**JUSTIFICACIÓ D’ALGORITMES DE LA PRÀCTICA DE PRO2**

1. **Algoritme iteratiu -** **City::TradeWith(City& other) a City.cc**

Nota: El codi es pot trobar després de la justificació.

**Precondició:** Les dues ciutats (*this* i *other*) són ciutats vàlides.

**Postcondició:** Les dues ciutats han fet tots els intercanvis possibles de productes.

**Invariant:** els dos iteradors **this\_product\_it** i **other\_product\_it** apunten a posicions vàlides de memòria.

**Funció de fita:** Com que els elements d’un set no tenen cap *índex* dins del conjunt, donar la funció de fita com a una equació és complicat. No obstant, internament un set ordena els seus elements, pel que utilitzarem el mètode fictici **getpos(iterator)** per a referir-nos a la posició interna d’un element al set, començant pel zero i acabant a la posició mida-1. Dit això, la funció de fita és

**min(this\_product\_set.size() - this\_product\_set.getpos(this\_product\_it) - 1,**

**other\_product\_set.size() - other\_product\_set.getpos(other\_product\_it) - 1)**

**Justificació de l’algoritme:**

Suposant que la precondició es compleix, les dues ciutats contenen un inventari vàlid, i, per tant, mitjançant el mètode **City::GetRawProductIds()**, podem obtenir un ***set<int>*** amb els productes disponibles de la ciutat. D’aquest set n’obtindrem l’iterador, que ens permetrà accedir de forma ordenada als identificadors dels productes disponibles de cada ciutat, de menor a major. Els dos iteradors, que apunten cadascun a l’inventari de la seva ciutat, apuntaran al primer element del set.

En cas que l’invariant no es compleixi, i algun dels iteradors apunti d’entrada a una posició no vàlida de la memòria (és a dir, a la posició **end()** del set), voldrà dir que alguna de les ciutats no té cap producte disponible, i, en conseqüència, tots els intercanvis possibles (que son zero) han estat fets, i, per tant, la postcondició es compleix. No cal executar cap iteració del bucle, i es pot finalitzar l’execució.

En cas que l’invariant es compleixi, i mentre l’invariant es compleixi, els dos iteradors apuntaran a dos identificadors de producte. Llavors, es poden donar 3 situacions:

**1) El primer identificador és menor que el segon identificador**: Això vol dir que, com que els identificadors estan ordenats en ordre ascendent i els iteradors apunten inicialment al valor més petit, la segona ciutat no conté el primer producte. Si la segona ciutat contingués el primer producte, el seu identificador hauria hagut d’aparèixer abans de l’identificador de producte actual de la segona ciutat, cosa que no ha fet. Per tant, es desplaça l’iterador de la primera ciutat a la següent posició,

**2) El primer identificador és major que el segon identificador**: S’aplica el mateix raonament que al cas 1), però amb els papers canviats. Per tant, es desplaça l’iterador de la segona ciutat a la següent posició.

**3) Els dos productes coincideixen:** Això significa que les dues ciutats contenen el mateix producte, pel que és possible que el puguin comerciar. Per tant, s’avaluen els dos escenaris possibles: que la primera ciutat li vengui producte a la segona i vici-versa.

*Nota: Es sobreentén que parlem del producte l’identificador del qual coincideix amb el del valor dels iteradors.*

Per a comprovar si la primera ciutat li pot vendre producte a la segona ciutat, es calcula el mínim entre la quantitat de producte excedent de la primera ciutat i la quantitat de producte necessitat per la segona ciutat. Si aquest nombre és major que zero es procedeix a fer l’intercanvi, restant-li a la primera ciutat la quantitat prèviament calculada del producte, i sumant-li a la segona ciutat.

En cas que no s’hagi produït venda, també es comprova si la segona ciutat li pot vendre producte a la primera, i es repeteix el procediment mencionat al paràgraf anterior intercanviant el paper de les dues ciutats.

Finalment, es desplacen els dos iteradors a les seves respectives següents posicions, ja que el producte que s’acaba d’avaluar no s’ha de tornar a comprovar, ja que s’acaben de fer tots els intercanvis possibles amb ell. *Aquí acaba la tercera situació.*

En els tres casos, com a mínim un dels dos iteradors s’ha incrementat, augmentant la seva posició relativa dins del set. Per tant, la funció de fita s’ha reduït. Llavors, el bucle torna al principi i torna a comprovar la condició.

Mentre es compleixi l’invariant hi haurà possibles combinacions de productes per a comerciar, i quan es deixi de complir, que voldrà dir que una de les ciutats ja no té més productes disponibles al seu inventari per a comerciar, ja que ja s’hauran comprovat tots, i la postcondició es complirà.

**Codi de la funció:**

**void City::TradeWith(City& other)**

**{**

**auto& this\_product\_set = this->GetRawProductIds();**

**auto& other\_product\_set = other.GetRawProductIds();**

**auto this\_product\_it = this\_product\_set.begin();**

**auto other\_product\_it = other\_product\_set.begin();**

**while(this\_product\_it != this\_product\_set.end()**

**&& other\_product\_it != other\_product\_set.end())**

**{**

**if(\*this\_product\_it > \*other\_product\_it)**

**other\_product\_it++;**

**else if(\*this\_product\_it < \*other\_product\_it)**

**this\_product\_it++;**

**else //(\*this\_product\_it == \*other\_product\_it)**

**{**

**int product\_id = \*this\_product\_it;**

**// this -> SELLER**

**// other -> BUYER**

**int trade\_amount = min(**

**this->GetProductExceedingAmount(product\_id),**

**other.GetProductMissingAmount(product\_id)**

**);**

**if(trade\_amount != 0)**

**{**

**this->WithdrawProductAmount(product\_id, trade\_amount);**

**other.RestockProductAmount(product\_id, trade\_amount);**

**}**

**else**

**{**

**// other -> SELLER**

**// this -> BUYER**

**trade\_amount = min(**

**other.GetProductExceedingAmount(product\_id),**

**this->GetProductMissingAmount(product\_id)**

**);**

**if(trade\_amount != 0)**

**{**

**other.WithdrawProductAmount(product\_id, trade\_amount);**

**this->RestockProductAmount(product\_id, trade\_amount);**

**}**

**}**

**this\_product\_it++;**

**other\_product\_it++;**

**}**

**}**

**}**

1. **Algoritme recursiu**