

Conflict Resolution psuedocode

FUNCTION main():

Allocate arrays

(minl,minv,meta,my_grp,order1,order2,
leaves,groupn,branch_minl,branch_minv,
MST1,MST2,weights)

Start timer

Call MST(MST1,MST2,0)

Stop timer

Print edge count and time

Write MST file

Free all memory

FUNCTION get_ready():

Read input parameters from input_params.json

Sanitize counts (bounds, connectivity, completeness, regular adjustment)

Open graph file; abort if missing

Read first header line:

for each node read (edge_count, zero_count) -> allocate compressed row

For each subsequent line (node i):

For each (j,weight):

Normalize weight > 0

Append either weight (if consecutive) or negative jump marker then weight

Mirror insertion for node j (undirected)

Close file; return success

DATA (compressed row format):

Sequence of ints; >0 = edge weight;

<0 = “next neighbor index” marker (skip range of zeros).

This saves space if possible else does nothing.

It is like eliminating and combining zeroes.

FUNCTION MST(MST1,MST2,id):

Phase 1 (initial minima):

Get the minima of each node

For each node u:

Scan its row; track smallest weight -> (minl[u],minv[u]);
map order1[u]=minl[u], order2[minl[u]]=u; my_grp[u]=u

Phase 2 (pair conflicts + branchs):

Get the conflicts and the branches

For each node u:

If no min edge -> continue

If mutual (**minl[minl[u]]==u**) and **minl[u]<u**:

Mark u group head (meta[u]=0),

Record edge (u,partner); register conflict group head; union partner under u

Else:

Add edge (u,minl[u]); union groups

If conflict <=1 return, got MST

Phase 3 (build chains):

Build a path to travel for next steps

For each leaf:

Follow minl links while order2[next]==current and not at head;

set order2 for chain nodes to leaf;

inherit group from head

order1 is the path of a branch.

order2 is the teller of end points.

Phase 4 (recompute minima excluding intra-group):

Now we have groups

For each node u:

Find representative my_grp[u]

Rescan row:

Skip used (weight==0), self (index==u),

intra-group edges (mark weight=0)

Keep smallest external edge -> (minl[u],minv[u])

Adjust chain linkage edge cases

Phase 5 (iterate till no conflict):

Now we just repeat 4 above

While conflict>1:

For each leaf branch:

If branch_minv cached >0 skip else scan its chain to pick node with smallest minv (whose minl!=-1);
store branch_minl/branch_minv

For each conflict head:

Among its leaves pick branch with smallest branch_minv;
store choice in meta[head]

Reset conflict count:

For each prior head:

Let x=meta[head]; y=minl[x];
if invalid or same group -> continue

If mutual conflict with other head and head id tie-break wins:

Merge groups, add edge, **record new conflict** head

Else:

Merge groups, add edge

If conflict<=1 return

For each leaf:

If its chosen branch edge now intra-group or consumed:

Invalidate branch_minv

Walk chain: recompute minima excluding intra-group exactly like Phase 4 (mark removed edges weight=0)

End

OUTPUT:

Console: counts, time. File: MST edge list.