Multiple Source Shortest Path with unit weights

1 Introduction

Given: Let G be a directed graph $(V, \vec{E})$, embedded on a surface with genus g. All edge weights are unit.
Find: Consider boundary f of G. $\forall v \in f$, find a shortest path to $\forall u \in V$.

Let T be the BFS (Breadth first search) tree of G, and C be the BFS co-tree in G. Then there is exactly $2g$ leftover edges $L = \{e_1, e_2, \ldots, e_{2g}\}$.

There exists a unique cycle λ_i in $C \cup e_i$, and $(\lambda_1, \lambda_2, \ldots, \lambda_{2g}) = \Lambda$ defining homology basis.
We define homological signature of an edge as follows:

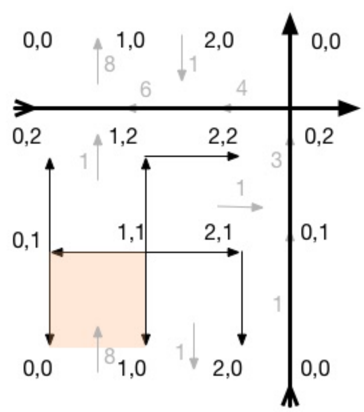
$$[e]_i = \begin{cases} 1 & \text{, if } e \in \lambda_i \\ -1 & \text{, if } rev(r) \in \lambda_i \\ 0 & \text{, otherwise} \end{cases}$$

Furthermore, we define leafmost term α recursively as follows:

$$\alpha(\vec{e}^*) = \begin{cases} 1 & \text{, if } rev(e^*) \text{ is a leaf dart in } C \\ \sum\limits_{\text{tail}(\vec{e}'^*)=\text{head}(\vec{e}^*)} \alpha(\vec{e}') & \text{, otherwise} \end{cases}$$

We can extend above definition with $\alpha(\vec{e}) = \alpha(\vec{e}^*)$ and $\alpha(e)^* = -\alpha(\text{rev}(\vec{e}^*))$.

Let $\tilde{w}(\vec{e}) = \langle 1, [\vec{e}], \alpha(\vec{e}) \rangle$ be new weight vector for each edge in G.



Def: An edge $\vec{e}$ is "holier" than $\vec{e}'$, if $\tilde{w}(\vec{e}) < \tilde{w}(\vec{e}')$ in lexicographic comparison. Therefore, we can define "holiness" of any $S \subset G$ as follows:

$$\text{Ho}(S) = \sum_{\vec{e} \in S} \tilde{w}(\vec{e})$$

Holiest tree is a spanning tree with minimal "holiness". We build Holiest tree rooted at r, using slight tweak in the Bellman-Ford algorithm for finding shortest path tree rooted at r.

BuildHoliestTree$(G, \tilde{w}, r)$:
  Set $dist[r] \leftarrow \langle 0, [\vec{0}], 0 \rangle$
    pred$(r) \leftarrow$ NULL
  for all $v : v \neq r$
    $dist[r] \leftarrow \langle \infty, [\infty], \infty \rangle$
    pred$(r) \leftarrow$ NULL
  put r into queue
  while queue is not empty:
    Let $u \leftarrow$ dequeue item
    for all $u \rightarrow v$
      if v is not marked
        mark v and put in the queue
      if isTense$(u \rightarrow v)$
        relax$(u \rightarrow v)$

isTense$(u \rightarrow v)$:
  return $dist[u] + \tilde{w}(u \rightarrow v) < dist[v]$

relax$(u \rightarrow v)$:
  $dist[v] \leftarrow dist[u] + \tilde{w}(u \rightarrow v)$
  pred[v] $\leftarrow u$

Observation: Each vertex will be added once to the queue.

Corollary: Each edge will be relaxed at most once.

Lemma-1: If there is no tense edge in G, then for each $v : r \rightarrow \ldots \rightarrow \text{pred}(\text{pred}(v)) \rightarrow \text{pred}(v) \rightarrow v$ is the holiest path from r to v.

