Algorithme hongrois

1. Le pseudo-code:

(Main)

#include <stdio.h>

#include <stdlib.h>

#define INFINI 1000

#include "lect\_prob.h"

#include "Hongrois.h"

#define data "instance.txt"*//"instance.txt"*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

void main**()**

**{**

int i**,**j**;**

**struct** probleme **\***prob**;**

**struct** mat\_cout **\*\***mat\_init**;**

prob**=**lecture**(**data**);**

mat\_init**=**mat\_ini**(**prob**->**cout\_affec**,**prob**);**

printf**(**"La matrice initiale est : \n"**);**

Affiche\_matrice**(**mat\_init**,**prob**);**

printf**(**"\n"**);**

first\_step**(**prob**,**prob**->**cout\_affec**,**prob**->**n**);**

printf**(**"La matrice apres retranchement des min sur les lignes est :\n"**);**

Affiche\_matrice**(**prob**->**cout\_affec**,**prob**);**

printf**(**"\n"**);**

**if(**firstest\_opt**(**prob**->**cout\_affec**,**prob**)==**0**)**

**{**

    final\_step**(**prob**->**cout\_affec**,**mat\_init**,**prob**);**

**}else** first\_sol\_finale**(**prob**->**cout\_affec**,**mat\_init**,**prob**);**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

[Hongrois.h](https://codes-sources.commentcamarche.net/source/view/46664/1152907#browser)

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*/\*fonction retournant le min d'une colonne\*/*

int min\_col **(struct** mat\_cout **\*\***mat**,** int col**,**int n**)**

**{**

    int i**;**

    int min**=**0**;**

    min**=**mat**[**0**][**col**].**cout**;** *//initialisation du cout min de la colonne*

**for(**i**=**1**;**i**<**n**;**i**++)**

**{**

**if** **(**mat**[**i**][**col**].**cout**<**min**)**

                min**=**mat**[**i**][**col**].**cout**;** *//nouveau cout minimum*

**}**

**return** min**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*//fonction retournant la matrice aprés retrait du cout min d'une colonne donnée*

void col\_modif**(struct** mat\_cout **\*\***mat**,**int col**,**int n**,**int min**)**

**{**

int i**;**

**for(**i**=**0**;**i**<**n**;**i**++)**

**if** **(**mat**[**i**][**col**].**cout**!=**INFINI**)**

        mat**[**i**][**col**].**cout**=**mat**[**i**][**col**].**cout**-**min**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*//fonction retournant la matrice aprés retrait du cout min de toutes les colonnes*

void cols\_modif**(struct** mat\_cout **\*\***mat**,**int n**)**

**{**

int i**,**min**=**0**;**

**for(**i**=**0**;**i**<**n**;**i**++)**

**{**

    min**=**min\_col **(**mat**,** i**,**n**);**

    col\_modif**(**mat**,**i**,**n**,**min**);**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*/\*fonction retournant le min d'une ligne donnée\*/*

int min\_lign **(struct** mat\_cout **\*\***mat**,**int lign**,**int n**)**

**{**

    int i**;**

    int min**=**0**;**

    min**=**mat**[**lign**][**0**].**cout**;** *//initialisation du cout min de la ligne*

**for(**i**=**1**;**i**<**n**;**i**++)**

**{**

**if** **(**mat**[**lign**][**i**].**cout**<**min**)**

                min**=**mat**[**lign**][**i**].**cout**;** *//nouveau cout minimum*

**}**

**return** min**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*//fonction retournant la matrice aprés retrait de coût min d'une ligne donnée*

void lign\_modif**(struct** mat\_cout **\*\***mat**,**int lign**,**int n**,**int min**)**

**{**

int i**;**

**for(**i**=**0**;**i**<**n**;**i**++)**

**{**

**if** **(**mat**[**lign**][**i**].**cout**!=**INFINI**)**

        mat**[**lign**][**i**].**cout**=**mat**[**lign**][**i**].**cout**-**min**;**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*//fonction retournant la matrice aprés retrait du cout min dans toutes les lignes*

void ligns\_modif**(struct** mat\_cout **\*\***mat**,**int n**)**

**{**

int i**,**min**=**0**;**

**for(**i**=**0**;**i**<**n**;**i**++)**

**{**

        min**=**min\_lign **(**mat**,** i**,**n**);**

        lign\_modif**(**mat**,**i**,**n**,**min**);**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

void first\_step**(struct** probleme **\***prob**,struct** mat\_cout **\*\***mat**,**int n**)**

