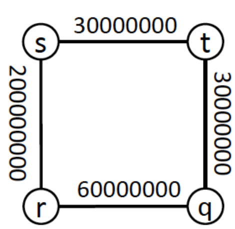
(i) The transform instance algorithm presents path costs between nodes of V.

(ii) the transform solution algorithm also preserves path costs between nodes u,v of V.

⇒ If the shortest path between u,v in G has cost C, then the cost is also C in G′ and cost is C in T′ ⇒ cost C in T, by (ii).

4. What can you say about the running time of the reduction algorithm?

(a) How many nodes does the graph G^t have, for the following WSP instance?



(140,000,000 - 3) nodes

(b) More generally, how many nodes does the graph G^t have? See if you can bound it as a function of n, the number of nodes of G, and the sum $W = {}_{(u,v)\in E}\, w(u, v)$ of the edge weights.

$\Theta(W) = \Theta(W + |V|)$

Step 5: Design a better algorithm.

1. Does bfs suggest any “greedy” strategies to building shortest path trees for weighted graphs, adding one node of G at a time? **Sketch out some ideas. Don’t concern yourselves too much about implementation details; think more about correctness.**

Use Dijkstra's algorithm

Step 5.

Try 1: Start with tree just containing s.

Add nodes one at a time, always using the cheapest possible edge.

Try 2: Add node one at a time, always choose the node with shortest possible path to s

2. Once you have a correct algorithm, fine-tune it to obtain a good runtime.