板寄せ

// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.7;

contract agreement {

// （２）買い手のステイタスの追加

uint256 public agreementeth;

uint256 newBlength;

uint256 newSlength;

BuyerStatus[] public Buyerstatus;

struct BuyerStatus {

address buyer;

uint256 kwh;

uint256 value;

uint256 sum;

uint256 totalCoin;

}

mapping(address => uint256) public balanceOf;

mapping(address => uint256) public electricityOf;

SellerStatus[] public Sellerstatus;

struct SellerStatus {

address seller;

uint256 kwh;

uint256 value;

uint256 sum;

uint256 totalCoin;

}

function pushBuyerStatus(address \_buyer, uint256 \_kwh, uint256 \_value, uint \_all, uint \_totalCoin) public

{

balanceOf[\_buyer] = \_totalCoin;

electricityOf[\_buyer] = \_all;

if (balanceOf[\_buyer] < \_value\*\_kwh){

revert();

}

Buyerstatus.push(BuyerStatus({

buyer: \_buyer,

kwh: \_kwh,

value: \_value,

sum:\_kwh,

totalCoin:\_totalCoin

}));

}

// （3）売り手のステイタスの追加

function pushSellerStatus(address \_seller, uint256 \_kwh, uint256 \_value, uint256 \_all, uint \_totalCoin) public

{

balanceOf[\_seller] = \_totalCoin;

electricityOf[\_seller] = \_all;

if (electricityOf[\_seller] < \_kwh){

revert();

}

Sellerstatus.push(SellerStatus({

seller: \_seller,

kwh: \_kwh,

value: \_value,

sum: \_kwh,

totalCoin:\_totalCoin

}));

}

// (4)取引実行(板寄せによる並び替え)

function Agreement () public {

address tmpb;

uint256 tmpk;

uint256 tmpv;

uint256 tmps;

for (uint256 i = 0; i < Buyerstatus.length; i++) {

for (uint256 j = (Buyerstatus.length-1); j > i; j--) {

if (Buyerstatus[j].value > Buyerstatus[j-1].value) {

tmpb = Buyerstatus[j].buyer;

tmpk = Buyerstatus[j].kwh;

tmpv = Buyerstatus[j].value;

tmps = Buyerstatus[j].sum;

Buyerstatus[j] = Buyerstatus[j-1];

Buyerstatus[j-1].buyer = tmpb;

Buyerstatus[j-1].kwh = tmpk;

Buyerstatus[j-1].value = tmpv;

Buyerstatus[j-1].sum = tmps;

}

}

}

for (uint256 k = 1; k < Buyerstatus.length; k++) {

Buyerstatus[k].sum = Buyerstatus[k].kwh + Buyerstatus[k-1].sum;

}

address tmpss;

uint tmpc;

uint tmpvv;

uint tmpsum;

for (uint256 x = 0; x < Sellerstatus.length; x++) {

for (uint256 y = (Sellerstatus.length-1); y > x; y--) {

if (Sellerstatus[y].value < Sellerstatus[y-1].value) {

tmpss = Sellerstatus[y].seller;

tmpc = Sellerstatus[y].kwh;

tmpvv = Sellerstatus[y].value;

tmpsum = Sellerstatus[y].sum;

Sellerstatus[y] = Sellerstatus[y-1];

Sellerstatus[y-1].seller = tmpss;

Sellerstatus[y-1].kwh = tmpc;

Sellerstatus[y-1].value = tmpvv;

Sellerstatus[y-1].sum = tmpsum;

}

}

}

for (uint256 n = 1; n < Sellerstatus.length; n++) {

Sellerstatus[n].sum = Sellerstatus[n].kwh + Sellerstatus[n-1].sum;

