ESINF

RELATÓRIO SPRINT 2

João Teixeira (1210957)
Jonas Antunes (1181478)
José Rente (1211155)
Marco Maia (1210951)
Ruben Ferreira (1210954)
2DD – G31

Índice

1 *US307: O(l*c)* 3

2 *US308: O(n^4)* 4

3 *US309: O(b*o*V*n)* 6

4 *US310: O(n^3)* 7

5 *US311: O(h*V*E) ~ O(V^4)* 11

Diagrama de Classes 1 2

Método 1: loadResources() [O(l*c)] em CSVLoaderHandler 3
Método 2: parse() [O(l*p)] em BundleParser [1] 3
Método 3: parse() [O(l*p)] em BundleParser [2] 4
Método 4: expBasketsListDay() [O(n^3)] em ExpBasketListHandler 4
Método 5: computeBundle() [O(n^2)] em ExpBasketListHandler 4
Método 6: selectProducer() [O(n)] em ExpBasketListHandler 5
Método 7: expBasketsList() [O(n^4)] em ExpBasketsListHandler 5
Método 8: expListNProducersDay() [O(b*o*V*n)] em ExpListNProducersHandler 6
Método 9: computeBundle() [O(o*V*n)] em ExpListNProducersHandler 6
Método 10: selectProducerForOrder() [O(V*n)] em ExpListNProducersHandler 6
Método 11: getNearestProducersToHub() [O(V)] em ExpListNProducersHandler 7
Método 12: expListNProducers() [O(b)] em ExpListNProducersHandler 7
Método 13: getAllbundlesStats() [O(n^2)] em ExpListStatsHandler 7
Método 14: statsEachBundle() [O(n)] em ExpListStatsHandler 8
Método 15: getAllClientsStats() [O(n^2)] em ExpListStatsHandler 8
Método 16: getAllProducerssStats() [O(n^3)] em ExpListStatsHandler 8
Método 17: clientStats() [O(n)] em ExpListStatsHandler 9
Método 18: producerStockStats() [O(n)] em ExpListStatsHandler 9
Método 19: producerBundleStats() [O(n^2)] em ExpListStatsHandler [1] 10
Método 20: producerBundleStats() [O(n^2)] em ExpListStatsHandler [2] 10
Método 21: getAllHubStats() [O(n^2)] em ExpListStatsHandler 11
Método 22: shortestRoute() [O(h*V*E) ~ O(V^4)] em ShortestPathHandler 11
Método 23: componentTSP() [O(V*E)] em TSP 12
Método 24: fromComponents() [O(h*V*E)] em TSP 13
Método 25: getDists() [O(V)] em TSP 13
Método 26: mergeRoutes() [O(h*V)] em TSP 14
Método 27: mstPrim() [O(V*E)] em MetricTSP 14
Método 28: mstPrimImpl() [O(V*E)] em MetricTSP 15
Método 29: mstBuild() [O(V)] em MetricTSP 15
Método 30: twosApproximation() [O(V*E)] em MetricTSP 16

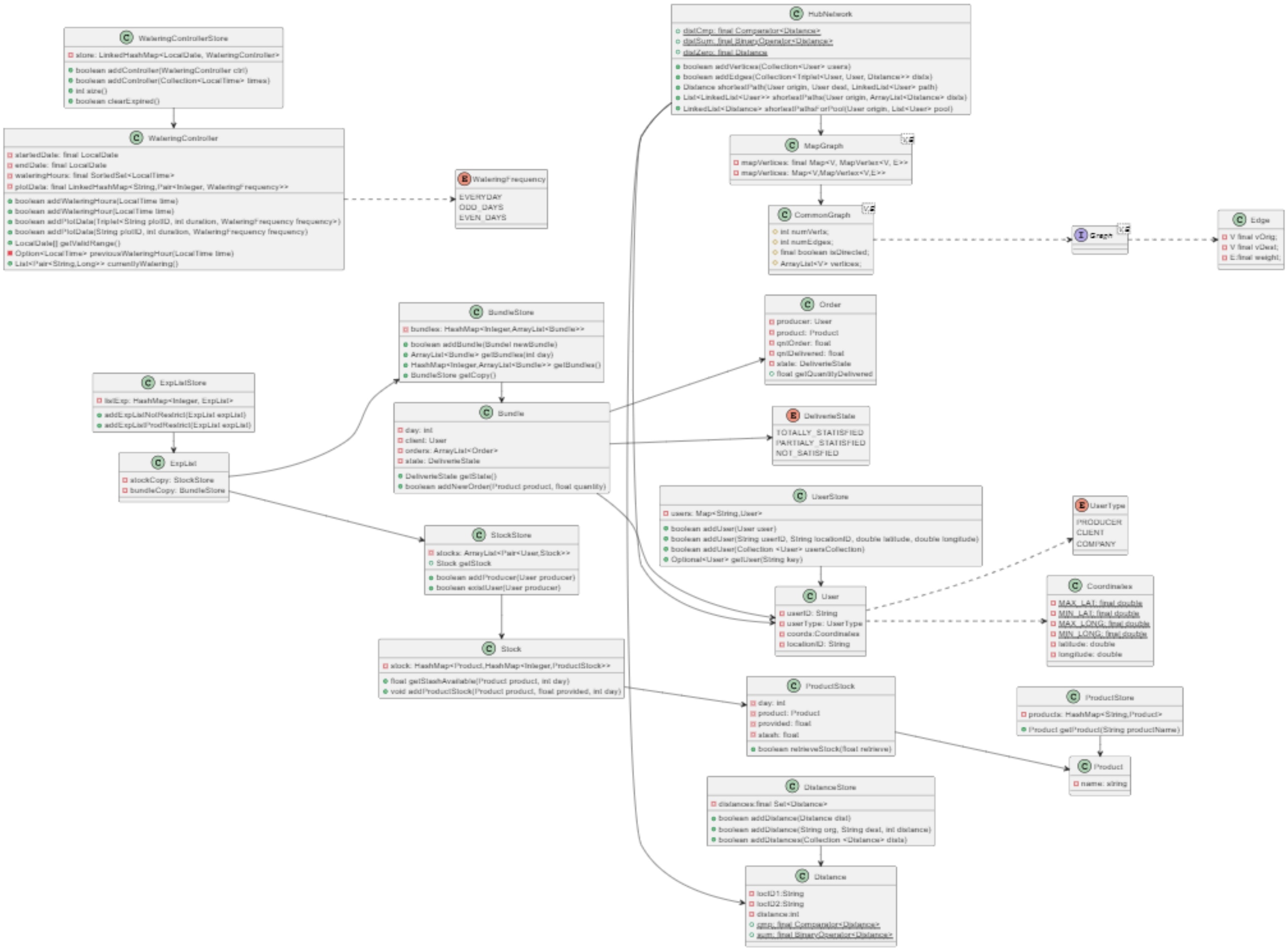
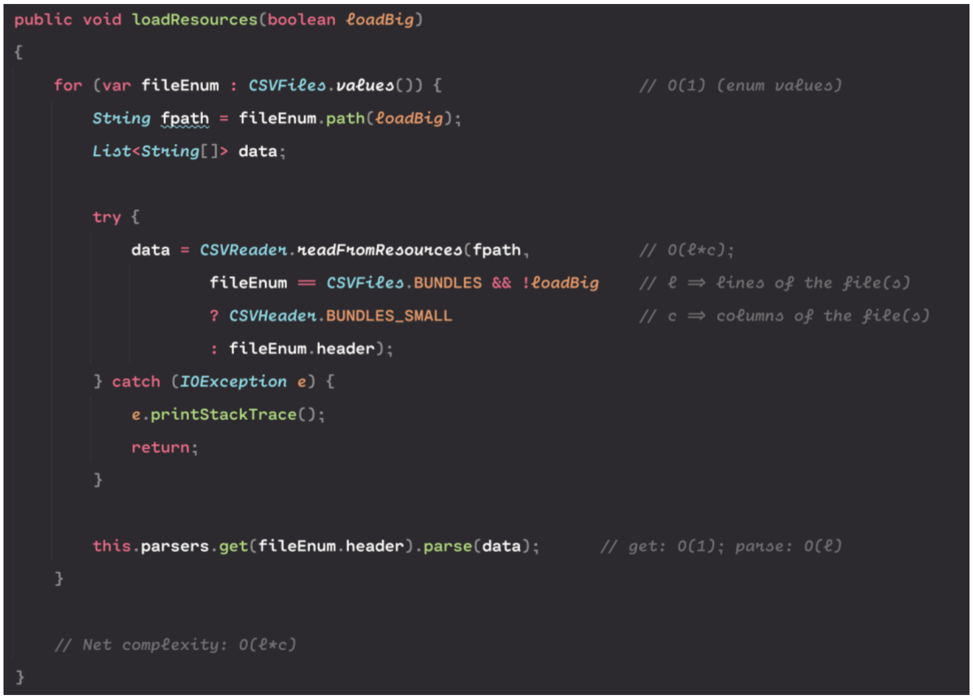

WateringControllerStore
store: LinkedHashMap<LocalDate, WateringController>
boolean addController(WateringController ctrl)
boolean addController(Collection<LocalTime> times)
int size()
boolean clearExpired()
WateringController
startedDate: final LocalDate
endDate: final LocalDate
wateringHours: final SortedSet<LocalTime>
plotData: final LinkedHashMap<String,Pair<Integer, WateringFrequency>>
boolean addWateringHours(LocalTime time)
boolean addWateringHour(LocalTime time)
boolean addPlotData(Triplet<String plotID, int duration, WateringFrequency frequency>)
boolean addPlotData(String plotID, int duration, WateringFrequency frequency)
LocalDate[] getValidRange()
Option<LocalTime> previousWateringHour(LocalTime time)
List<Pair<String,Long>> currentlyWatering()
WateringFrequency
EVERYDAY
ODD_DAYS
EVEN_DAYS
HubNetwork
distCmp: final Comparator<Distance>
distSum: final BinaryOperator<Distance>
distZero: final Distance
boolean addVertices(Collection<User> users)
boolean addEdges(Collection<Triplet<User, User, Distance>> dists)
Distance shortestPath(User origin, User dest, LinkedList<User> path)
List<LinkedList<User>> shortestPaths(User origin, ArrayList<Distance> dists)
LinkedList<Distance> shortestPathsForPool(User origin, List<User> pool)
MapGraph
V,E
mapVertices: final Map<V, MapVertex<V, E>>
mapVertices: Map<V,MapVertex<V,E>>
CommonGraph
V,E
int numVerts;
int numEdges;
final boolean isDirected;
ArrayList<V> vertices;
Graph
V,E
Edge
V final vOrig;
V final vDest;
E:final weight;
BundleStore
bundles: HashMap<Integer,ArrayList<Bundle>>
boolean addBundle(Bundel newBundle)
ArrayList<Bundle> getBundles(int day)
HashMap<Integer,ArrayList<Bundle>> getBundles()
BundleStore getCopy()
Order
producer: User
product: Product
qntOrder: float
qntDelivered: float
state: DeliverieState
float getQuantityDelivered
ExpListStore
listExp: HashMap<Integer, ExpList>
addExpListNotRestrict(ExpList expList)
addExpListProdRestrict(ExpList expList)
Bundle
day: int
client: User
orders: ArrayList<Order>
state: DeliverieState
DeliverieState getState()
boolean addNewOrder(Product product, float quantity)
DeliverieState
TOTALLY_STATISFIED
PARTIALY_STATISFIED
NOT_SATISFIED
ExpList
stockCopy: StockStore
bundleCopy: BundleStore
UserStore
users: Map<String,User>
boolean addUser(User user)
boolean addUser(String userID, String locationID, double latitude, double longitude)
boolean addUser(Collection <User> usersCollection)
Optional<User> getUser(String key)
UserType
PRODUCER
CLIENT
COMPANY
StockStore
stocks: ArrayList<Pair<User,Stock>>
Stock getStock
boolean addProducer(User producer)
boolean existUser(User producer)
User
userID: String
userType: UserType
coords:Coordinates
locationID: String
Coordinates
MAX_LAT: final double
MIN_LAT: final double
MAX_LONG: final double
MIN_LONG: final double
latitude: double
longitude: double
Stock
stock: HashMap<Product,HashMap<Integer,ProductStock>>
float getStashAvailable(Product product, int day)
void addProductStock(Product product, float provided, int day)
ProductStock
day: int
product: Product
provided: float
stash: float
boolean retrieveStock(float retrieve)
ProductStore
products: HashMap<String,Product>
Product getProduct(String productName)
Product
name: string
DistanceStore
distances:final Set<Distance>
boolean addDistance(Distance dist)
boolean addDistance(String org, String dest, int distance)
boolean addDistances(Collection <Distance> dists)
Distance
locID1:String
locID2:String
distance:int
cmp: final Comparator<Distance>
sum: final BinaryOperator<Distance>