Proof: Let's prove it by induction on $dist[v][0]$ distance from the root r.
Base: $dist[v].length = 0$, then v = r, so the claim holds trivially.
Induction Step: Suppose the claim is true for all vertex $v \in V$ such that $dist[v].length < d$ for some d. Consider vertex v such that $dist[v].length = d$. By induction hypothesis, all vertices with $dist[u].length = d - 1$ have "holiest" path correctly updated. By definition, $dist[v] = \min_{u \rightarrow v} dist[u] + \tilde{w}(u \rightarrow v)$, here $dist[u].length = d - 1$. By Induction hypothesis, $dist[u]$ is not tense and can construct "holiest" path to u, so if there is no tense edge in G then $dist[v] = \min\{dist[u] + \tilde{w}(u \rightarrow v)\}$ holds. □

Corollary: The algorithm will produce "holiest" tree rooted at r in linear time.
We now have produced our initial "Holiest" tree.

2 Moving Along an Edge

Consider a single edge uv, which is on the boundary face f of G. Suppose we already computed the holy-tree T_u rooted at u. We transform T_u into the holy-tree T_v as follows. First, we insert a new vertex s in the interior of the uv, bisecting it into two edges su and sv with weights:

$$w_0(s \rightarrow u) = \langle 0, [\vec{0}], 0 \rangle$$

$$w_0(s \rightarrow v) = \langle 1, [w(u \rightarrow v)], \alpha(w(u \rightarrow v)) \rangle = w(u \rightarrow v)$$

Observe that this condition implies $s = u$, therefore $T_s = T_u$. We reduce distances to u and v as follows:

$$w_\epsilon(s \rightarrow u) = \langle 0, -[w(u \rightarrow v)], -\alpha(w(u \rightarrow v)) \rangle$$
$$w_\epsilon(s \rightarrow v) = \langle 1, [\vec{0}], 0 \rangle$$

Since we reduced distance to all vertices in the graph equally, the process does not introduce any pivots. Then we define a parametric weights as follows:

$$w_\lambda(s \rightarrow u) = \langle 0, -[w(u \rightarrow v)], -\alpha(w(u \rightarrow v)) \rangle - \lambda$$

$$w_\lambda(s \to v) = \langle 1, [\vec{0}], 0\rangle + \lambda$$

Every other dart $x \to y$ has constant parametric weight $w_\lambda(x \to y) = w(x \to y)$. We then maintain the holy tree T_λ rooted at s, with respect to the weight function w_λ, as λ increases continuously from 0 to $\langle 1, [\vec{0}], 0\rangle$. When $\lambda = w(u \to v)$, $T_\lambda = T_v$.

In the following algorithm, **pred** defines Holy tree rooted at u, and **dist** is corresponding distance to each vertex in the graph.

MoveAlongEdge$(G, u \to v, dist, pred)$:
  Add new vertex s
  $pred[u], pred[v] \leftarrow s$
  $\lambda \leftarrow 0$

  $w(s \to u) \leftarrow \langle 0, -[w(s \to u)], -\alpha(w(s \to u))\rangle$
  AddSubtree$(\langle 0, -[w(s \to u)], -\alpha(w(s \to u))\rangle, u)$

  $w(s \to v) \leftarrow \langle 1, [\vec{0}], 0\rangle$
  AddSubtree$(\langle 0, -[w(s \to u)], -\alpha(w(s \to u))\rangle, v)$

  while $\lambda < \langle 1, [\vec{0}], 0\rangle$:
    pivot $\leftarrow$ FindNextPivot
    If **pivot** is non NULL **AND** $(\lambda + slack(\textbf{pivot})/2) < \langle 1, [\vec{0}], 0\rangle$
      Pivot(**pivot**)
      $\lambda \leftarrow \lambda + slack(\textbf{pivot})/2$
    else
      $\delta = \langle 1, [\vec{0}], 0\rangle - \lambda$
      AddSubtree(δ, u)
      AddSubtree$(-\delta, v)$
      $\lambda \leftarrow \lambda + \delta$