}

uint sagr = 0;

uint bagr = 0;

for ( sagr ; sagr < Sellerstatus.length;){

for (bagr ; bagr < Buyerstatus.length; ){

if(Buyerstatus[bagr].value > Sellerstatus[sagr].value){

if(Buyerstatus[bagr].sum <= Sellerstatus[sagr].sum){

bagr++;}

else if(Buyerstatus[bagr].sum > Sellerstatus[sagr].sum){

sagr++;}

}

else{

if (Sellerstatus[sagr].sum>Buyerstatus[bagr-1].sum){

agreementeth = Sellerstatus[sagr].value;

}

if (Sellerstatus[sagr-1].sum<Buyerstatus[bagr].sum){

agreementeth = Buyerstatus[bagr].value;

}

if (Sellerstatus[sagr].value==Buyerstatus[bagr].value){

agreementeth = Buyerstatus[bagr].value;

}

break;

}

}

if (agreementeth == Sellerstatus[sagr].value||agreementeth == Buyerstatus[bagr].value) {

break;

}

}

if (Sellerstatus[sagr].value > agreementeth) {

sagr -= 1 ;

}

if (Buyerstatus[bagr].value < agreementeth) {

bagr -= 1 ;

}

if (Buyerstatus[bagr].sum < Sellerstatus[sagr].sum && Buyerstatus[bagr].value == Buyerstatus[bagr+1].value) {

for ( uint b = 1; b < Buyerstatus.length; b++) {

if (Buyerstatus[bagr + b].sum > Sellerstatus[sagr].sum) {

bagr += b;

break;

}

}

}

if (Sellerstatus[sagr].sum < Buyerstatus[bagr].sum && Sellerstatus[sagr].value == Sellerstatus[sagr+1].value) {

for ( uint b = 1; b < Sellerstatus.length; b++) {

if (Sellerstatus[sagr + b].sum > Buyerstatus[bagr].sum) {

sagr += b;

break;

}

}

}

//取引開始(電気とトークン双方)

//約定価格以上のバイヤーの取引

for (uint a = 0; a < bagr ; a++){

electricityOf[Buyerstatus[a].buyer] += Buyerstatus[a].kwh;

balanceOf[Buyerstatus[a].buyer] -= agreementeth \* Buyerstatus[a].kwh;

Buyerstatus[a].kwh = 0;

}

//約定価格以下のセーラーの取引

for (uint a = 0; a < sagr ; a++) {

electricityOf[Sellerstatus[a].seller] -= Sellerstatus[a].kwh;

balanceOf[Sellerstatus[a].seller] += agreementeth \* Sellerstatus[a].kwh;

Sellerstatus[a].kwh = 0;

}

//約定価格でのセーラーとバイヤーの取引

if(Buyerstatus[bagr].sum > Sellerstatus[sagr].sum){

uint gapa = Buyerstatus[bagr].sum - Sellerstatus[sagr].sum;

balanceOf[Buyerstatus[bagr].buyer] -= (Buyerstatus[bagr].kwh - gapa) \* agreementeth;

balanceOf[Sellerstatus[sagr].seller] += Sellerstatus[sagr].kwh \* agreementeth;

electricityOf[Buyerstatus[bagr].buyer] = electricityOf[Buyerstatus[bagr].buyer] + Buyerstatus[bagr].kwh - gapa;

Buyerstatus[bagr].kwh = gapa;

electricityOf[Sellerstatus[sagr].seller] -= Sellerstatus[sagr].kwh;

Sellerstatus[sagr].kwh = 0;

}

else if(Buyerstatus[bagr].sum < Sellerstatus[sagr].sum){

uint gapb = Sellerstatus[sagr].sum - Buyerstatus[bagr].sum;

balanceOf[Sellerstatus[sagr].seller] += (Sellerstatus[sagr].kwh - gapb) \* agreementeth;

balanceOf[Buyerstatus[bagr].buyer] -= Buyerstatus[bagr].kwh \* agreementeth;

electricityOf[Sellerstatus[sagr].seller] = electricityOf[Sellerstatus[sagr].seller] - Sellerstatus[sagr].kwh + gapb;