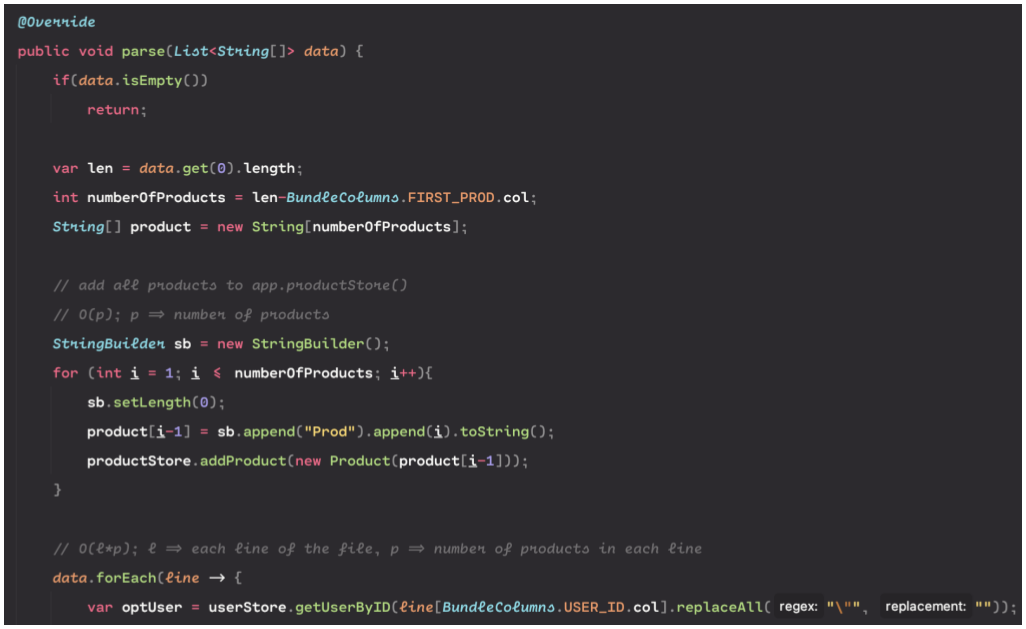
Diagrama de Classes 1

1 US307: O(l*c)



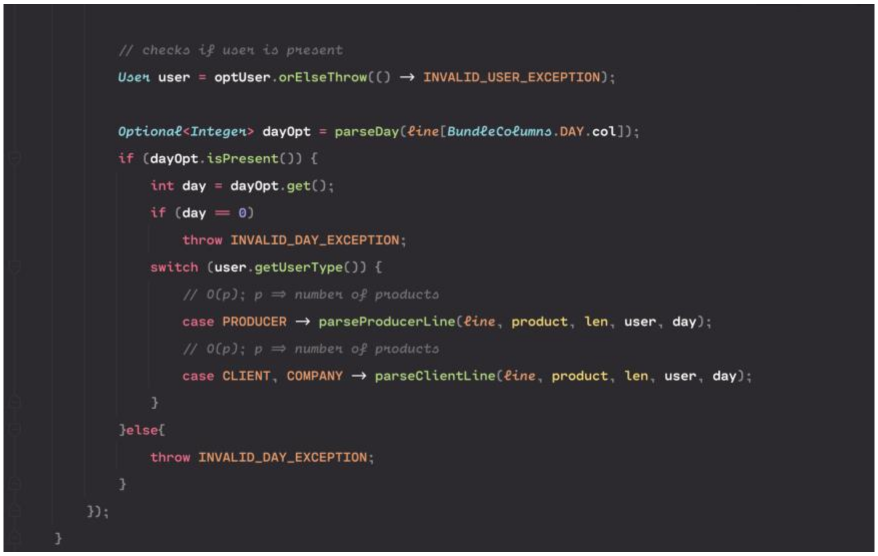
```
public void loadResources(boolean loadBig)
{
    for (var fileEnum : CSVFiles.values()) {                        // O(1) (enum values)
        String fpath = fileEnum.path(loadBig);
        List<String[]> data;

        try {
            data = CSVReader.readFromResources(fpath,               // O(ℓ*c);
                    fileEnum == CSVFiles.BUNDLES && !loadBig        // ℓ ⇒ lines of the file(s)
                    ? CSVHeader.BUNDLES_SMALL                       // c ⇒ columns of the file(s)
                    : fileEnum.header);
        } catch (IOException e) {
            e.printStackTrace();
            return;
        }

        this.parsers.get(fileEnum.header).parse(data);      // get: O(1); parse: O(ℓ)
    }

    // Net complexity: O(ℓ*c)
}
```

*Método 1: loadResources() [O(l*c)] em CSVLoaderHandler*



```
@Override
public void parse(List<String[]> data) {
    if(data.isEmpty())
        return;

    var len = data.get(0).length;
    int numberOfProducts = len-BundleColumns.FIRST_PROD.col;
    String[] product = new String[numberOfProducts];

    // add all products to app.productStore()
    // O(p); p ⇒ number of products
    StringBuilder sb = new StringBuilder();
    for (int i = 1; i ≤ numberOfProducts; i++){
        sb.setLength(0);
        product[i-1] = sb.append("Prod").append(i).toString();
        productStore.addProduct(new Product(product[i-1]));
    }

    // O(ℓ*p); ℓ ⇒ each line of the file, p ⇒ number of products in each line
    data.forEach(line → {
        var optUser = userStore.getUserByID(line[BundleColumns.USER_ID.col].replaceAll( regex: "\"",  replacement: ""));
```

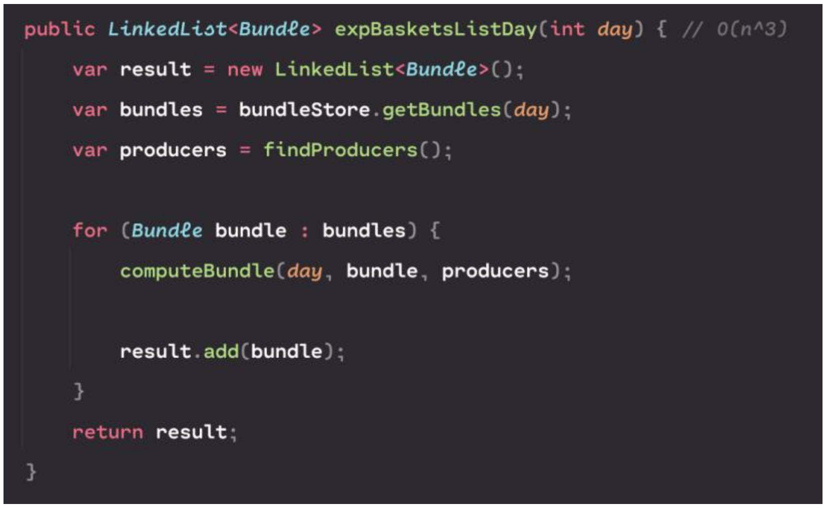
*Método 2: parse() [O(l*p)] em BundleParser [1]*



```java
            // checks if user is present
            User user = optUser.orElseThrow(() -> INVALID_USER_EXCEPTION);

            Optional<Integer> dayOpt = parseDay(line[BundleColumns.DAY.col]);
            if (dayOpt.isPresent()) {
                int day = dayOpt.get();
                if (day == 0)
                    throw INVALID_DAY_EXCEPTION;
                switch (user.getUserType()) {
                    // O(p); p => number of products
                    case PRODUCER -> parseProducerLine(line, product, len, user, day);
                    // O(p); p => number of products
                    case CLIENT, COMPANY -> parseClientLine(line, product, len, user, day);
                }
            }else{
                throw INVALID_DAY_EXCEPTION;
            }
        });
    }
```

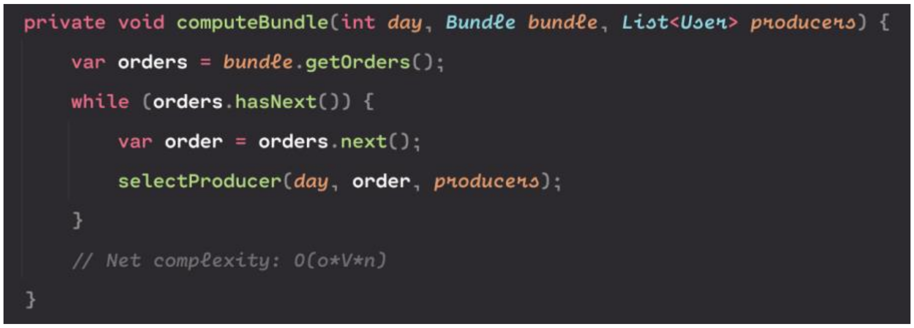
*Método 3: parse() [O(l*p)] em BundleParser [2]*

2 US308: O(n^4)



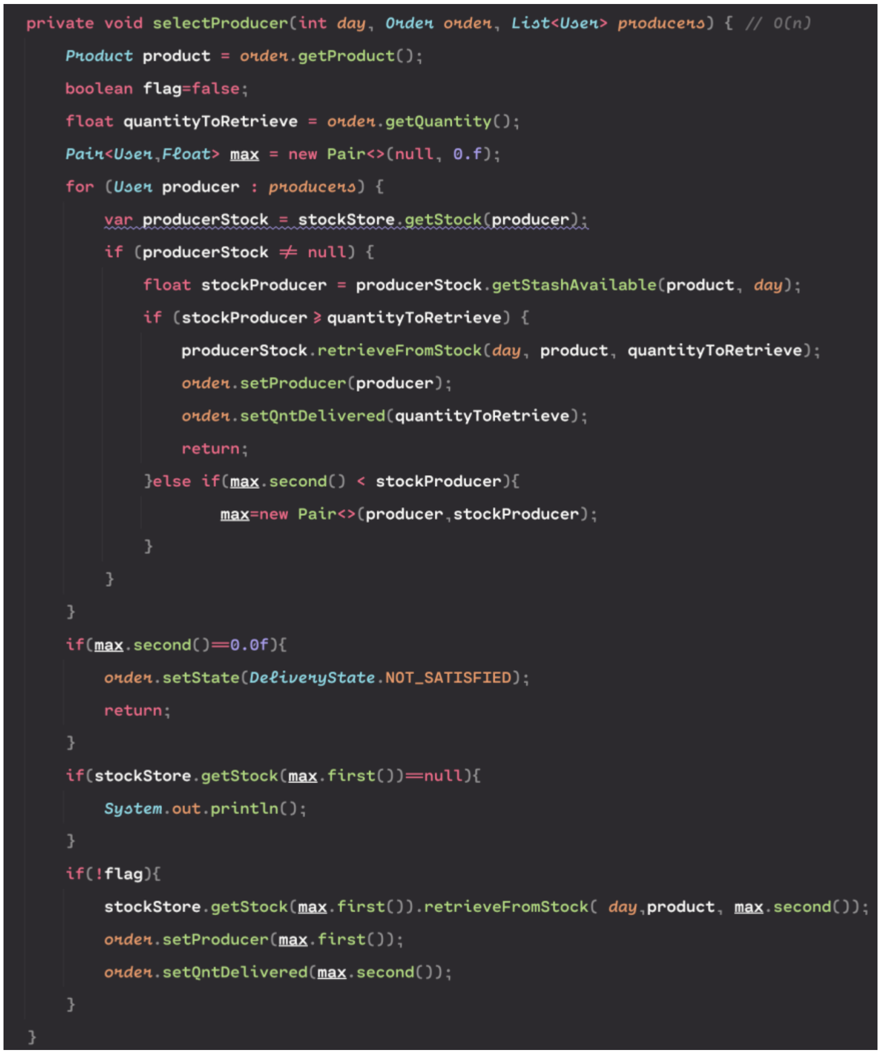
```java
public LinkedList<Bundle> expBasketsListDay(int day) { // O(n^3)
    var result = new LinkedList<Bundle>();
    var bundles = bundleStore.getBundles(day);
    var producers = findProducers();

    for (Bundle bundle : bundles) {
        computeBundle(day, bundle, producers);

        result.add(bundle);
    }
    return result;
}
```