**{**

cols\_modif**(**mat**,**n**);**

printf**(**"La matrice apres retranchement des min sur les colonnes est :\n"**);**

Affiche\_matrice**(**mat**,**prob**);**

printf**(**"\n"**);**

ligns\_modif**(**mat**,**n**);**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*/\*fonction retournant la ligne contenant le nb min de zéro\*/*

int lign\_min **(struct** probleme **\***prob**)**

**{**

    int i**,**no\_lign**;**

    int min**=**INFINI**;**

    no\_lign**=**0**;**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**if** **((**prob**->**etat**[**i**].**o\_lign**<**min**)&&(**prob**->**etat**[**i**].**o\_lign**!=**0**))**

**{**

                min**=**prob**->**etat**[**i**].**o\_lign**;** *//nouveau ligne minimale*

                no\_lign**=**i**;**

**}**

**}**

**return** no\_lign**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*///rechercher le nb de zéro dans chaque lign et dans chaque col/////*

void firstab\_nbzero**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**)**

**{**

int i**,**j**;**

**for** **(**i**=**0**;** i**<**prob**->**n**;**i**++)**

**{**

        prob**->**etat**[**i**].**o\_lign**=**0**;**

        prob**->**etat**[**i**].**o\_col**=**0**;**

**}**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**for(**j**=**0**;**j**<**prob**->**n**;**j**++)**

**{**

**if** **(**mat**[**i**][**j**].**cout**==**0**)**

**{**

             prob**->**etat**[**i**].**o\_lign**++;**

             prob**->**etat**[**j**].**o\_col**++;**

**}**

**}**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

void find\_nbzero**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**)**

**{**

int i**,**j**;**

**for** **(**i**=**0**;** i**<**prob**->**n**;**i**++)**

**{**

        prob**->**etat**[**i**].**o\_lign**=**0**;**

        prob**->**etat**[**i**].**o\_col**=**0**;**

**}**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**for(**j**=**0**;**j**<**prob**->**n**;**j**++)**

**{**

**if** **((**mat**[**i**][**j**].**cout**==**0**)&&(**mat**[**i**][**j**].**encadre**==**1**))**

**{**

             prob**->**etat**[**i**].**o\_lign**++;**

             prob**->**etat**[**j**].**o\_col**++;**

**}**

**}**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

int firstest\_opt**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**)**

**{**

int i**,**j**,**cpt**=**0**,**cptr**=**0**;**

firstab\_nbzero**(**mat**,**prob**);**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**if(**prob**->**etat**[**i**].**o\_lign**==**1**)** cpt**++;**

**if(**prob**->**etat**[**i**].**o\_col**==**1**)** cptr**++;**

**}**

**if((**cpt**==**prob**->**n**)&&(**cptr**==**prob**->**n**))**

**return** 1**;**

**return** 0**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

int test\_opt**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**)**

**{**

int i**,**j**,**cpt**=**0**,**cptr**=**0**;**

find\_nbzero**(**mat**,**prob**);**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**if(**prob**->**etat**[**i**].**o\_lign**==**1**)** cpt**++;**

