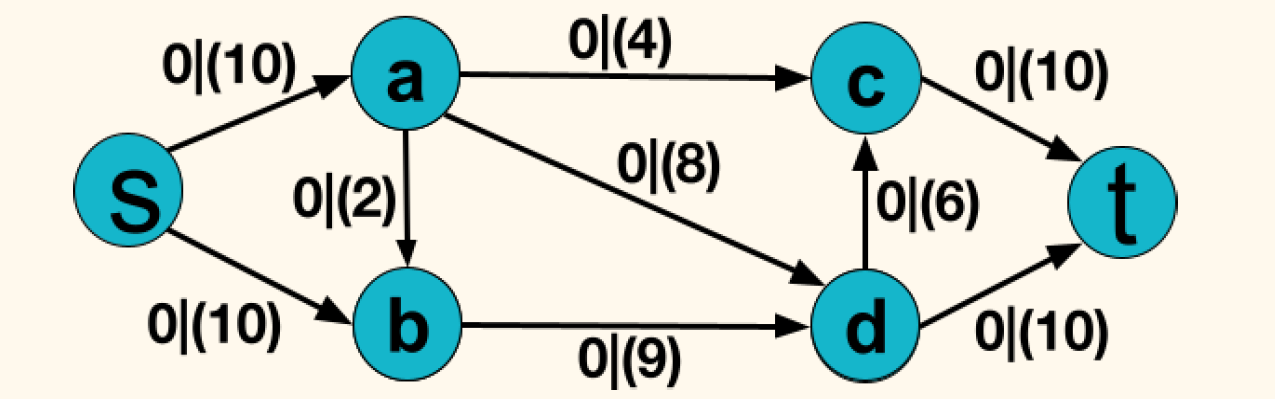
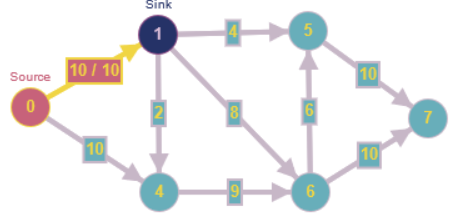
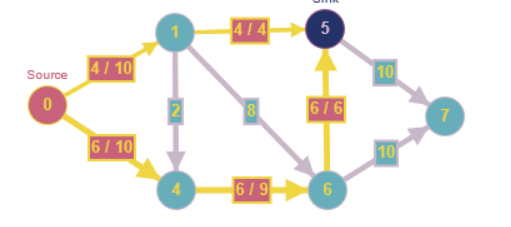
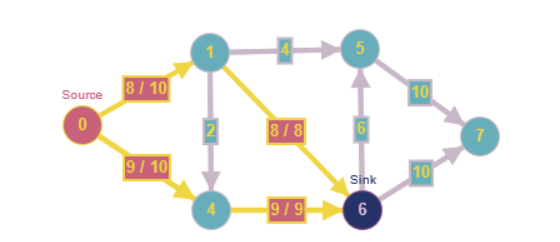
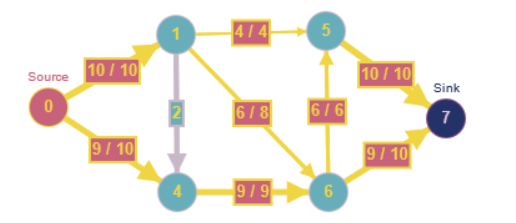
Rapport TD&P n°6

Partie 1 :   