Método 4: expBasketsListDay() [O(n^3)] em ExpBasketListHandler



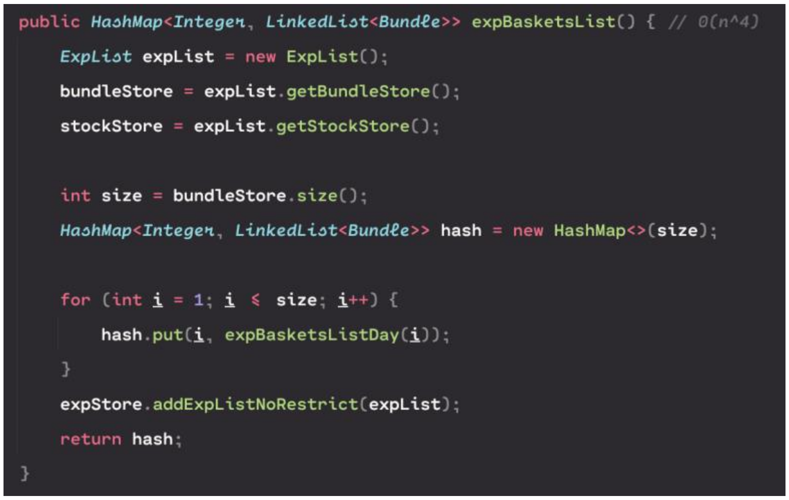
```java
private void computeBundle(int day, Bundle bundle, List<User> producers) {
    var orders = bundle.getOrders();
    while (orders.hasNext()) {
        var order = orders.next();
        selectProducer(day, order, producers);
    }
    // Net complexity: O(o*V*n)
}
```

Método 5: computeBundle() [O(n^2)] em ExpBasketListHandler



```java
private void selectProducer(int day, Order order, List<User> producers) { // O(n)
    Product product = order.getProduct();
    boolean flag=false;
    float quantityToRetrieve = order.getQuantity();
    Pair<User,Float> max = new Pair<>(null, 0.f);
    for (User producer : producers) {
        var producerStock = stockStore.getStock(producer);
        if (producerStock != null) {
            float stockProducer = producerStock.getStashAvailable(product, day);
            if (stockProducer >= quantityToRetrieve) {
                producerStock.retrieveFromStock(day, product, quantityToRetrieve);
                order.setProducer(producer);
                order.setQntDelivered(quantityToRetrieve);
                return;
            }else if(max.second() < stockProducer){
                    max=new Pair<>(producer,stockProducer);
            }
        }
    }
    if(max.second()==0.0f){
        order.setState(DeliveryState.NOT_SATISFIED);
        return;
    }
    if(stockStore.getStock(max.first())==null){
        System.out.println();
    }
    if(!flag){
        stockStore.getStock(max.first()).retrieveFromStock( day,product, max.second());
        order.setProducer(max.first());
        order.setQntDelivered(max.second());
    }
}
```

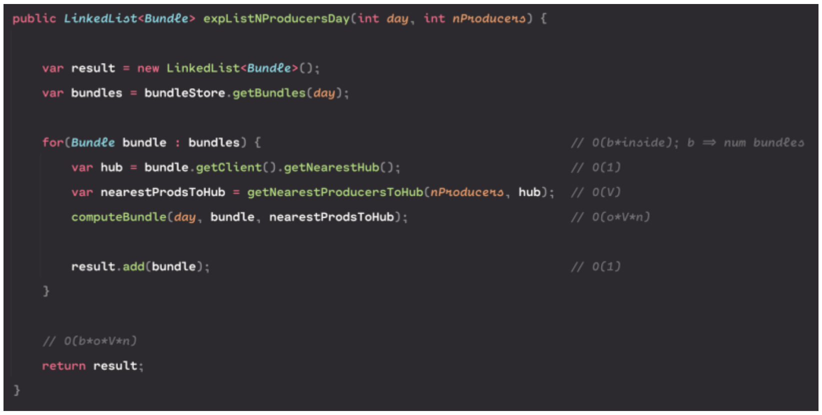
Método 6: selectProducer() [O(n)] em ExpBasketListHandler



```java
public HashMap<Integer, LinkedList<Bundle>> expBasketsList() { // O(n^4)
    ExpList expList = new ExpList();
    bundleStore = expList.getBundleStore();
    stockStore = expList.getStockStore();

    int size = bundleStore.size();
    HashMap<Integer, LinkedList<Bundle>> hash = new HashMap<>(size);

    for (int i = 1; i <= size; i++) {
        hash.put(i, expBasketsListDay(i));
    }
    expStore.addExpListNoRestrict(expList);
    return hash;
}
```

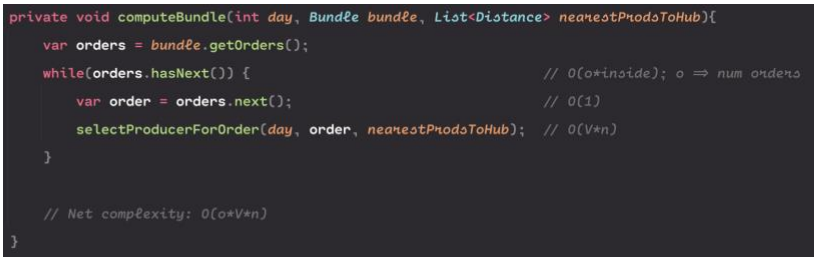
Método 7: expBasketsList() [O(n^4)] em ExpBasketsListHandler

3 US309: O(b*o*V*n)



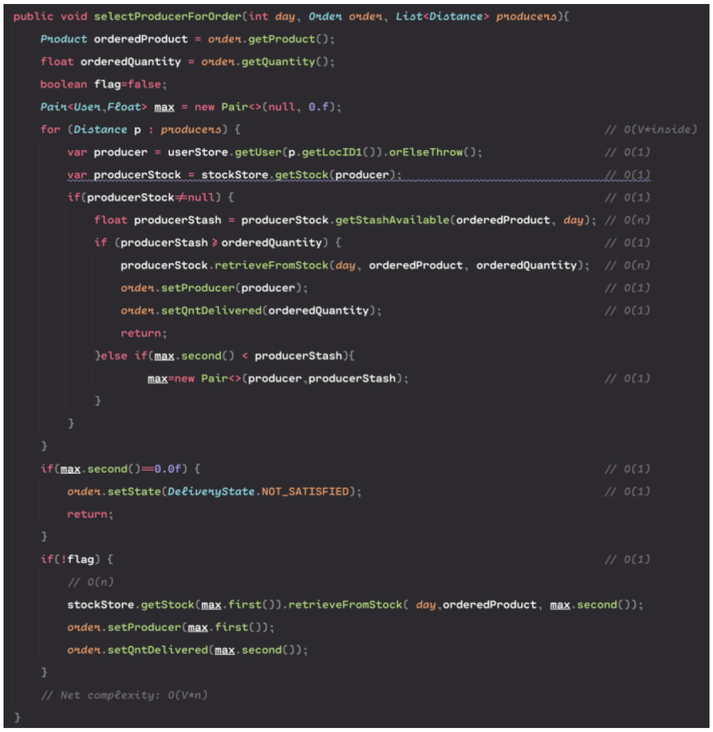
```java
public LinkedList<Bundle> expListNProducersDay(int day, int nProducers) {

    var result = new LinkedList<Bundle>();
    var bundles = bundleStore.getBundles(day);

    for(Bundle bundle : bundles) {                                          // O(b*inside); b ⇒ num bundles
        var hub = bundle.getClient().getNearestHub();                       // O(1)
        var nearestProdsToHub = getNearestProducersToHub(nProducers, hub);  // O(V)
        computeBundle(day, bundle, nearestProdsToHub);                      // O(o*V*n)

        result.add(bundle);                                                 // O(1)
    }

    // O(b*o*V*n)
    return result;
}
```

*Método 8: expListNProducersDay() [O(b*o*V*n)] em ExpListNProducersHandler*



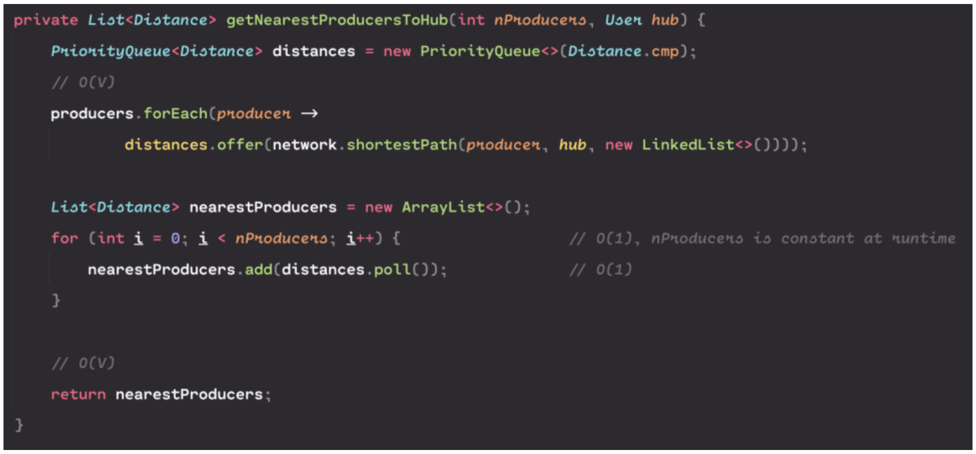
```java
private void computeBundle(int day, Bundle bundle, List<Distance> nearestProdsToHub){
    var orders = bundle.getOrders();
    while(orders.hasNext()) {                                  // O(o*inside); o ⇒ num orders
        var order = orders.next();                             // O(1)
        selectProducerForOrder(day, order, nearestProdsToHub); // O(V*n)
    }

    // Net complexity: O(o*V*n)
}
```