**if(**prob**->**etat**[**i**].**o\_col**==**1**)** cptr**++;**

**}**

**if((**cpt**==**prob**->**n**)&&(**cptr**==**prob**->**n**))**

**return** 1**;**

**return** 0**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

void search\_affec**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**)**

**{**

int cpt**,**i**,**j**,**lignemin**,**indi\_o**,**fine**;**

int termine**;**

**do**

**{**

lignemin**=**lign\_min**(**prob**);**

fine**=**0**;**

j**=**0**;**

**if(**prob**->**etat**[**lignemin**].**o\_lign**!=**0**)**

**{**

**do**

**{**

**if** **((**mat**[**lignemin**][**j**].**cout**==**0**)&&(**mat**[**lignemin**][**j**].**normal**==**0**))** j**++;**

**else**

**{**

**if** **((**mat**[**lignemin**][**j**].**cout**==**0**)&&(**mat**[**lignemin**][**j**].**normal**==**1**))**

**{**

            mat**[**lignemin**][**j**].**encadre**=**1**;**

            mat**[**lignemin**][**j**].**normal**=**0**;**

            indi\_o**=**j**;**

            prob**->**etat**[**lignemin**].**oenca\_lign**++;**

            prob**->**etat**[**indi\_o**].**oenca\_col**++;**

            prob**->**etat**[**lignemin**].**o\_lign**--;**

            prob**->**etat**[**indi\_o**].**o\_col**--;**

            fine**=**1**;**

**}**

         j**++;**

**}**

**}while(**fine**!=**1**);**

**}**

**if(**prob**->**etat**[**lignemin**].**o\_lign**!=**0**)**

**{**

**for(**j**=**indi\_o**;**j**<**prob**->**n**;**j**++)**

**if** **((**mat**[**lignemin**][**j**].**cout**==**0**)&&(**mat**[**lignemin**][**j**].**normal**==**1**))**

**{**

            mat**[**lignemin**][**j**].**barre**=**1**;**

            mat**[**lignemin**][**j**].**normal**=**0**;**

            prob**->**etat**[**lignemin**].**o\_lign**--;**

            prob**->**etat**[**lignemin**].**obarre\_lign**++;**

            prob**->**etat**[**j**].**obarre\_col**++;**

            prob**->**etat**[**j**].**o\_col**--;**

**}**

**}**

**if(**prob**->**etat**[**indi\_o**].**o\_col**!=**0**)**

**{**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**if((**mat**[**i**][**indi\_o**].**cout**==**0**)&&(**mat**[**i**][**indi\_o**].**normal**==**1**))**

**{**

        mat**[**i**][**indi\_o**].**barre**=**1**;**

        mat**[**i**][**indi\_o**].**normal**=**0**;**

        prob**->**etat**[**i**].**o\_lign**--;**

        prob**->**etat**[**indi\_o**].**o\_col**--;**

        prob**->**etat**[**i**].**obarre\_lign**++;**

        prob**->**etat**[**indi\_o**].**obarre\_col**++;**

**}**

**}**

termine**=**0**;**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**if(**prob**->**etat**[**i**].**o\_lign**==**0**)** termine**++;**

**}while(**termine**!=**prob**->**n**);**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

int min\_minitab**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**)**

**{**

int i**,**j**,**min**=**INFINI**;**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**while** **((**prob**->**etat**[**i**].**lign\_mark**!=**1**)&&(**i**<**prob**->**n**))** i**++;**

**if** **(**i**==**prob**->**n**)** **break;**

**for(**j**=**0**;**j**<**prob**->**n**;**j**++)**

**{**

**while** **((**prob**->**etat**[**j**].**col\_mark**==**1**)&&(**j**<**prob**->**n**))** j**++;**

**if(**j**!=**prob**->**n**)**    **if(**mat**[**i**][**j**].**cout**<**min**)**min**=**mat**[**i**][**j**].**cout**;**

**}**

**}**

**return** min**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

void mintab\_act**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**,**int min**)**

**{**

    int i**,**j**;**

*//retrancher la val de min aux col non barrées*

**for(**j**=**0**;**j**<**prob**->**n**;**j**++)**

**{**

**while** **((**prob**->**etat**[**j**].**col\_mark**==**1**)&&(**j**<**prob**->**n**))**

**{**

             j**++;**

**};**

**if(**j**!=**prob**->**n**)**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**if(**mat**[**i**][**j**].**cout**!=**INFINI**)**

                    mat**[**i**][**j**].**cout**-=**min**;**

**}**

*//ajouter la val de min aux lign barrées*

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**while** **((**prob**->**etat**[**i**].**lign\_mark**==**1**)&&(**i**<**prob**->**n**))**

**{**

             i**++;**

**};**

**if** **(**i**==**prob**->**n**)** **break;**

**for(**j**=**0**;**j**<**prob**->**n**;**j**++)**

**if(**mat**[**i**][**j**].**cout**!=**INFINI**)**

                    mat**[**i**][**j**].**cout**+=**min**;**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*détermination des affectations interressentes\*\*\*\*\*\*\*\*\*\*\*\*\*/*

int good\_affec**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**)**

**{**

int i**,**j**,**finish**,**min\_tab**=**0**,**termine**=**0**;**

*/////////////marquer les lignes qui ne contiennent aucun zéro/////////////*

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**if(**prob**->**etat**[**i**].**oenca\_lign**==**0**)**

