We will assume that no exception was thrown to see the worst-case scenario, we will enter every last "else if" to add more to our complexity count, and we will work with the worst case for the methods called.

**Class Controller Method checkInPatient**

AggravationEnum aggravationPatient = AggravationEnum.NONE; **+1**

StatusPatientEnum statusPatient = StatusPatientEnum.PRIORITY\_GENERAL; **+1**

if(unit==1){ **+1**

statusPatient = StatusPatientEnum.PRIORITY\_HEMATOLOGY; **+1**

}else{

statusPatient = StatusPatientEnum.PRIORITY\_GENERAL;

}

int priority= 1; **+1**

//Array of states

//Array of aggravations

AggravationEnum [] aggravations = {AggravationEnum.OLD\_AGE, AggravationEnum.PREGNANT, AggravationEnum.CHILD, AggravationEnum.NONE}; **+1**

aggravationPatient = aggravations[aggravation-1]; **+1**

if(aggravation == 1 || aggravation==3){ **+1**

priority+=1;

}

else if (aggravation == 2 ){ **+1**

priority+=2; **+1**

}

Patient patient= new Patient(name, date, causeOfAdmission, priority, aggravationPatient, statusPatient, id); **+1**

PatientNode <Patient> patientNode= new PatientNode<>(priority, patient, id); **+1**

hash.chainedHashInsert(patientNode); **+n**

allPatients.add(patient); **+n**

try {

if (unit==1){ **+1**

priorityHematology.Insert(patientNode);

}else if(unit == 2){ **+1**

priorityGeneral.Insert(patientNode); }**+1**

hash.chainedHashInsert(patientNode); **+n**

addToStack(new PatientStackNode(patientNode,unit)); **+n**

}catch (Exception e) {

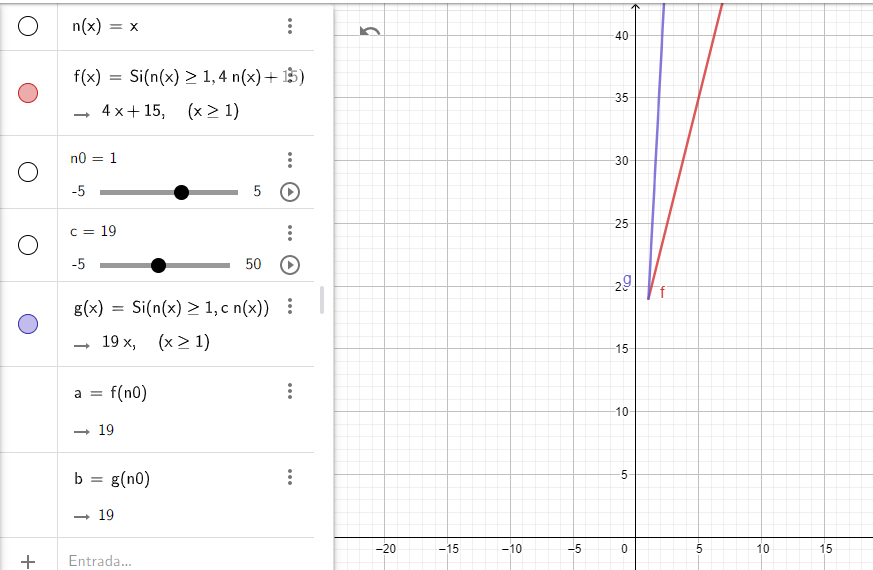
System.out.println(e.getMessage());

}

**Total: 4n + 15**

**and we can prove it with and**

**See the purple function:**

****

**Class Controller Method printPatients**

Heap tempGeneral= new Heap(); **+1**

Heap tempHematology= new Heap(); **+1**

for(int i =0; i < priorityGeneral.getHeapSize();i++){ **+ (n+1)**

try {

tempGeneral.Insert(priorityGeneral.getArray().get(i)); **+**

}catch (Exception e){}

}

for(int i =0; i < priorityHematology.getHeapSize();i++){ **+ (n+1)**

try {

tempHematology.Insert(priorityHematology.getArray().get(i)); **+**

}catch (Exception e){}

}

tempHematology.BuildHeap(); **+**

tempGeneral.BuildHeap(); **+**

int general= priorityGeneral.getHeapSize(); **+1**

int hematology= priorityHematology.getHeapSize(); **+1**

String out="PATIENTS IN THE CLINIC"; **+1**

out+="\n Patients in Hematology: (" + hematology +")"; **+1**

for(int i=0; i<hematology; i++){ **+ (n+1)**

try {

PatientNode aux=tempHematology.ExtractMax(); **+**

out += "\n ("+ (i+1)+") "+aux.getNamePatient() + " with id: " +((Patient)aux.getPatient()).getId() ; **+n**

}catch (Exception e){ }

}

out+="\n Patients in General attention: (" + general +")" ; **+1**

for(int i=0; i<general; i++){**+ (n+1)**

try {

PatientNode aux=tempHematology.ExtractMax(); **+**

out += "\n (" + (i + 1) + ")" +aux.getNamePatient() + " with id: " +((Patient)aux.getPatient()).getId() ; **+n**

}catch (Exception e){ }

}

out+="\n Patients available for check out: (" + checkOut.size() +")"; **+1**

for(PatientNode p: checkOut){ **+ (n+1)**

out+="\n "+ ((Patient)p.getPatient()).getName() + " with id: " + ((Patient)p.getPatient()).getId(); **+n**

}

tempGeneral= priorityGeneral; **+1**

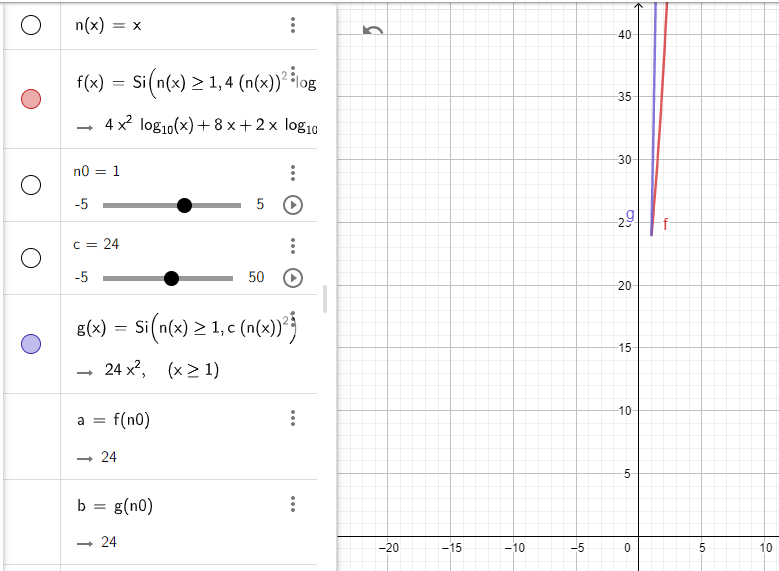
tempHematology= priorityHematology; **+1**

return out; **+1**

**Total:**

**and we can prove it with and**

**See the purple function:**

****