L'algorisme implementat per a la resolució de QAP és el proposat a la pàgina web de l'assignatura, seguint els passos de [https://www.cs.upc.edu/prop/data/uploads/prop_q1_2023_2024.pdf]. En aquest apartat de documentació es comenten alguns dels aspectes de disseny de l'algorisme més importants, així com una descripció general del seu funcionament. Per a consultar informació més en profunditat es poden consultar els comentaris disponibles al codi font.

L'algorisme implementat s'ha separat en tres classes diferenciades: *BranchBound*, *Hungarian* i *CompleteAssignation*.

class *BranchBound*

És la classe encarregada d'executar l'algorisme de branching, i retornar una solució a partir de la qual es genera el teclat.

Branching: Per a l'algorisme de branching s'ha escollit una estratègia similar a *eager*, on anem desenvolupant l'arbre de solucions fulla per fulla, en funció de la solució parcial que tenim i dels caràcters que queden per emplaçar en una tecla.

L'algorisme parteix d'una solució parcial (la qual pot estar o no buida), i calcula el cost d'afegir tots els caràcters no emplaçats a la següent tecla disponible. Escull el caràcter que ha proporcionat un cost més baix, actualitza la solució parcial afegint el nou caràcter a la següent posició i repeteix el bucle fins a tenir tots els caràcters emplaçats en una tecla.

S'ha optimitzat la velocitat de l'algoritme utilitzant una estratègia *greedy*. Després de realitzar múltiples proves, s'ha optat per a inicialitzar la solució parcial amb els 4 caràcters més freqüents a les primeres posicions del teclat. D'aquesta forma es redueix el temps d'execució de l'Hungarian Algorithm, ja que disminueix la mida de les matrius amb les que ha de treballar. També ofereix cotes amb valors molt baixos (com més baixos millor), ja que la distribució del teclat s'ha dissenyat de forma que les primeres posicions de la solució parcial corresponen a les tecles més cèntriques en el teclat, així que assignem per defecte els caràcters més freqüents a les tecles que són accessibles més fàcilment.

Bounding: En aquest cas, la funció de bounding que implementa l'algorisme és la cota Gilmore, descrita al PDF. El primer terme de la cota de Gilmore es pot calcular de forma exacta, però per als termes 2 i 3 utilitzem una aproximació calculada mitjançant Hungarian. Cal destacar que per a realitzar el càlcul de la cota donada una solució parcial són necessàries les classes Hungarian i CompleteAssignation descrites posteriorment.

Cal destacar que en la solució proposada es tenen en compte les freqüències de caràcters en ambdós sentits, és a dir, la freqüència en que el caràcter 'a' va després de 'b', i també la freqüència en què el caràcter 'b' va després de 'a'. Per a calcular el cost de l'aresta entre dos caràcters 'a' i 'b', situats en dues tecles diferents, calculem la mitjana de la freqüència A→B i B→A, i la multipliquem per la distància entre les dues tecles on es troben emplaçats els caràcters.

Estructures de dades: Per a obtenir les freqüències i distàncies corresponents treballem amb arrays multidimensionals (matrius). El tipus de dades clau utilitzat en aquest algoritme és també el ArrayList, gràcies a la flexibilitat que ofereix a l'hora d'afegir, modificar i consultar valors emmagatzemats.

Cal comentar també que la tant les solucions parcials com la solució final de l'algorisme s'implementen en forma d'ArrayList. Per a interpretar un ArrayList solució: l'índex de l'element de llista representa l'ID de la tecla on es troba un caràcter, el valor de l'element d'una llista indica l'ID del caràcter.

Exemple: Solució Parcial {3, 1, 0} → En aquesta solució parcial, el caràcter amb ID=3 es troba a l'índex 0, per tant emplaçat a la tecla 0. De la mateixa forma, el caràcter ID=1 es trobarà a la tecla 1 i el caràcter ID=0 a la tecla 2.

class *Hungarian*

És la classe encarregada d'executar l'Hungarian Algorithm, algorisme el qual a partir d'una matriu ens ofereix una assignació òptima, a partir de la qual obtenim una aproximació del càlcul dels termes 2 i 3 de la cota de Gilmore. Requereix de la classe CompleteAssignation.

Implementació de l'algorisme: l'Hungarian Algorithm s'ha implementat seguint els passos descrits en el PDF proporcionat. S'ha separat el càlcul de l'assignació més completa possible en una altra classe degut a la seva complexitat. El resultat proporcionat per l'Hungarian és en forma d'ArrayList, on l'índex de cada element representa la fila i el valor representa la columna on es troba un zero de l'assignació.

class *CompleteAssignation*

És la classe encarregada de trobar la millor assignació possible donada una matriu, s'utilitza tant en passos intermitjos de l'Hungarian com per a trobar l'assignació completa final. La seva implementació és crítica per a l'eficiència de l'algorisme.

Implementació de l'algorisme: Per a trobar la millor assignació possible s'ha d'implementar un algorisme de *backtracking*, el qual és recursiu. Degut al gran nombre de combinacions que ha de processar aquest esdevé el coll d'ampolla de tot l'algorisme que l'implementa.