*Método 9: computeBundle() [O(o*V*n)] em ExpListNProducersHandler*



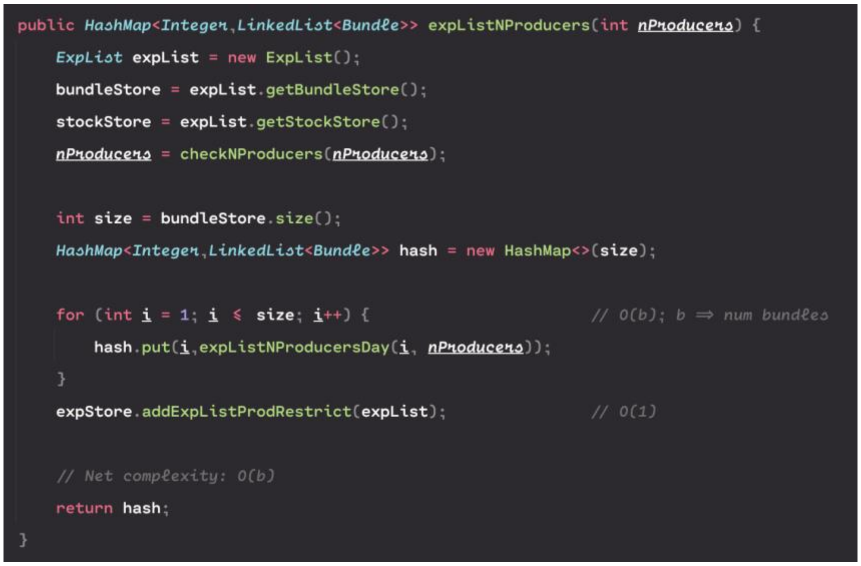
```java
public void selectProducerForOrder(int day, Order order, List<Distance> producers){
    Product orderedProduct = order.getProduct();
    float orderedQuantity = order.getQuantity();
    boolean flag=false;
    Pair<User,Float> max = new Pair<>(null, 0.f);
    for (Distance p : producers) {                                                 // O(V*inside)
        var producer = userStore.getUser(p.getLocID1()).orElseThrow();             // O(1)
        var producerStock = stockStore.getStock(producer);                         // O(1)
        if(producerStock≠null) {                                                   // O(1)
            float producerStash = producerStock.getStashAvailable(orderedProduct, day); // O(n)
            if (producerStash ≥ orderedQuantity) {                                 // O(1)
                producerStock.retrieveFromStock(day, orderedProduct, orderedQuantity);  // O(n)
                order.setProducer(producer);                                       // O(1)
                order.setQntDelivered(orderedQuantity);                            // O(1)
                return;
            }else if(max.second() < producerStash){
                    max=new Pair<>(producer,producerStash);                        // O(1)
            }
        }
    }
    if(max.second()==0.0f) {                                                       // O(1)
        order.setState(DeliveryState.NOT_SATISFIED);                               // O(1)
        return;
    }
    if(!flag) {                                                                    // O(1)
        // O(n)
        stockStore.getStock(max.first()).retrieveFromStock( day,orderedProduct, max.second());
        order.setProducer(max.first());
        order.setQntDelivered(max.second());
    }
    // Net complexity: O(V*n)
}
```

*Método 10: selectProducerForOrder() [O(V*n)] em ExpListNProducersHandler*



```java
private List<Distance> getNearestProducersToHub(int nProducers, User hub) {
    PriorityQueue<Distance> distances = new PriorityQueue<>(Distance.cmp);
    // O(V)
    producers.forEach(producer →
            distances.offer(network.shortestPath(producer, hub, new LinkedList<>())));

    List<Distance> nearestProducers = new ArrayList<>();
    for (int i = 0; i < nProducers; i++) {                 // O(1), nProducers is constant at runtime
        nearestProducers.add(distances.poll());            // O(1)
    }

    // O(V)
    return nearestProducers;
}
```

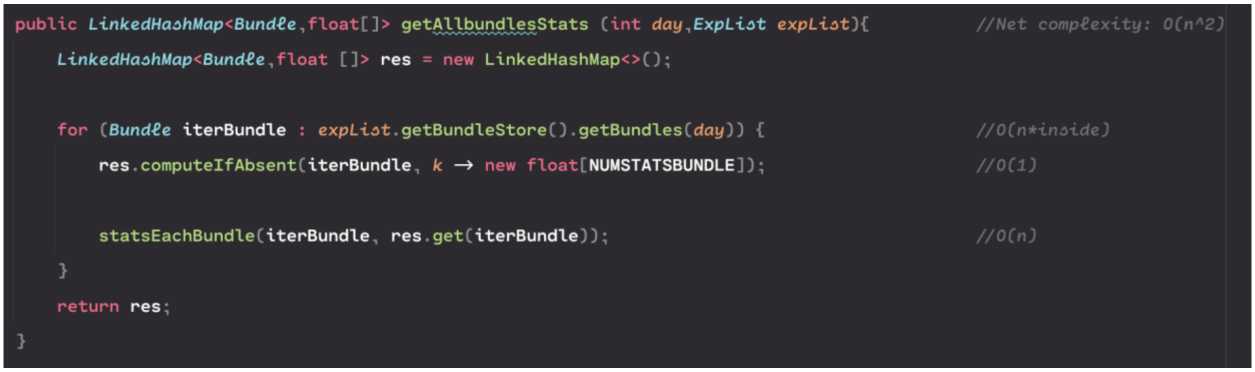
Método 11: getNearestProducersToHub() [O(V)] em ExpListNProducersHandler



```java
public HashMap<Integer,LinkedList<Bundle>> expListNProducers(int nProducers) {
    ExpList expList = new ExpList();
    bundleStore = expList.getBundleStore();
    stockStore = expList.getStockStore();
    nProducers = checkNProducers(nProducers);

    int size = bundleStore.size();
    HashMap<Integer,LinkedList<Bundle>> hash = new HashMap<>(size);

    for (int i = 1; i ≤ size; i++) {                      // O(b); b ⇒ num bundles
        hash.put(i,expListNProducersDay(i, nProducers));
    }
    expStore.addExpListProdRestrict(expList);             // O(1)

    // Net complexity: O(b)
    return hash;
}
```

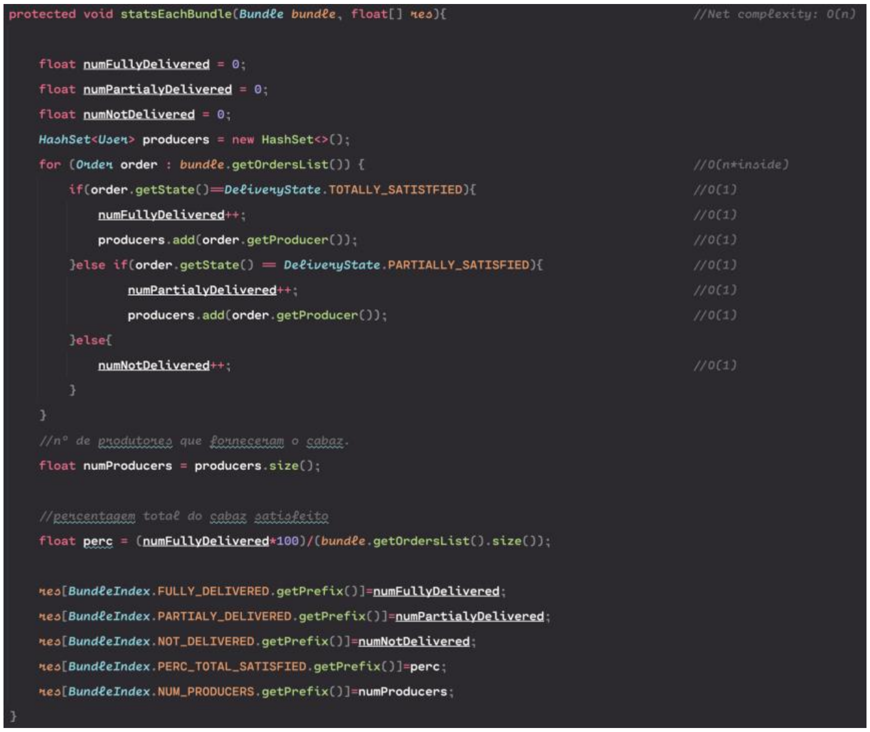
Método 12: expListNProducers() [O(b)] em ExpListNProducersHandler

4 US310: O(n^3)



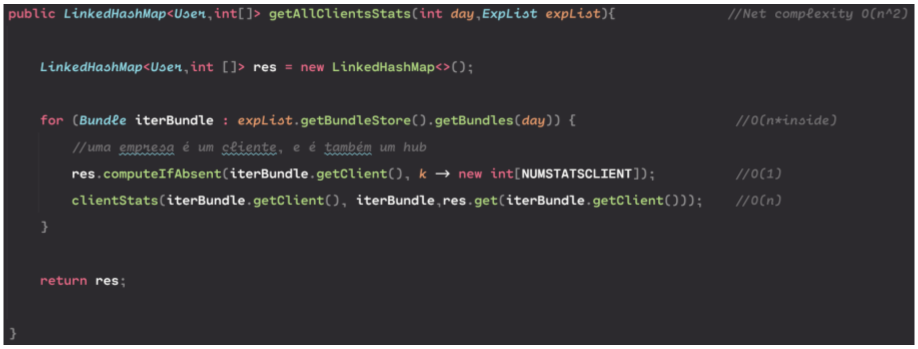
```java
public LinkedHashMap<Bundle,float[]> getAllbundlesStats (int day,ExpList expList){          //Net complexity: O(n^2)
    LinkedHashMap<Bundle,float []> res = new LinkedHashMap<>();

    for (Bundle iterBundle : expList.getBundleStore().getBundles(day)) {                    //O(n*inside)
        res.computeIfAbsent(iterBundle, k → new float[NUMSTATSBUNDLE]);                     //O(1)

        statsEachBundle(iterBundle, res.get(iterBundle));                                   //O(n)
    }
    return res;
}
```

Método 13: getAllbundlesStats() [O(n^2)] em ExpListStatsHandler



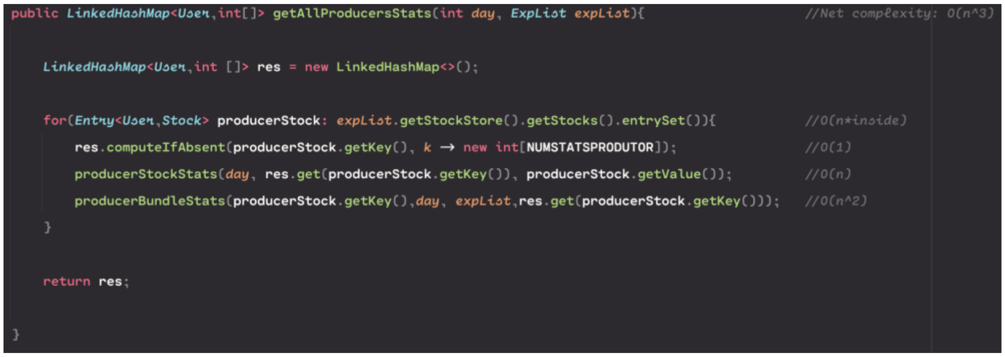
```
protected void statsEachBundle(Bundle bundle, float[] res){                          //Net complexity: O(n)

    float numFullyDelivered = 0;
    float numPartialyDelivered = 0;
    float numNotDelivered = 0;
    HashSet<User> producers = new HashSet<>();
    for (Order order : bundle.getOrdersList()) {                                      //O(n*inside)
        if(order.getState()==DeliveryState.TOTALLY_SATISTFIED){                       //O(1)
            numFullyDelivered++;                                                      //O(1)
            producers.add(order.getProducer());                                       //O(1)
        }else if(order.getState() == DeliveryState.PARTIALLY_SATISFIED){              //O(1)
                numPartialyDelivered++;                                               //O(1)
                producers.add(order.getProducer());                                   //O(1)
        }else{
            numNotDelivered++;                                                        //O(1)
        }
    }
    //nº de produtores que forneceram o cabaz.
    float numProducers = producers.size();

    //percentagem total do cabaz satisfeito
    float perc = (numFullyDelivered*100)/(bundle.getOrdersList().size());

    res[BundleIndex.FULLY_DELIVERED.getPrefix()]=numFullyDelivered;
    res[BundleIndex.PARTIALY_DELIVERED.getPrefix()]=numPartialyDelivered;
    res[BundleIndex.NOT_DELIVERED.getPrefix()]=numNotDelivered;
    res[BundleIndex.PERC_TOTAL_SATISFIED.getPrefix()]=perc;
    res[BundleIndex.NUM_PRODUCERS.getPrefix()]=numProducers;
}
```