**{**

*//printf("lign markee = %d \n",i+1);*

        prob**->**etat**[**i**].**lign\_mark**=**1**;**

        termine**++;**

**}**

**if** **(**termine**==**0**)** **return** 0**;**

**else**

**{**

**do**

**{**

finish**=**0**;**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**if(**prob**->**etat**[**i**].**obarre\_col**!=**0**)**

**{**

**for(**j**=**0**;**j**<**prob**->**n**;**j**++)**

**if((**mat**[**j**][**i**].**barre**==**1**)&&(**prob**->**etat**[**j**].**lign\_mark**==**1**)&&(**prob**->**etat**[**i**].**col\_mark**==**0**))**

**{**

        prob**->**etat**[**i**].**col\_mark**=**1**;**

        finish**++;**

*//printf("finish 1=%d\n",finish);*

**}**

**}**

**}**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**if(**prob**->**etat**[**i**].**oenca\_lign**!=**0**)**

**{**

**for(**j**=**0**;**j**<**prob**->**n**;**j**++)**

**if((**mat**[**i**][**j**].**encadre**==**1**)&&(**prob**->**etat**[**j**].**col\_mark**==**1**)&&(**prob**->**etat**[**i**].**lign\_mark**==**0**))**

**{**

        prob**->**etat**[**i**].**lign\_mark**=**1**;**

        finish**++;**

*//printf("finish 2=%d\n",finish);*

**}**

**}**

**}**

**}while** **((**finish**==**1**)||(**finish**==**2**));**

min\_tab**=**min\_minitab**(**mat**,**prob**);**

mintab\_act**(**mat**,**prob**,**min\_tab**);**

**}**

**return** 1**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*détermination des affectations interressentes\*\*\*\*\*\*\*\*\*\*\*\*\*/*

void sol\_finale**(struct** mat\_cout **\*\***mat**,struct** mat\_cout **\*\***mat\_ini**,struct** probleme **\***prob**)**

**{**

int i**,**j**,**sol**=**0**;**

int **\***sol\_vect**;**

sol\_vect**=(**int**\*)** malloc**(**prob**->**n **\*** **sizeof(**int**));**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**for(**j**=**0**;**j**<**prob**->**n**;**j**++)**

**{**

**if** **((**mat**[**i**][**j**].**cout**==**0**)&&(**mat**[**i**][**j**].**encadre**==**1**))**

**{**

                sol\_vect**[**i**]=**j**;**

*//printf("j = %d\n",j+1);*

**}**

**}**

**}**

printf**(**"la solution finale est :\n"**);**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

    printf**(**"%d "**,**mat\_ini**[**i**][**sol\_vect**[**i**]].**cout**);**

    sol**+=**mat\_ini**[**i**][**sol\_vect**[**i**]].**cout**;**

**if** **(**i**!=**prob**->**n-1**)** printf**(**"+ "**);**

**if** **(**i**==**prob**->**n-1**)**

**{**

        printf**(**"= "**);**

        printf**(**"%d"**,**sol**);**

        printf**(**"\n"**);**

**}**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

void first\_sol\_finale**(struct** mat\_cout **\*\***mat**,struct** mat\_cout **\*\***mat\_ini**,struct** probleme **\***prob**)**

**{**

int i**,**j**,**sol**=**0**;**

int **\***sol\_vect**;**

sol\_vect**=(**int**\*)** malloc**(**prob**->**n **\*** **sizeof(**int**));**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

**for(**j**=**0**;**j**<**prob**->**n**;**j**++)**

**{**

**if** **(**mat**[**i**][**j**].**cout**==**0**)**

                            sol\_vect**[**i**]=**j**;**

**}**

**}**

printf**(**"la solution finale est :\n"**);**

**for(**i**=**0**;**i**<**prob**->**n**;**i**++)**

**{**

    printf**(**"%d "**,**mat\_ini**[**i**][**sol\_vect**[**i**]].**cout**);**

    sol**+=**mat\_ini**[**i**][**sol\_vect**[**i**]].**cout**;**

**if** **(**i**!=**prob**->**n-1**)** printf**(**"+ "**);**

**if** **(**i**==**prob**->**n-1**)**

**{**

        printf**(**"= "**);**

        printf**(**"%d"**,**sol**);**

        printf**(**"\n"**);**

**}**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

void final\_step**(struct** mat\_cout **\*\***mat**,struct** mat\_cout **\*\***mat\_ini**,struct** probleme **\***prob**)**