Sellerstatus[sagr].kwh = gapb;

electricityOf[Buyerstatus[bagr].buyer] = electricityOf[Buyerstatus[bagr].buyer] + Buyerstatus[bagr].kwh;

Buyerstatus[bagr].kwh = 0;

}

//取引希望電力が0の人消去

for (uint b = 0 ; b < Buyerstatus.length && b < Sellerstatus.length; b++){

//buyerの消去

if (Buyerstatus[b].kwh == 0){

delete Buyerstatus[b];

}

//sellerの消去

if (Sellerstatus[b].kwh == 0){

delete Sellerstatus[b];

}

}

//板情報のリセット

uint z = 0 ;

for (uint j = 0 ; j < Buyerstatus.length; j++){

if (Buyerstatus[j].kwh == 0) {

z++;

}

}

newBlength = Buyerstatus.length - z;

if ( z != 0 ){

for (uint k = 0 ; k < newBlength ; k++){

Buyerstatus[k] = Buyerstatus[k + z];

delete Buyerstatus[k + z];

}

}

//Sellerstatusの板情報更新

uint h = 0 ;

for (uint a = 0 ; a < Sellerstatus.length; a++){

if( Sellerstatus[a].kwh == 0) {

h++;

}

}

newSlength = Sellerstatus.length - h;

if ( h != 0 ){

for (uint b = 0 ; b < newSlength ; b++){

Sellerstatus[b] = Sellerstatus[h + b];

delete Sellerstatus[h + b];

}

}

for (uint i = 0 ; i < Buyerstatus.length && i < Sellerstatus.length; i++){ //buyer=sellerじゃないと成り立たない。課題の一つ

//buyerの消去

if (Buyerstatus[i].kwh == 0){

delete Buyerstatus[i];

}

//sellerの消去

if (Sellerstatus[i].kwh == 0){

delete Sellerstatus[i];

}

}

}

}

ザラ場

// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.7;

import "agreement.sol";

contract zaraba is agreement {

function entryForBuyer (address \_buyer, uint256 \_kwh, uint256 \_value, uint \_all, uint \_totalCoin) public

{

balanceOf[\_buyer] = \_totalCoin;

electricityOf[\_buyer] = \_all;

if (balanceOf[\_buyer] < \_value\*\_kwh ){

revert();

}

Buyerstatus.push(BuyerStatus({

buyer: \_buyer,

kwh: \_kwh,

value: \_value,

sum:\_kwh,

totalCoin:\_totalCoin

}));

agreementeth = Sellerstatus[0].value;

if ( \_value == agreementeth ) {

if ( \_kwh < Sellerstatus[0].kwh ) {

electricityOf[\_buyer] += \_kwh ;

electricityOf[Sellerstatus[0].seller] -= \_kwh ;

Sellerstatus[0].kwh -= \_kwh;

balanceOf[\_buyer] -= \_kwh\*Sellerstatus[0].value ;

balanceOf[Sellerstatus[0].seller] += \_kwh\*Sellerstatus[0].value;

delete Buyerstatus[Buyerstatus.length-1];

}

else if ( \_kwh == Sellerstatus[0].kwh ) {

electricityOf[\_buyer] += \_kwh ;

electricityOf[Sellerstatus[0].seller] -= \_kwh ;

Sellerstatus[0].kwh -= \_kwh;

balanceOf[\_buyer] -= \_kwh\*Sellerstatus[0].value ;

balanceOf[Sellerstatus[0].seller] += \_kwh\*Sellerstatus[0].value;

delete Buyerstatus[Buyerstatus.length-1];

delete Sellerstatus[0];

for (uint b = 0 ; b < newSlength ; b++) {

Sellerstatus[b] = Sellerstatus[b+1];

delete Sellerstatus[b+1];

}

newSlength -= 1;