Método 14: statsEachBundle() [O(n)] em ExpListStatsHandler



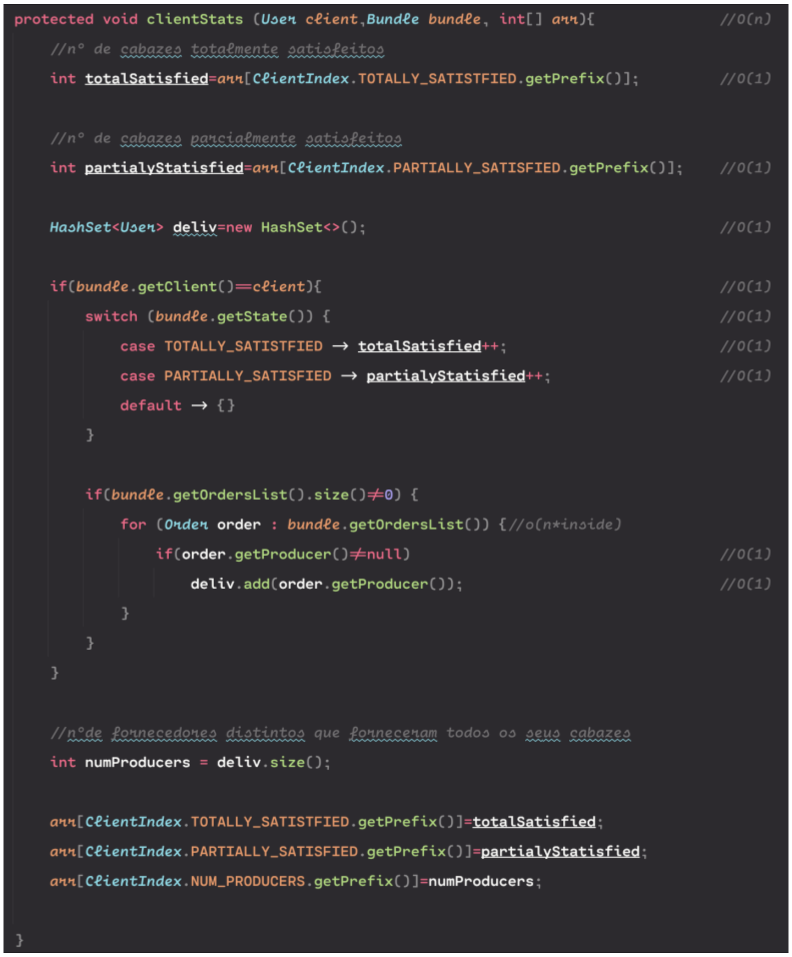
```
public LinkedHashMap<User,int[]> getAllClientsStats(int day,ExpList expList){          //Net complexity O(n^2)

    LinkedHashMap<User,int []> res = new LinkedHashMap<>();

    for (Bundle iterBundle : expList.getBundleStore().getBundles(day)) {                 //O(n*inside)
        //uma empresa é um cliente, e é também um hub
        res.computeIfAbsent(iterBundle.getClient(), k -> new int[NUMSTATSCLIENT]);       //O(1)
        clientStats(iterBundle.getClient(), iterBundle,res.get(iterBundle.getClient()));  //O(n)
    }

    return res;

}
```

Método 15: getAllClientsStats() [O(n^2)] em ExpListStatsHandler



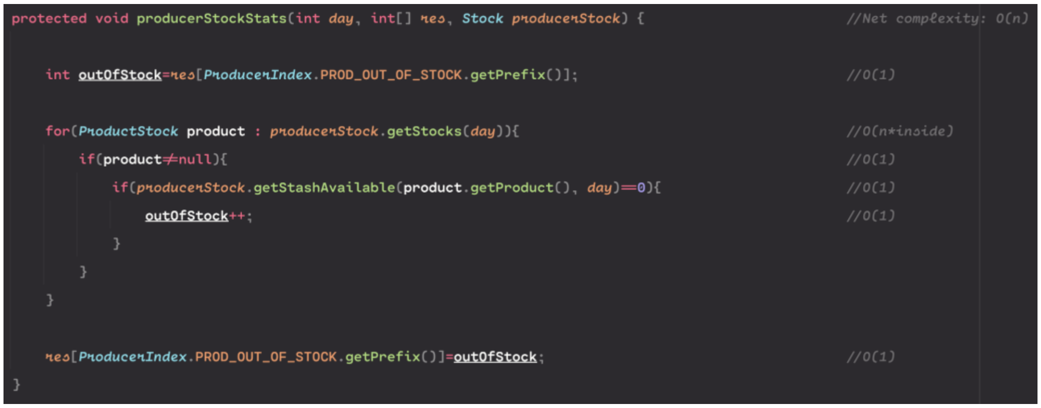
```
public LinkedHashMap<User,int[]> getAllProducersStats(int day, ExpList expList){                     //Net complexity: O(n^3)

    LinkedHashMap<User,int []> res = new LinkedHashMap<>();

    for(Entry<User,Stock> producerStock: expList.getStockStore().getStocks().entrySet()){             //O(n*inside)
        res.computeIfAbsent(producerStock.getKey(), k -> new int[NUMSTATSPRODUTOR]);                  //O(1)
        producerStockStats(day, res.get(producerStock.getKey()), producerStock.getValue());           //O(n)
        producerBundleStats(producerStock.getKey(),day, expList,res.get(producerStock.getKey()));     //O(n^2)
    }

    return res;

}
```

Método 16: getAllProducerssStats() [O(n^3)] em ExpListStatsHandler



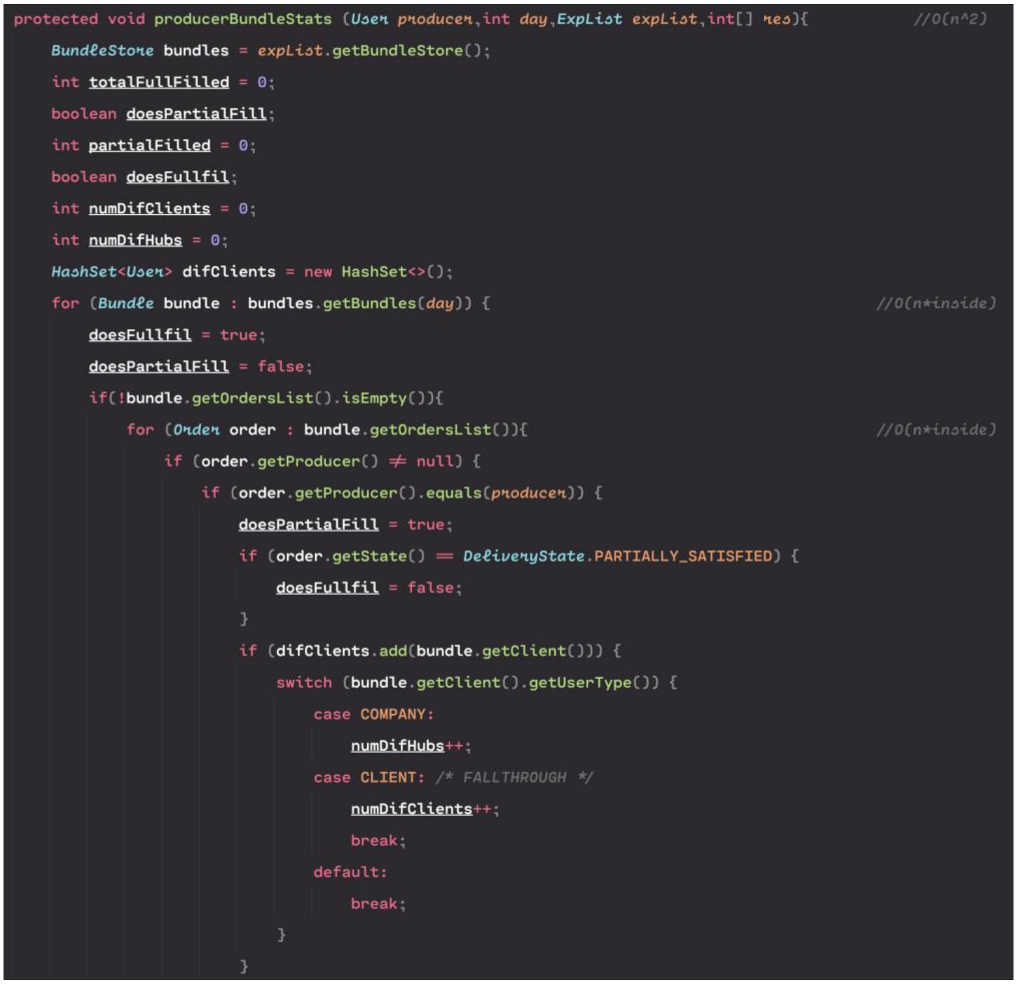
```java
protected void clientStats (User client,Bundle bundle, int[] arr){                    //O(n)
    //n° de cabazes totalmente satisfeitos
    int totalSatisfied=arr[ClientIndex.TOTALLY_SATISTFIED.getPrefix()];               //O(1)

    //n° de cabazes parcialmente satisfeitos
    int partialyStatisfied=arr[ClientIndex.PARTIALLY_SATISFIED.getPrefix()];          //O(1)

    HashSet<User> deliv=new HashSet<>();                                              //O(1)

    if(bundle.getClient()==client){                                                   //O(1)
        switch (bundle.getState()) {                                                  //O(1)
            case TOTALLY_SATISTFIED → totalSatisfied++;                               //O(1)
            case PARTIALLY_SATISFIED → partialyStatisfied++;                          //O(1)
            default → {}
        }

        if(bundle.getOrdersList().size()≠0) {
            for (Order order : bundle.getOrdersList()) {//o(n*inside)
                if(order.getProducer()≠null)                                          //O(1)
                    deliv.add(order.getProducer());                                   //O(1)
            }
        }
    }

    //n°de fornecedores distintos que forneceram todos os seus cabazes
    int numProducers = deliv.size();

    arr[ClientIndex.TOTALLY_SATISTFIED.getPrefix()]=totalSatisfied;
    arr[ClientIndex.PARTIALLY_SATISFIED.getPrefix()]=partialyStatisfied;
    arr[ClientIndex.NUM_PRODUCERS.getPrefix()]=numProducers;

}
```

Método 17: clientStats() [O(n)] em ExpListStatsHandler



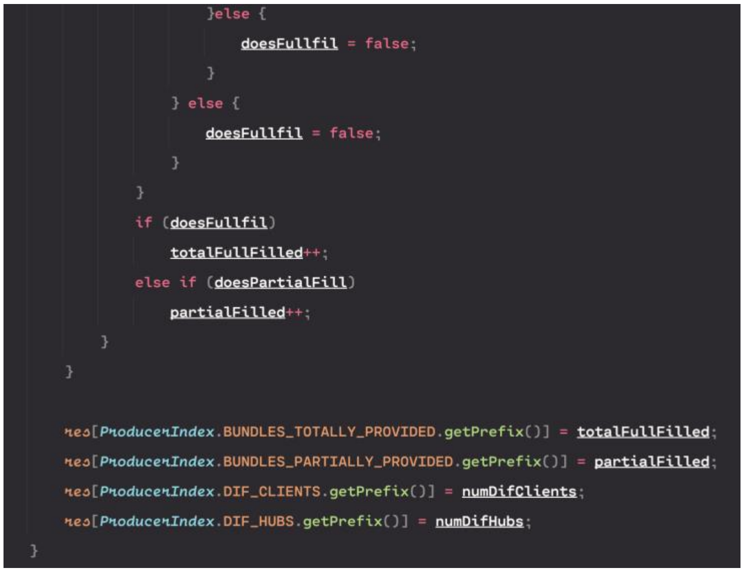
```java
protected void producerStockStats(int day, int[] res, Stock producerStock) {          //Net complexity: O(n)

    int outOfStock=res[ProducerIndex.PROD_OUT_OF_STOCK.getPrefix()];                  //O(1)

    for(ProductStock product : producerStock.getStocks(day)){                         //O(n*inside)
        if(product≠null){                                                             //O(1)
            if(producerStock.getStashAvailable(product.getProduct(), day)==0){        //O(1)
                outOfStock++;                                                         //O(1)
            }
        }
    }

    res[ProducerIndex.PROD_OUT_OF_STOCK.getPrefix()]=outOfStock;                      //O(1)
}
```

