Fork-Fulkerson

pseudo code:

Maxflow getMaximumFlow(graph){

mf = new maxflow (graph);

Subgraph path = new subgraph;

double min;

while(path = findpath (graph,mf,source,targe) != null){

min = findmin(graph,path);

mf.updateresidual;

mf.updateresidualbackward;}

return mf;}

findpath(graph,mf,source,target){

if(source == target) return path;

tmp;

incomingedges = target.getincomingedges

for(edges:incomingedges){

if(path.contains(edge.getsource()))) continue;

if(mf.getResidual(edge) <=0) continue;

tmp = path

tmp.addEdge(edge);

tmp = getpath(graph, tmp ,source,edge.getsource())

if (tmp != null) return tmp; }

return null;}

getmin

double min

min = getResidual(path.get(0));

for(int i=1; i<= path.size();i++){

min = math.min(min,getResidual(path(i)))}

return min

updateResidualEdges

for(edges : path){

updateedge(edge, min)}

maxflow += min

updateResidualEdge(edge,min)

prev = getflow(edge);

if(prev == null) prev == 0;

edge.put(prev +min);

updatebackward

for(edge: path.getedges()){

for(redge: graph.getincomingedge(edge.getsource))

if(redge.getsource() == edge.gettarge())

{updateResidual(edge,-min)break;}}

A\* search

priorityqueue queue(distancepair)

int[] prev

double [] distance

set visited

init (prev,distance,target)

queue.add(new DP(target, 0))

while (! queue.isEmpty()){

DP u = queue.poll();

visited.add(u.vertex)

incomingedges = getIncomingedges(u.vertex)

for(edge: incomingedges){

if(!visited.contains(edge.getsource()))

dist = distance[u.vertex] + edge.weight();

if(dist < distance[edge.getsource])

distance[edge.getsource] = dist + herioha

prev [edge.getsource] = u.vertex

queue.add(new DP(edgegetsource,dist))}

Cycle

function getcycle(){

Deque unvisited

for (int i = 0; i< = graph.size();i++)

unvisited.add(i)

While(!unvisited.isEmpty()){

v=unvisited.poll()

if(getcycleAux(v)) return true}

return false;}

function getcycleAux(v){

unvisited.remove(v)

visited.add(v)

incomingedges = v.getincomingedges

for(edge:incomingedges){

if(visited.contains(edge.getsource))

return true;

if(getcycleAux(edge.getsource))

return true;}

}

Function topologicalsort(graph){

Deque<Integer> global = getverticeswithnoincomingedge(graph)

Deque,<edge>[] alloutgoingedge = getalloutgoingedges(graph)

List<Integer> order

while(!= global.isEmpty()){

local;

v = global.poll()

order.add(v)

List<edges> outgoingedges = alloutgoingedges.get(v)

while(outgoingedges.isEmpty){

edge = outgoingEdges.poll()

u = edge.gettarget()

List<edge>incomingedge = u.getincomingedge}

incomingedge.remove(edge)

if(incomingedge.isempty) local.add(u)}

while (!local.isempty){

global.addFirst.(local.removeLast()) //depth first}

}

Chu-Liu-Edmonds'

forest = inti forest;

while(true){

true = getMinimumIncomingEdges()

cycles = tree.getcycles

if(cycles.isEmpty){

addEdgesFromCycles(tree,cyclisEdges)

break;}

else forest = updateWeights

addAll(cyclicEdges,cycles)

}

Function getMinimumIncomingEdges()

for(subtree;forsest){

edge min = null

for(vertex:subtree){

for(edges: vertex.getincomingedge){

if(!subtree.contains(edge.getsource())&&(min == null || min.getweight() > edge.getweight()))}

min = edge;}

if(min!= null)

tree..addedge(min)}

Function updateEdgesweight{

for(cycle : cycles){

set = getvertices(cycle);

all.addall(set);

forest.add(set);

for(edge:cycle)

{

for(iedge : graph.getincomingedgess(edge.getsource)){if(!set.contains(iedges.getsoure))

iedge.addWeight(getChainedWeight(cycle,edge.getsource))}}

}

for(int i = 0 i <size i++)

for(!all.contains(i)){

new set add (i)

forest .add (newset)}

return forest;}

getchianweight{

sum = 0

for(edge:cycle){

edge.gettarget != source

sum += edge.weight}

return sum}

addEdgesfromcycles

{ targets = tree,gettartges();

for(edge : cyclicEdges){

if(!targets.contain(edge.getTarget)

tree.addEdge(edge)}

}

Prim

priorityqueue<edge> queue = new priorityqueue,.()

queue.add(queue,visited,graph,0)

while(!queue.isEmpty()){

edge = queue.poll

if(!visitied.contains(edge.getsource))

tree.add(edge)

if(tree.size+1 == graph.size()) break;

visited.add(target)

queue.addAll(graph.getincomingedge(edge))

}