```
class HardSubStructureFinder {
private:
    double max_subjet_mass, mass_drop_threshold, Rfilt, minpt_subjet;
    double mhmax, mhmin, mh; double zcut, rcut_factor;
    size_t nfilt;
    inline void find_structures (const fastjet::PseudoJet & this_jet) {
        fastjet::PseudoJet parent1(0,0,0,0), parent2(0,0,0,0);
        bool haskid=this_jet.validated_cs()->has_parents(this_jet,parent1,parent2);
        if(haskid) {
            if (parent1.m()<parent2.m()) {std::swap(parent1,parent2);}
            double kidmass    = parent1.m()  + parent2.m() ;
            double parentmass = this_jet.m()               ;
            if(kidmass<parentmass*mass_drop_threshold){
                t_parts.push_back(parent1);
                t_parts.push_back(parent2);
                return;
            } else if (parent1.m()>mass_drop_threshold*parent2.m()) {find_structures(parent1);return;}
        } else {return;}
        if(this_jet.m()<max_subjet_mass){t_parts.push_back(this_jet);}
        else {
            fastjet::PseudoJet parent1(0,0,0,0), parent2(0,0,0,0);
            bool haskid = this_jet.validated_cs()->has_parents(this_jet,parent1,parent2);
            if (haskid) {
                if (parent1.m()<parent2.m()) {std::swap(parent1,parent2);}
                find_structures(parent1);
                if (parent1.m()<mass_drop_threshold*this_jet.m()) {find_structures(parent2);}
            }
        }
    }
    inline void run (fastjet::PseudoJet&injet) {
        t_parts.clear(); find_structures(injet);
        if(t_parts.size()>1) {
            t_parts=sorted_by_pt(t_parts);
            size_t i=0; size_t j=1;
            triple = fastjet::join (t_parts[i],t_parts[j]) ;
            filt_tau_R = std::min ( Rfilt , 0.5*sqrt(t_parts[i].squared_distance(t_parts[j])) ) ;
            fastjet::JetDefinition filtering_def(fastjet::cambridge_algorithm,filt_tau_R);
            fastjet::Filter filter(filtering_def,fastjet::SelectorNHardest(nfilt)*fastjet::SelectorPtMin(minpt_subjet));
            taucandidate=filter(triple); filteredjetmass=taucandidate.m();
            if((mhmin<filteredjetmass)&&(filteredjetmass<mhmax)&&(taucandidate.pieces().size()>1)){
                fastjet::JetDefinition    reclustering (fastjet::cambridge_algorithm,10.0)   ;
                fastjet::ClusterSequence cs_top_sub    (taucandidate.pieces(),reclustering) ;
                tau_subs=sorted_by_pt(cs_top_sub.exclusive_jets(2));
                if (tau_subs[1].perp()>minpt_subjet) {
                    HiggsTagged=true;
                    Higgs=tau_subs[0]+tau_subs[1];
                    deltah=CPPFileIO::mymod(taucandidate.m()-mh);
                    tau_hadrons=taucandidate.constituents();
                    double Rprun=injet.validated_cluster_sequence()->jet_def().R();
                    fastjet::JetDefinition jet_def_prune(fastjet::cambridge_algorithm, Rprun);
                    fastjet::Pruner pruner(jet_def_prune,zcut,rcut_factor);
                    prunedjet=pruner(triple);
                    prunedmass=prunedjet.m();
                    unfiltered_mass=triple.m();
                }
            }
        }
    }
    inline void initialize () {
        t_parts.clear(); tau_subs.clear(); tau_hadrons.clear();
        max_subjet_mass=30; Rfilt=0.3; minpt_subjet=20;
        mass_drop_threshold=0.7; nfilt=4; filteredjetmass=0.0;
        mh=125.0; mhmax=mh+100.0; mhmin=mh-100.0; filt_tau_R=0;
        zcut=0.1; rcut_factor=0.5; prunedmass=0.0; unfiltered_mass=0.0;
        deltah=10000; HiggsTagged=false;
    }
public:
    double             filteredjetmass , deltah    , filt_tau_R   , prunedmass    , unfiltered_mass ;
    pseudojets         tau_subs        , t_parts   , tau_hadrons ;
    fastjet::PseudoJet prunedjet       , triple    , Higgs        , taucandidate ;
    bool HiggsTagged ;
    inline void operator () () {initialize();}
    inline void operator () (fastjet::PseudoJet&injet) {run(injet);}
    HardSubStructureFinder(){initialize();}
    ~HardSubStructureFinder(){}
};
```