Método 18: producerStockStats() [O(n)] em ExpListStatsHandler



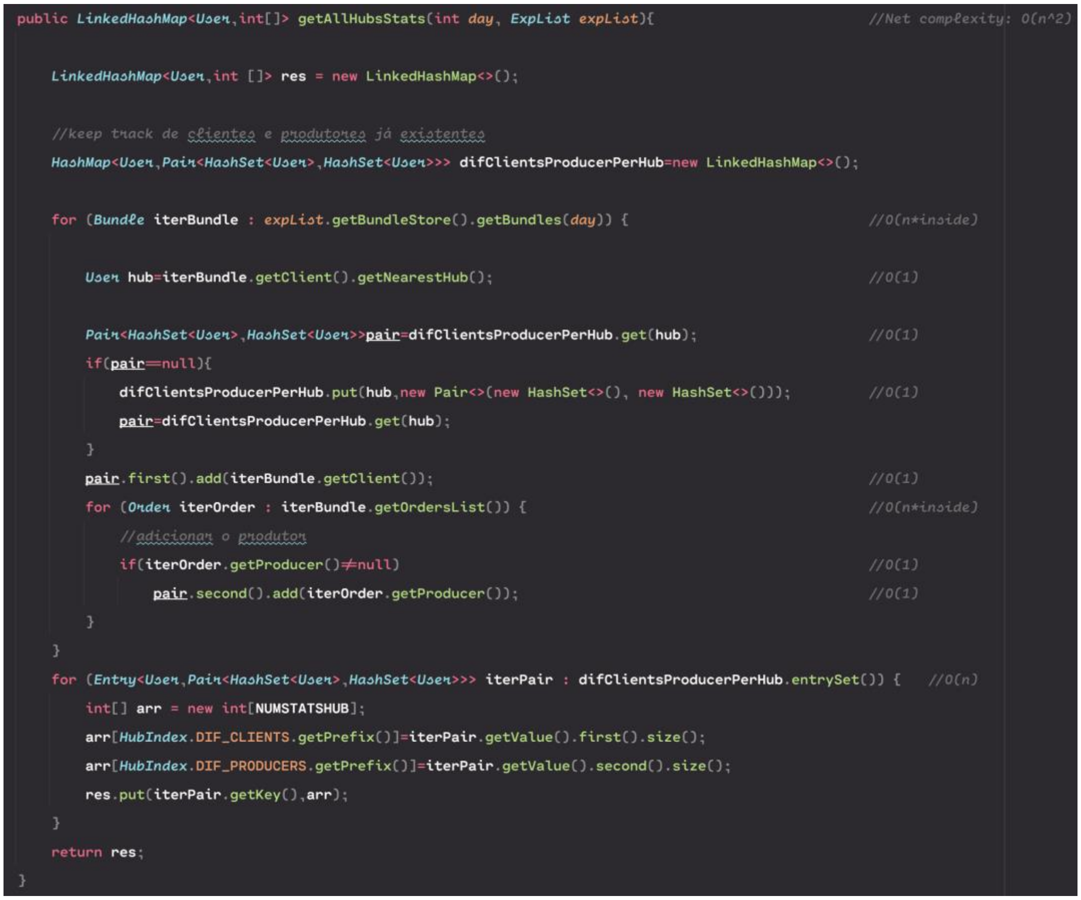
```java
protected void producerBundleStats (User producer,int day,ExpList expList,int[] res){          //O(n^2)
    BundleStore bundles = expList.getBundleStore();
    int totalFullFilled = 0;
    boolean doesPartialFill;
    int partialFilled = 0;
    boolean doesFullfil;
    int numDifClients = 0;
    int numDifHubs = 0;
    HashSet<User> difClients = new HashSet<>();
    for (Bundle bundle : bundles.getBundles(day)) {                                                //O(n*inside)
        doesFullfil = true;
        doesPartialFill = false;
        if(!bundle.getOrdersList().isEmpty()){
            for (Order order : bundle.getOrdersList()){                                            //O(n*inside)
                if (order.getProducer() ≠ null) {
                    if (order.getProducer().equals(producer)) {
                        doesPartialFill = true;
                        if (order.getState() == DeliveryState.PARTIALLY_SATISFIED) {
                            doesFullfil = false;
                        }
                        if (difClients.add(bundle.getClient())) {
                            switch (bundle.getClient().getUserType()) {
                                case COMPANY:
                                    numDifHubs++;
                                case CLIENT: /* FALLTHROUGH */
                                    numDifClients++;
                                    break;
                                default:
                                    break;
                            }
                        }
```

Método 19: producerBundleStats() [O(n^2)] em ExpListStatsHandler [1]



```java
                    }else {
                        doesFullfil = false;
                    }
                } else {
                    doesFullfil = false;
                }
            }
            if (doesFullfil)
                totalFullFilled++;
            else if (doesPartialFill)
                partialFilled++;
        }
    }

    res[ProducerIndex.BUNDLES_TOTALLY_PROVIDED.getPrefix()] = totalFullFilled;
    res[ProducerIndex.BUNDLES_PARTIALLY_PROVIDED.getPrefix()] = partialFilled;
    res[ProducerIndex.DIF_CLIENTS.getPrefix()] = numDifClients;
    res[ProducerIndex.DIF_HUBS.getPrefix()] = numDifHubs;
}
```

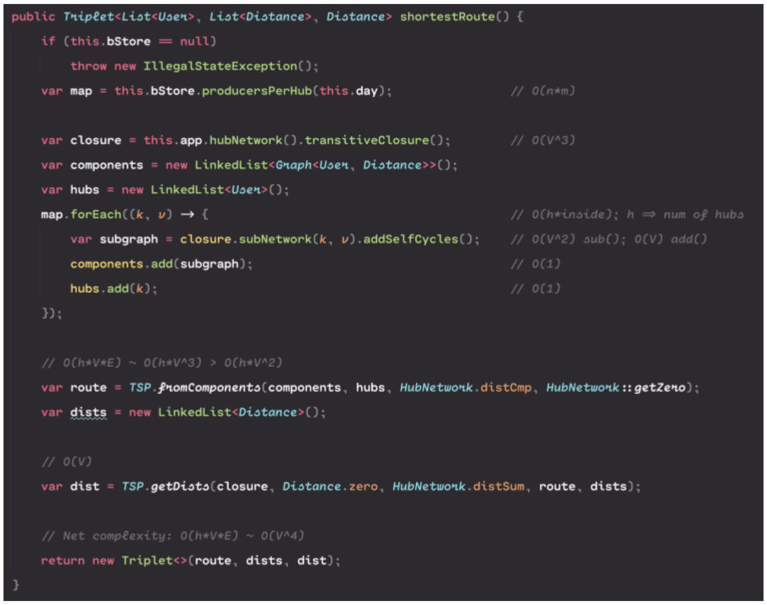
Método 20: producerBundleStats() [O(n^2)] em ExpListStatsHandler [2]



```
public LinkedHashMap<User,int[]> getAllHubsStats(int day, ExpList expList){                //Net complexity: O(n^2)

    LinkedHashMap<User,int []> res = new LinkedHashMap<>();

    //keep track de clientes e produtores já existentes
    HashMap<User,Pair<HashSet<User>,HashSet<User>>> difClientsProducerPerHub=new LinkedHashMap<>();

    for (Bundle iterBundle : expList.getBundleStore().getBundles(day)) {                   //O(n*inside)

        User hub=iterBundle.getClient().getNearestHub();                                   //O(1)

        Pair<HashSet<User>,HashSet<User>>pair=difClientsProducerPerHub.get(hub);           //O(1)
        if(pair==null){
            difClientsProducerPerHub.put(hub,new Pair<>(new HashSet<>(), new HashSet<>()));   //O(1)
            pair=difClientsProducerPerHub.get(hub);
        }
        pair.first().add(iterBundle.getClient());                                          //O(1)
        for (Order iterOrder : iterBundle.getOrdersList()) {                               //O(n*inside)
            //adicionar o produtor
            if(iterOrder.getProducer()!=null)                                              //O(1)
                pair.second().add(iterOrder.getProducer());                                //O(1)
        }
    }
    for (Entry<User,Pair<HashSet<User>,HashSet<User>>> iterPair : difClientsProducerPerHub.entrySet()) {   //O(n)
        int[] arr = new int[NUMSTATSHUB];
        arr[HubIndex.DIF_CLIENTS.getPrefix()]=iterPair.getValue().first().size();
        arr[HubIndex.DIF_PRODUCERS.getPrefix()]=iterPair.getValue().second().size();
        res.put(iterPair.getKey(),arr);
    }
    return res;
}
```

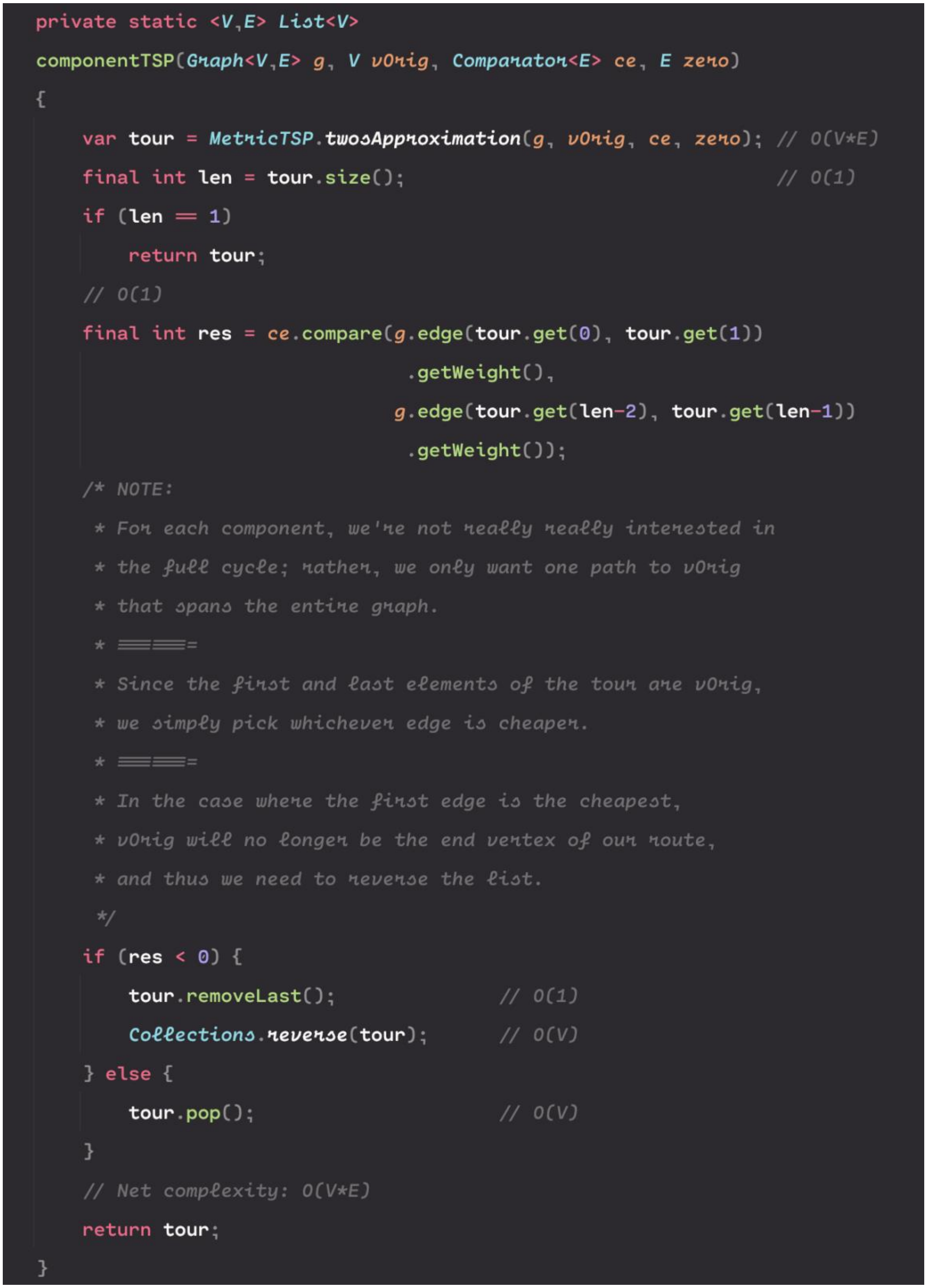
Método 21: getAllHubStats() [O(n^2)] em ExpListStatsHandler