}

//ここもテスト

else if ( \_kwh > Sellerstatus[0].kwh) {

electricityOf[\_buyer] += Sellerstatus[0].kwh;

electricityOf[Sellerstatus[0].seller] -= Sellerstatus[0].kwh;

Buyerstatus[Buyerstatus.length -1].kwh -= Sellerstatus[0].kwh;

balanceOf[\_buyer] -= Sellerstatus[0].kwh \* Sellerstatus[0].value;

balanceOf[Sellerstatus[0].seller] += Sellerstatus[0].kwh \* Sellerstatus[0].value;

delete Sellerstatus[0];

for (uint b = 0 ; b < newSlength ; b++) {

Sellerstatus[b] = Sellerstatus[b+1];

delete Sellerstatus[b+1];

}

newSlength -= 1;

for ( uint b = newBlength ; b > 0 ; b--) {

Buyerstatus[b] = Buyerstatus[b-1];

}

Buyerstatus[0] = Buyerstatus[Buyerstatus.length-1];

newBlength += 1 ;

delete Buyerstatus[Buyerstatus.length-1];

}

//ここまで

}

if ( \_value > agreementeth) {

//セラーの一人目だけで取引が済む場合

if ( \_kwh < Sellerstatus[0].kwh ) {

electricityOf[\_buyer] += \_kwh ;

electricityOf[Sellerstatus[0].seller] -= \_kwh ;

Sellerstatus[0].kwh -= \_kwh;

balanceOf[\_buyer] -= \_kwh\*Sellerstatus[0].value ;

balanceOf[Sellerstatus[0].seller] += \_kwh\*Sellerstatus[0].value;

delete Buyerstatus[Buyerstatus.length-1];

}

//セラー1人ではバイヤーの需要に答えられない場合(未完成)

if ( \_kwh > Sellerstatus[0].kwh ) {

//Sellerの電力数え上げのための変数定義

uint x = 0 ;

for ( uint32 b = 0 ; b < newSlength ; b++) {

x += Sellerstatus[b].kwh ;

electricityOf[Sellerstatus[b].seller] -= Sellerstatus[b].kwh;

balanceOf[Sellerstatus[b].seller] += Sellerstatus[b].kwh\*Sellerstatus[b].value;

balanceOf[\_buyer] -= Sellerstatus[b].kwh\*Sellerstatus[b].value;

Sellerstatus[b].kwh = 0 ;

agreementeth = Sellerstatus[0].value;

electricityOf[\_buyer] += \_kwh ;

if (\_kwh <= x) {

Sellerstatus[b].kwh = x - \_kwh;

electricityOf[Sellerstatus[b].seller] += Sellerstatus[b].kwh;

balanceOf[Sellerstatus[b].seller] -= Sellerstatus[b].kwh\*Sellerstatus[b].value;

balanceOf[\_buyer] += Sellerstatus[b].kwh\*Sellerstatus[b].value;

break ;

}

//ここからテスト

if (Sellerstatus[b].value == \_value ) {

Buyerstatus[Buyerstatus.length - 1].kwh = \_kwh - x;

for ( uint j = newBlength; j > 0 ; j--) {

Buyerstatus[j]=Buyerstatus[j-1];

}

Buyerstatus[0]=Buyerstatus[Buyerstatus.length-1];

newBlength += 1;

break;

}

//ここまで

delete Buyerstatus[Buyerstatus.length-1];

}

}

//sellerの整理

uint32 a = 0;

for ( a ; a < newSlength; a++) {

if (Sellerstatus[a].kwh == 0) {

delete Sellerstatus[a];

}

if (Sellerstatus[a].kwh != 0) {

break;

}

}

for (uint32 b = 0; b < newSlength; b++){

Sellerstatus[b] = Sellerstatus[b+a];

delete Sellerstatus[b+a];

}

newSlength -= a ;

