## Ahmet EROL

## 1814094

# EE441, HW #2

### Create Flight

The creation of flight is a simple object constructor. The constructor function is as below.

EEAircraft::EEAircraft(int direction\_,float latitude\_,float longitude\_) {

direction=direction\_;

latitude=latitude\_;

longitude=longitude\_;

first=NULL;

last=NULL;

}

It can be easily seen that time and space complexity of creation is O(1).

### Register Passengers

The integrate method function implement the addNewCapsule function recursively. This method adds new capsules in sorted order starting from maximum longitude. In the worst case scenario, the new capsule will be added end of list. Then the complexity is can be expressed as O(N).

EECapsule\* addNewCapsule(EECapsule\* currentCapsule,EECapsule\* newCapsule,const EECapsule\* first) {

EECapsule\* nextCapsule;

nextCapsule=currentCapsule->getNext();

float currentCapsuleLongitude=currentCapsule->getLongitude();

float nextCapsuleLongitude=nextCapsule->getLongitude();

float newCapsuleLongitude=newCapsule->getLongitude();

if(newCapsuleLongitude>=currentCapsuleLongitude) {

// new capsule longitude is greater than current one

if(nextCapsuleLongitude<=currentCapsuleLongitude) {

// current capsule is at top position

// add new capsule here

return addAfter(currentCapsule,newCapsule);

}

else {

// current capsule is not at top

if(newCapsuleLongitude<=nextCapsuleLongitude) {

// new capsule is between next and current

return addAfter(currentCapsule,newCapsule);

}

else {

addNewCapsule(currentCapsule->getNext(),newCapsule,first);

}

}

}

else {

if(currentCapsule->getPrev()==first) {

// bottom capsule

return addAfter(currentCapsule->getPrev(),newCapsule);

}

else {

// new capsule longitude is smaller than current one, add before

addNewCapsule(currentCapsule->getPrev(),newCapsule,first);

}

}

}

### Simulate Flight

In the simulation of flight the following code segment is used.

if(flight.hasCapsule()) {

cout<<"The registration has been closed and the aircraft to take off is:"<<endl;

flight.setStarting();

flight.showInfo();

while(flight.hasCapsule()) {

EECapsule\* dropped;

dropped = flight.drop();

dropped->showInfo(flight.max\_string);

}

}

Firstly setStarting() method is used. This method uses findAircraft function recursively.

EECapsule\* findAircraft(EECapsule\* currentCapsule,const float longitude,const EECapsule\* first) {

float currentCapsuleLongitude=currentCapsule->getLongitude();

float nextCapsuleLongitude=currentCapsule->getNext()->getLongitude();

float prevCapsuleLongitude=currentCapsule->getPrev()->getLongitude();

// we reached end point

if(currentCapsule==first) return currentCapsule;

else {

// contunie

if(currentCapsule->getLongitude()>=longitude) return currentCapsule;

else return findAircraft(currentCapsule->getNext(),longitude,first);

}

}

In the worst case scenario, the space and time complexity of this function can be expressed as O(N).

Since the while loop will run until the flight has no capsule, the complexity of while part again can be expressed as O(N).

Then total complexity can be expressed as O(N+N)=O(2N)=O(N)