5 US311: O(h*V*E) ~ O(V^4)



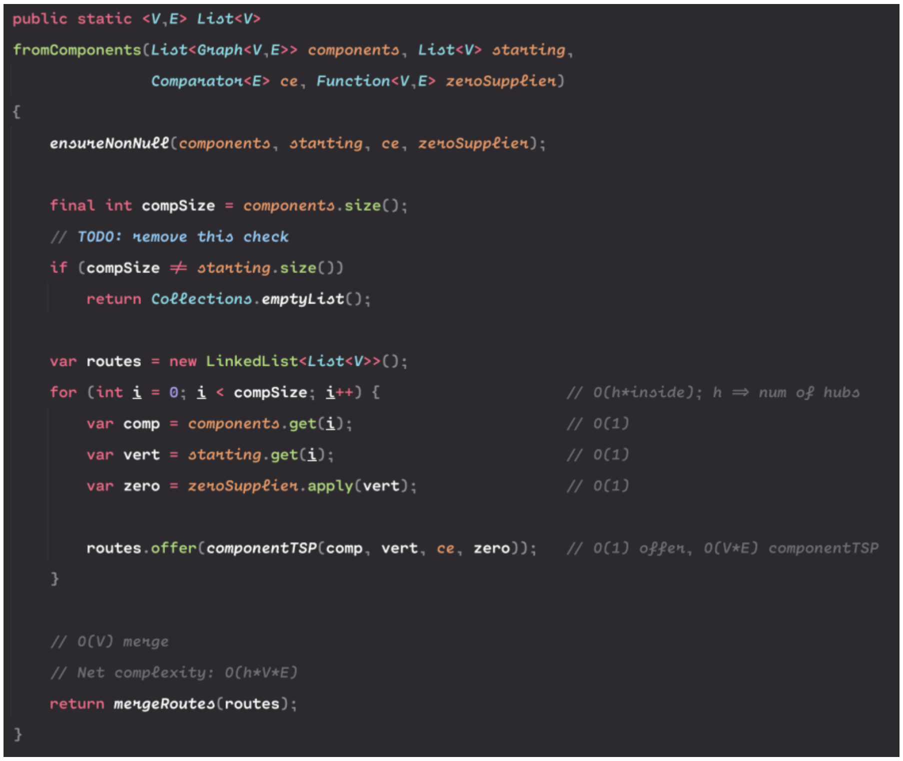
```
public Triplet<List<User>, List<Distance>, Distance> shortestRoute() {
    if (this.bStore == null)
        throw new IllegalStateException();
    var map = this.bStore.producersPerHub(this.day);                  // O(n*m)

    var closure = this.app.hubNetwork().transitiveClosure();          // O(V^3)
    var components = new LinkedList<Graph<User, Distance>>();
    var hubs = new LinkedList<User>();
    map.forEach((k, v) -> {                                           // O(h*inside); h => num of hubs
        var subgraph = closure.subNetwork(k, v).addSelfCycles();      // O(V^2) sub(); O(V) add()
        components.add(subgraph);                                     // O(1)
        hubs.add(k);                                                  // O(1)
    });

    // O(h*V*E) ~ O(h*V^3) > O(h*V^2)
    var route = TSP.fromComponents(components, hubs, HubNetwork.distCmp, HubNetwork::getZero);
    var dists = new LinkedList<Distance>();

    // O(V)
    var dist = TSP.getDists(closure, Distance.zero, HubNetwork.distSum, route, dists);

    // Net complexity: O(h*V*E) ~ O(V^4)
    return new Triplet<>(route, dists, dist);
}
```

*Método 22: shortestRoute() [O(h*V*E) ~ O(V^4)] em ShortestPathHandler*



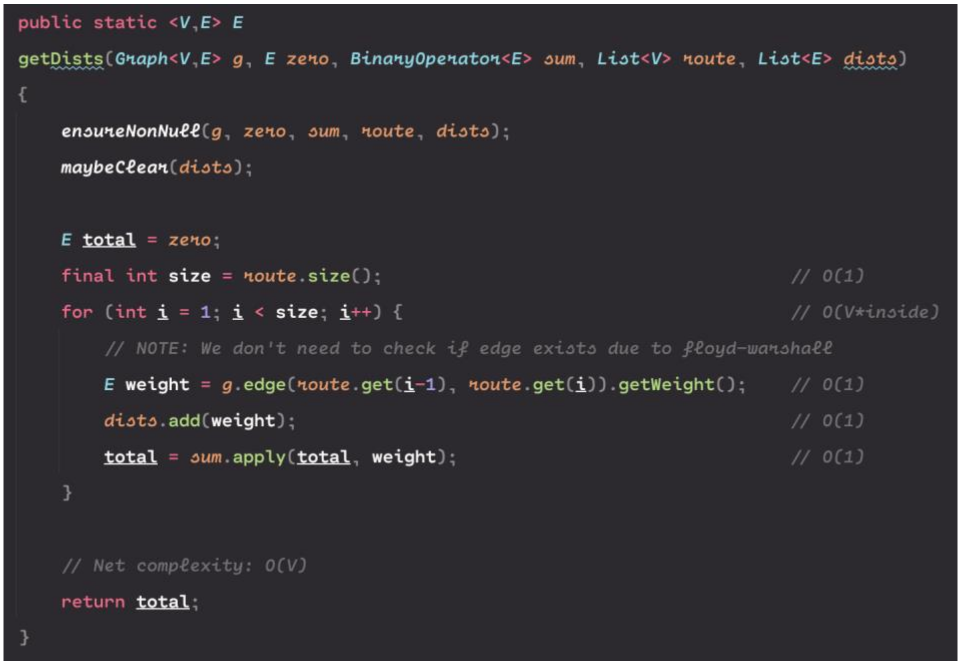
```java
private static <V,E> List<V>
componentTSP(Graph<V,E> g, V vOrig, Comparator<E> ce, E zero)
{
    var tour = MetricTSP.twosApproximation(g, vOrig, ce, zero); // O(V*E)
    final int len = tour.size();                               // O(1)
    if (len == 1)
        return tour;
    // O(1)
    final int res = ce.compare(g.edge(tour.get(0), tour.get(1))
                               .getWeight(),
                              g.edge(tour.get(len-2), tour.get(len-1))
                               .getWeight());
    /* NOTE:
     * For each component, we're not really really interested in
     * the full cycle; rather, we only want one path to vOrig
     * that spans the entire graph.
     * =======
     * Since the first and last elements of the tour are vOrig,
     * we simply pick whichever edge is cheaper.
     * =======
     * In the case where the first edge is the cheapest,
     * vOrig will no longer be the end vertex of our route,
     * and thus we need to reverse the list.
     */
    if (res < 0) {
        tour.removeLast();            // O(1)
        Collections.reverse(tour);    // O(V)
    } else {
        tour.pop();                   // O(V)
    }
    // Net complexity: O(V*E)
    return tour;
}
```

*Método 23: componentTSP() [O(V*E)] em TSP*



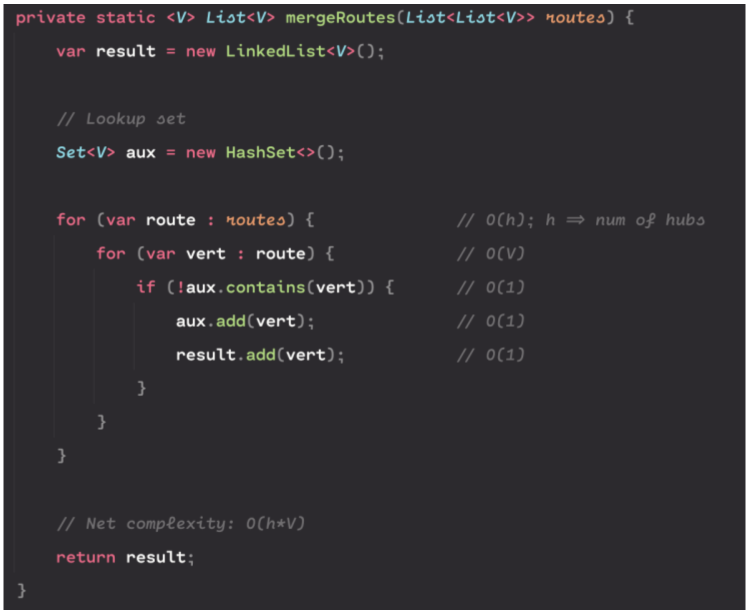
```
public static <V,E> List<V>
fromComponents(List<Graph<V,E>> components, List<V> starting,
               Comparator<E> ce, Function<V,E> zeroSupplier)
{
    ensureNonNull(components, starting, ce, zeroSupplier);

    final int compSize = components.size();
    // TODO: remove this check
    if (compSize ≠ starting.size())
        return Collections.emptyList();

    var routes = new LinkedList<List<V>>();
    for (int i = 0; i < compSize; i++) {                   // O(h*inside); h ⇒ num of hubs
        var comp = components.get(i);                      // O(1)
        var vert = starting.get(i);                        // O(1)
        var zero = zeroSupplier.apply(vert);               // O(1)

        routes.offer(componentTSP(comp, vert, ce, zero));  // O(1) offer, O(V*E) componentTSP
    }

    // O(V) merge
    // Net complexity: O(h*V*E)
    return mergeRoutes(routes);
}
```

*Método 24: fromComponents() [O(h*V*E)] em TSP*



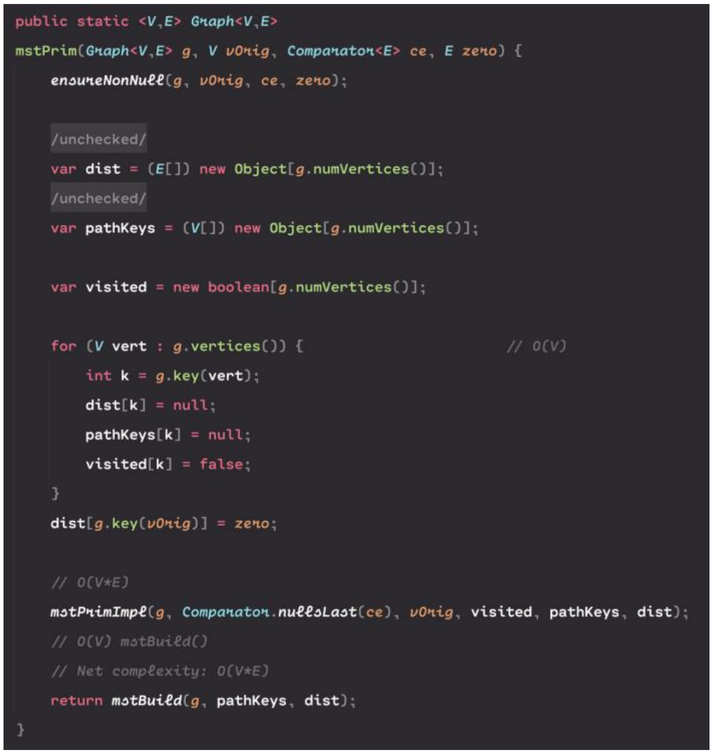
```
public static <V,E> E
getDists(Graph<V,E> g, E zero, BinaryOperator<E> sum, List<V> route, List<E> dists)
{
    ensureNonNull(g, zero, sum, route, dists);
    maybeClear(dists);

    E total = zero;
    final int size = route.size();                                      // O(1)
    for (int i = 1; i < size; i++) {                                    // O(V*inside)
        // NOTE: We don't need to check if edge exists due to floyd-warshall
        E weight = g.edge(route.get(i-1), route.get(i)).getWeight();    // O(1)
        dists.add(weight);                                              // O(1)
        total = sum.apply(total, weight);                               // O(1)
    }

    // Net complexity: O(V)
    return total;
}
```