}

if (\_value < agreementeth) {

//板状にある価格がプッシュされた場合順番を整理する

if (\_value <= Buyerstatus[newBlength - 1].value) {

Buyerstatus[newBlength] = Buyerstatus[Buyerstatus.length -1];

newBlength += 1;

}

//新規追加項目

else if (\_value > Buyerstatus[0].value){

for (uint b = newBlength; b > 0 ; b--) {

Buyerstatus[b]=Buyerstatus[b-1];

}

Buyerstatus[0]=Buyerstatus[Buyerstatus.length -1];

newBlength += 1;

}

//

else {

uint32 x = 0 ;

for (uint32 b = 0; b < newBlength; b++) {

x++ ;

if ( \_value == Buyerstatus[b].value ) {

break;

}

}

for ( uint b = newBlength; b > x ; b--) {

Buyerstatus[b]=Buyerstatus[b-1];

}

Buyerstatus[x]=Buyerstatus[Buyerstatus.length-1];

newBlength += 1;

}

delete Buyerstatus[Buyerstatus.length-1];

}

}

function entryForSeller(address \_seller, uint256 \_kwh, uint256 \_value, uint256 \_all, uint \_totalCoin) public

{

balanceOf[\_seller] = \_totalCoin;

electricityOf[\_seller] = \_all;

if (electricityOf[\_seller] < \_kwh){

revert();

}

Sellerstatus.push(SellerStatus({

seller: \_seller,

kwh: \_kwh,

value: \_value,

sum: \_kwh,

totalCoin:\_totalCoin

}));

agreementeth = Buyerstatus[0].value;

if ( \_value == agreementeth) {

//バイヤーの一人目だけで取引が済む場合

if ( \_kwh < Buyerstatus[0].kwh ) {

electricityOf[\_seller] -= \_kwh ;

electricityOf[Buyerstatus[0].buyer] += \_kwh ;

Buyerstatus[0].kwh -= \_kwh;

balanceOf[\_seller] += \_kwh\*Buyerstatus[0].value ;

balanceOf[Buyerstatus[0].buyer] -= \_kwh\*Buyerstatus[0].value;

delete Sellerstatus[Sellerstatus.length-1];

}

else if ( \_kwh == Buyerstatus[0].kwh ) {

electricityOf[\_seller] -= \_kwh ;

electricityOf[Buyerstatus[0].buyer] += \_kwh ;

Buyerstatus[0].kwh -= \_kwh;

balanceOf[\_seller] += \_kwh\*Buyerstatus[0].value ;

balanceOf[Buyerstatus[0].buyer] -= \_kwh\*Buyerstatus[0].value;

delete Sellerstatus[Sellerstatus.length-1];

delete Buyerstatus[0];

for (uint b = 0 ; b < newBlength ; b++) {

Buyerstatus[b] = Buyerstatus[b+1];

delete Buyerstatus[b+1];

}

}

//バイヤー1人ではセラーの需要に答えられない場合

else if ( \_kwh > Buyerstatus[0].kwh ) {

electricityOf[\_seller] -= Buyerstatus[0].kwh;

electricityOf[Buyerstatus[0].buyer] += Buyerstatus[0].kwh;

Sellerstatus[Sellerstatus.length -1].kwh -= Buyerstatus[0].kwh;

balanceOf[\_seller] += Buyerstatus[0].kwh \* Buyerstatus[0].value;

balanceOf[Buyerstatus[0].buyer] -= Buyerstatus[0].kwh \* Buyerstatus[0].value;

delete Buyerstatus[0];

for (uint b = 0 ; b < newBlength ; b++) {

Buyerstatus[b] = Buyerstatus[b+1];

delete Buyerstatus[b+1];

}

newBlength -= 1;

for ( uint b = newSlength ; b > 0 ; b--) {

Sellerstatus[b] = Sellerstatus[b-1];