**{**

int pass**=**0**;**

**do**

**{**

        printf**(**"<---Dans l'iteration no %d---> \n"**,**pass+1**);**

        init\_etat**(**prob**);**

        firstab\_nbzero**(**mat**,**prob**);**

        search\_affec**(**mat**,**prob**);**

        printf**(**"La matrice apres la recherche du maximum d'affectations est :\n"**);**

        Affiche\_matrice**(**mat**,**prob**);**

        printf**(**"\n"**);**

**if** **(**good\_affec**(**mat**,**prob**)==**0**)break;**

        printf**(**"La matrice apres la recherche des affectations interessentes est :\n"**);**

        Affiche\_matrice**(**mat**,**prob**);**

        printf**(**"\n"**);**

        pass**++;**

**}while(**test\_opt**(**mat**,**prob**)==**0**);**

sol\_finale**(**mat**,**mat\_ini**,**prob**);**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

[lect\_prob.h](https://codes-sources.commentcamarche.net/source/view/46664/1152908#browser)

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

**struct** tab\_etat

**{**

        int o\_lign**;***//indique le nombre de zéro dans la ligne i*

        int oenca\_lign**;**

        int obarre\_lign**;**

        int o\_col**;***//indique le nombre de zéro dans la colonne i*

        int oenca\_col**;**

        int obarre\_col**;**

        int lign\_mark**;***//inique si la ligne i est marquée d'un astèrisque*

        int col\_mark**;***//inique si la colonne i est marquée d'un astèrisque*

**};**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

**struct** mat\_cout

**{**

        int cout**;***//le cout de réalisation de la tàche i par la machine j*

        int normal**;***//prend 1 si oui, 0 sinon*

        int encadre**;***//prend 1 si oui, 0 sinon*

        int barre**;***//prend 1 si oui, 0 sinon*

**}** **;**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

**struct** probleme

**{**

            int n**;***//nb de tàches ou de machines*

**struct** mat\_cout **\*\***cout\_affec**;**

**struct** tab\_etat **\***etat**;** *//le nombre de zéros dans chaque ligne*

*//et dans chaque colonne, les lignes et les*

*//colonnes marqués*

**}** **;**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*//afficher la matrice : mat\_cout*

void Affiche\_matrice**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**)**

**{**

    int i**,**j**;**

**for** **(**i **=** 0**;** i **<** prob**->**n**;** i**++)**

**{**

**for(**j **=** 0**;** j **<** prob**->**n**;** j**++)**

                        printf**(**"%d \t"**,**prob**->**cout\_affec**[**i**][**j**].**cout**);**

*//printf("cout %d %d= %d \t",i+1,j+1,prob->cout\_affec[i][j].cout);*

                printf**(**"\n"**);**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

void init\_etat**(struct** probleme **\***prob**)**

**{**

int i**,**j**;**

prob**->**etat**=(struct** tab\_etat **\*)** malloc**(**prob**->**n **\*** **sizeof(struct** tab\_etat**));**

**for** **(**i**=**0**;** i**<**prob**->**n**;**i**++)**

**for(**j**=**0**;** j**<**prob**->**n**;** j**++)**

**{**

            prob**->**cout\_affec**[**i**][**j**].**normal**=**1**;**

            prob**->**cout\_affec**[**i**][**j**].**encadre**=**0**;**

            prob**->**cout\_affec**[**i**][**j**].**barre**=**0**;**

**}**

**for** **(**i**=**0**;** i**<**prob**->**n**;**i**++)**

**{**

        prob**->**etat**[**i**].**oenca\_lign**=**0**;**

        prob**->**etat**[**i**].**obarre\_lign**=**0**;**

        prob**->**etat**[**i**].**oenca\_col**=**0**;**

        prob**->**etat**[**i**].**obarre\_col**=**0**;**

        prob**->**etat**[**i**].**lign\_mark**=**0**;**

        prob**->**etat**[**i**].**col\_mark**=**0**;**

**}**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

**struct** probleme **\***creer\_pb**(**int n**)**

**{**

**struct** probleme **\***nouveau**;**

nouveau **=** **(struct** probleme **\*)**malloc**(sizeof(struct** probleme**));**