Método 25: getDists() [O(V)] em TSP



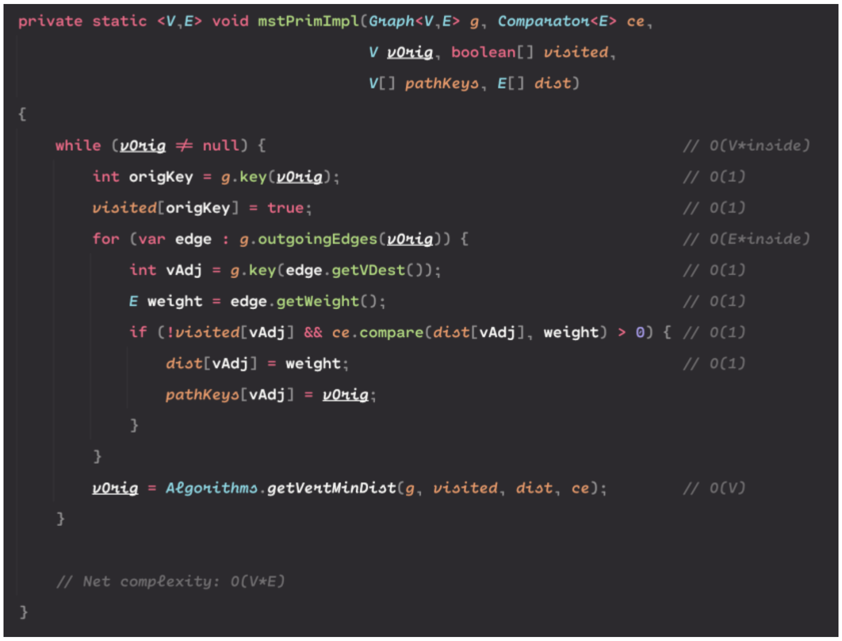
```
private static <V> List<V> mergeRoutes(List<List<V>> routes) {
    var result = new LinkedList<V>();

    // Lookup set
    Set<V> aux = new HashSet<>();

    for (var route : routes) {              // O(h); h ⇒ num of hubs
        for (var vert : route) {            // O(V)
            if (!aux.contains(vert)) {      // O(1)
                aux.add(vert);              // O(1)
                result.add(vert);           // O(1)
            }
        }
    }

    // Net complexity: O(h*V)
    return result;
}
```

*Método 26: mergeRoutes() [O(h*V)] em TSP*



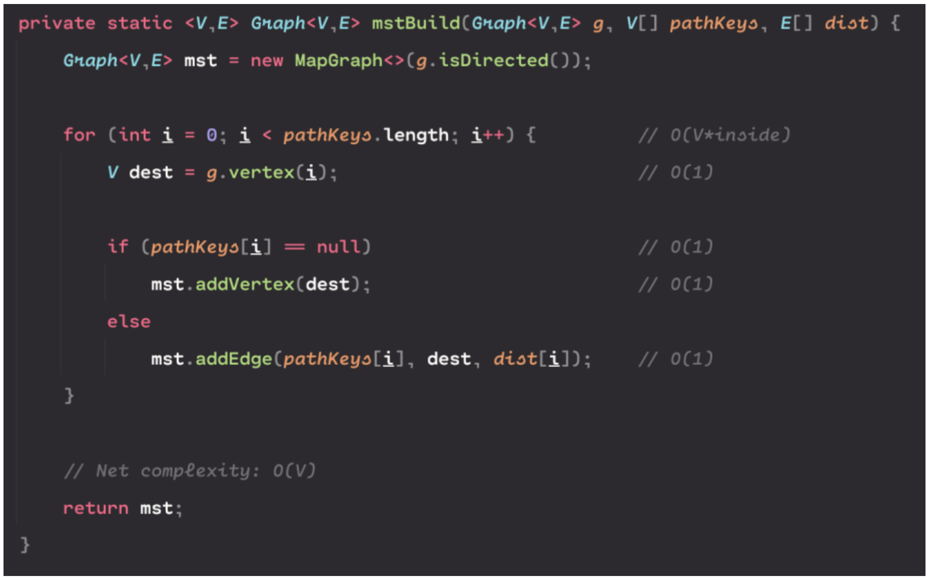
```
public static <V,E> Graph<V,E>
mstPrim(Graph<V,E> g, V vOrig, Comparator<E> ce, E zero) {
    ensureNonNull(g, vOrig, ce, zero);

    /unchecked/
    var dist = (E[]) new Object[g.numVertices()];
    /unchecked/
    var pathKeys = (V[]) new Object[g.numVertices()];

    var visited = new boolean[g.numVertices()];

    for (V vert : g.vertices()) {                       // O(V)
        int k = g.key(vert);
        dist[k] = null;
        pathKeys[k] = null;
        visited[k] = false;
    }
    dist[g.key(vOrig)] = zero;

    // O(V*E)
    mstPrimImpl(g, Comparator.nullsLast(ce), vOrig, visited, pathKeys, dist);
    // O(V) mstBuild()
    // Net complexity: O(V*E)
    return mstBuild(g, pathKeys, dist);
}
```

*Método 27: mstPrim() [O(V*E)] em MetricTSP*



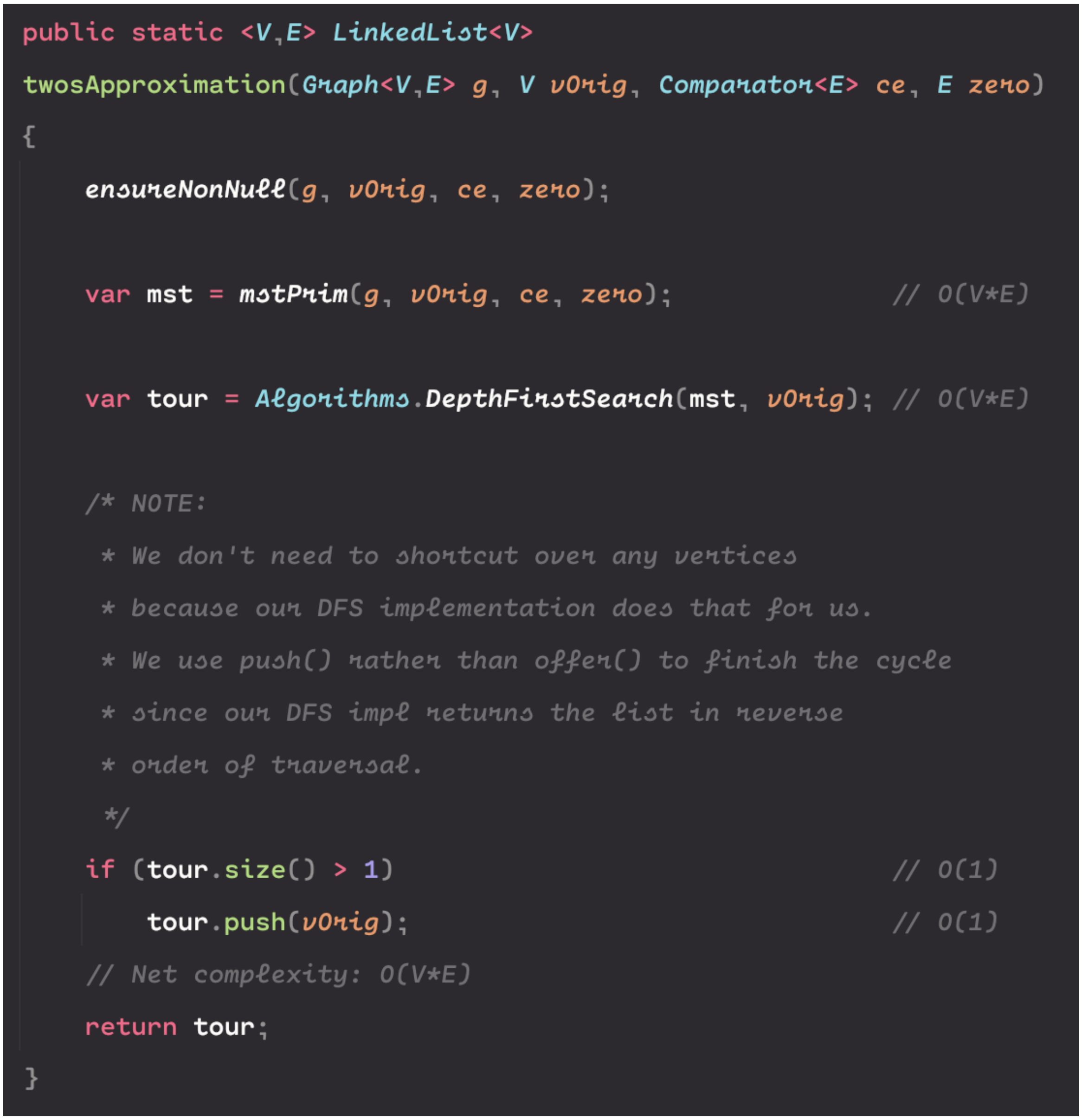
```java
private static <V,E> void mstPrimImpl(Graph<V,E> g, Comparator<E> ce,
                                      V vOrig, boolean[] visited,
                                      V[] pathKeys, E[] dist)
{
    while (vOrig ≠ null) {                                                  // O(V*inside)
        int origKey = g.key(vOrig);                                         // O(1)
        visited[origKey] = true;                                            // O(1)
        for (var edge : g.outgoingEdges(vOrig)) {                           // O(E*inside)
            int vAdj = g.key(edge.getVDest());                              // O(1)
            E weight = edge.getWeight();                                    // O(1)
            if (!visited[vAdj] && ce.compare(dist[vAdj], weight) > 0) { // O(1)
                dist[vAdj] = weight;                                        // O(1)
                pathKeys[vAdj] = vOrig;
            }
        }
        vOrig = Algorithms.getVertMinDist(g, visited, dist, ce);        // O(V)
    }

    // Net complexity: O(V*E)
}
```

*Método 28: mstPrimImpl() [O(V*E)] em MetricTSP*



```java
private static <V,E> Graph<V,E> mstBuild(Graph<V,E> g, V[] pathKeys, E[] dist) {
    Graph<V,E> mst = new MapGraph<>(g.isDirected());

    for (int i = 0; i < pathKeys.length; i++) {          // O(V*inside)
        V dest = g.vertex(i);                           // O(1)

        if (pathKeys[i] == null)                        // O(1)
            mst.addVertex(dest);                        // O(1)
        else
            mst.addEdge(pathKeys[i], dest, dist[i]);    // O(1)
    }

    // Net complexity: O(V)
    return mst;
}
```

Método 29: mstBuild() [O(V)] em MetricTSP



```java
public static <V,E> LinkedList<V>
twosApproximation(Graph<V,E> g, V vOrig, Comparator<E> ce, E zero)
{
    ensureNonNull(g, vOrig, ce, zero);

    var mst = mstPrim(g, vOrig, ce, zero);                    // O(V*E)

    var tour = Algorithms.DepthFirstSearch(mst, vOrig); // O(V*E)

    /* NOTE:
     * We don't need to shortcut over any vertices
     * because our DFS implementation does that for us.
     * We use push() rather than offer() to finish the cycle
     * since our DFS impl returns the list in reverse
     * order of traversal.
     */
    if (tour.size() > 1)                                      // O(1)
        tour.push(vOrig);                                     // O(1)
    // Net complexity: O(V*E)
    return tour;
}
```

*Método 30: twosApproximation() [O(V*E)] em MetricTSP*