}

Sellerstatus[0] = Sellerstatus[Sellerstatus.length-1];

newSlength += 1 ;

delete Sellerstatus[Sellerstatus.length-1];

}

}

if (\_value < agreementeth) {

//セラーの一人目だけで取引が済む場合

if ( \_kwh < Buyerstatus[0].kwh ) {

electricityOf[\_seller] -= \_kwh ;

electricityOf[Buyerstatus[0].buyer] += \_kwh ;

Buyerstatus[0].kwh -= \_kwh;

balanceOf[\_seller] += \_kwh\*Buyerstatus[0].value ;

balanceOf[Buyerstatus[0].buyer] -= \_kwh\*Buyerstatus[0].value;

delete Sellerstatus[Sellerstatus.length-1];

}

//セラー1人ではバイヤーの需要に答えられない場合(未完成)

if ( \_kwh > Buyerstatus[0].kwh ) {

//Sellerの電力数え上げのための変数定義

uint x = 0 ;

for ( uint32 b = 0 ; b < newBlength ; b++) {

x += Buyerstatus[b].kwh ;

electricityOf[Buyerstatus[b].buyer] += Buyerstatus[b].kwh;

balanceOf[Buyerstatus[b].buyer] -= Buyerstatus[b].kwh\*Buyerstatus[b].value;

balanceOf[\_seller] += Buyerstatus[b].kwh\*Buyerstatus[b].value;

Buyerstatus[b].kwh = 0 ;

agreementeth = Buyerstatus[b].value;

electricityOf[\_seller] -= \_kwh ;

if (\_kwh <= x) {

Buyerstatus[b].kwh = x - \_kwh;

electricityOf[Buyerstatus[b].buyer] -= Buyerstatus[b].kwh;

balanceOf[Buyerstatus[b].buyer] += Buyerstatus[b].kwh\*Buyerstatus[b].value;

balanceOf[\_seller] -= Buyerstatus[b].kwh\*Buyerstatus[b].value;

break ;

}

//ここからテスト

if (Buyerstatus[b].value == \_value ) {

Sellerstatus[Sellerstatus.length - 1].kwh = \_kwh - x;

for ( uint j = newSlength; j > 0 ; j--) {

Sellerstatus[j]=Sellerstatus[j-1];

}

Sellerstatus[0]=Sellerstatus[Sellerstatus.length-1];

newSlength += 1;

break;

}

//ここまで

delete Sellerstatus[Sellerstatus.length-1];

}

}

//sellerの整理

uint32 a = 0;

for ( a ; a < newBlength; a++) {

if (Buyerstatus[a].kwh == 0) {

delete Buyerstatus[a];

}

if (Buyerstatus[a].kwh != 0) {

break;

}

}

for (uint32 b = 0; b < newBlength; b++){

Buyerstatus[b] = Buyerstatus[b+a];

delete Buyerstatus[b+a];

}

newBlength -= a ;

}

if (\_value > agreementeth) {

//板状にある価格がプッシュされた場合順番を整理する

if (\_value >= Sellerstatus[newSlength - 1].value) {

Sellerstatus[newSlength] = Sellerstatus[Sellerstatus.length -1];

newSlength += 1;

}

else if (\_value < Sellerstatus[0].value){

for (uint b = newSlength; b > 0 ; b--) {

Sellerstatus[b]=Sellerstatus[b-1];

}

Sellerstatus[0]=Sellerstatus[Sellerstatus.length -1];

newSlength += 1;

}

else {

uint32 x = 0 ;

for (uint32 b = 0; b < newSlength; b++) {

x++ ;

if ( \_value == Sellerstatus[b].value ) {

break;

}

}

for ( uint b = newSlength; b > x ; b--) {

Sellerstatus[b]=Sellerstatus[b-1];

}

Sellerstatus[x]=Sellerstatus[Sellerstatus.length-1];

newSlength += 1;

}

delete Sellerstatus[Sellerstatus.length -1 ];

}

}

}