**if** **(**nouveau**!=**NULL**)**

**{**

nouveau**->**n**=**n**;**

**}**

**return** nouveau**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

**struct** probleme **\*** lecture**(**char **\*** ch**)**

**{**

FILE **\***ftp**;**

**struct** probleme **\***prob**;**

int i**,**j**,**n**;**

**if** **((**ftp**=**fopen**(**ch **,**"r"**))==**NULL**)**

**{**

    printf**(**"erreur!!! impossible d\'ouvrir le fichier voulu\n"**);**

**}**

**else**

**{**

fscanf**(**ftp**,**"%d"**,&**n**);**

printf**(**"le nombre de taches: %d \n"**,**n**);**

prob**=** creer\_pb**(**n**);**

*//////////////\*\*\*Allocation dynamique de la matrice\*\*\*//////////////*

prob**->**cout\_affec**=(struct** mat\_cout **\*\*)** malloc**(**prob**->**n **\*** **sizeof(struct** mat\_cout**\*));**

**for** **(**i **=** 0**;** i **<** prob**->**n**;** i**++)**

        prob**->**cout\_affec**[**i**]=** **(struct** mat\_cout **\*)** malloc**(**prob**->**n **\*** **sizeof(struct** mat\_cout**));**

**for** **(**i**=**0**;** i**<**prob**->**n**;**i**++)**

**for(**j**=**0**;** j**<**prob**->**n**;** j**++)**

            fscanf**(**ftp**,**"%d"**,&**prob**->**cout\_affec**[**i**][**j**].**cout**);**

**}**

fclose**(**ftp**);**

prob**->**etat**=(struct** tab\_etat **\*)** malloc**(**prob**->**n **\*** **sizeof(struct** tab\_etat**));**

**return** prob**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

**struct** mat\_cout **\*\*** mat\_ini**(struct** mat\_cout **\*\***mat**,struct** probleme **\***prob**)**

**{**

int i**,**j**;**

**struct** mat\_cout **\*\***mat\_init**;**

mat\_init**=(struct** mat\_cout **\*\*)** malloc**(**prob**->**n **\*** **sizeof(struct** mat\_cout**\*));**

**for** **(**i **=** 0**;** i **<** prob**->**n**;** i**++)**

        mat\_init**[**i**]=** **(struct** mat\_cout **\*)** malloc**(**prob**->**n **\*** **sizeof(struct** mat\_cout**));**

**for** **(**i**=**0**;** i**<**prob**->**n**;**i**++)**

**for(**j**=**0**;** j**<**prob**->**n**;** j**++)**

            mat\_init**[**i**][**j**]=**mat**[**i**][**j**];**

**return** mat\_init**;**

**}**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

     EXEMPLE

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

5  
4 9 90 8 65  
0 8 0 76 5  
54 8 0 9 9  
6 0 0 0 2  
65 98 5 4 1

Exemple

2-Utilité des différents paramètres:

On va donner à cet algorithme une matrice carré, là où les valeurs infinies restent inchangées qu’on va le traiter pour obtenir une solution optimale.

3- application de l’algorithme hongrois approché pour résoudre la problème d'optimisation

  C'est un algorithme qui permet de trouver un [couplage parfait](https://fr.wikipedia.org/wiki/Couplage_(th%C3%A9orie_des_graphes)) de poids maximum dans un [graphe biparti](https://fr.wikipedia.org/wiki/Graphe_biparti) dont les arêtes sont valuées. De façon alternative, il permet de trouver un couplage parfait de poids minimum dans un tel graphe.

L'algorithme procède en trois étapes successives.  
1- éliminer le plus petit élément de chacune des colonnes puis celui de chacune des lignes de la matrice des coûts.  
2- Si, cette solution n'est pas optimale, on considère la ligne ayant un nombre minimal de zéros. On encadre l'un des zéros de cette ligne, puis on barre les zéros qui se trouvent sur la même ligne ou la même colonne que le zéro encadré. On procède de même pour toutes les lignes.  
3- On marque toutes les lignes qui ne contiennent aucun zéro encadré  
On marque les lignes qui ont un zéro encadré dans une colonne marqué  
On marque les colonnes qui ont un ou plusieurs zéros barrés dans une ligne marquée.  
On répète ces trois marquages successifs jusqu'à ce que l'on ne puisse plus affecter de nouveaux marquages.