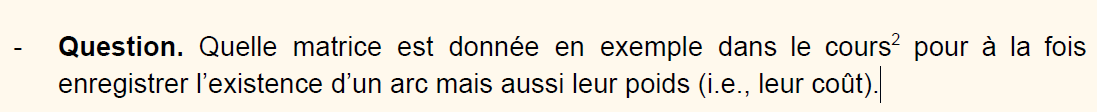
Flow : 10

Flow : 10

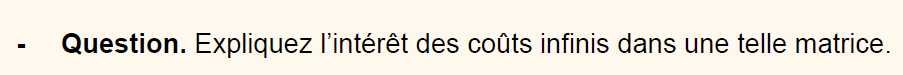
Flow : 17

Flow : 19

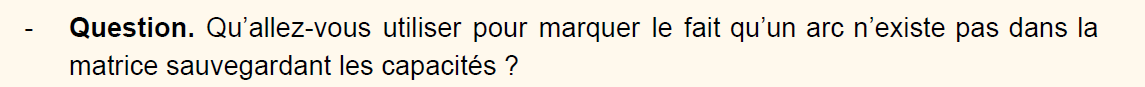
Partie 2 :



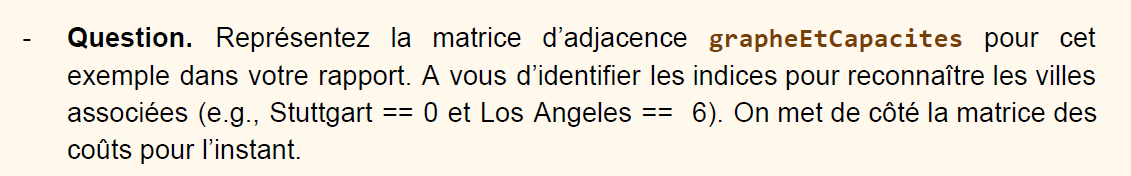
C’est la matrice cheminCourt[GrapheSize][GrapheSize]. Elle de la même taille que arcs et elle sauvegarde la distance totale du plus court chemin pour chaque couple de sommet (I.e cheminCourt[sommet1][sommet2] => distance la plus courte entre 2 sommets).



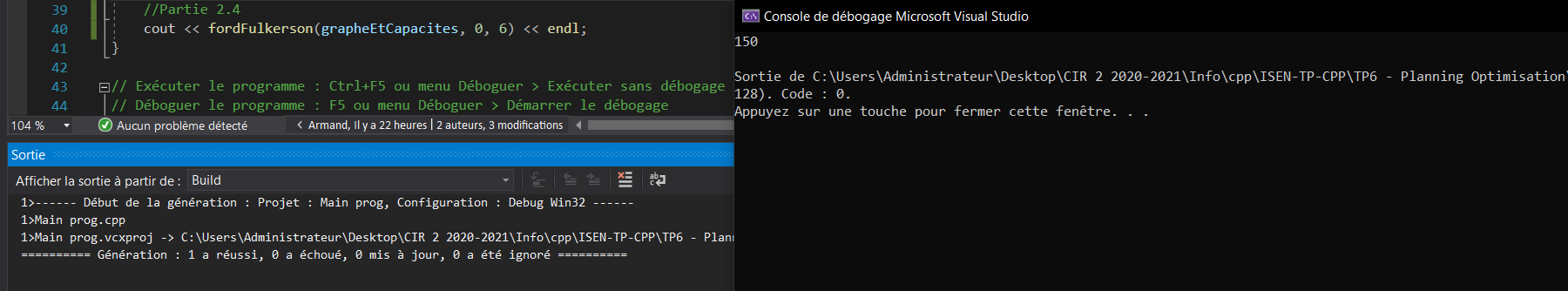
Les coups infinis ayant une valeur démesurée, n’importe quel chemin peu importe sa taille va être sauvegardé dans la matrice cheminCourt pour par la suite être trié parmi les autres. L’intérêt est alors que s’il existe un chemin en l’indice 0 et l’indice 4 par exemple, on aura une valeur différente de « l’infini » ce qui prouvera son existence.



L’arc serait initialisé en ayant des valeurs infinies partout si bien que quand il passera dans l’algorithme de Ford-Fulkerson (ff), si aucun chemin n’existe ces valeurs ne seront pas modifiées et facilement repérable.

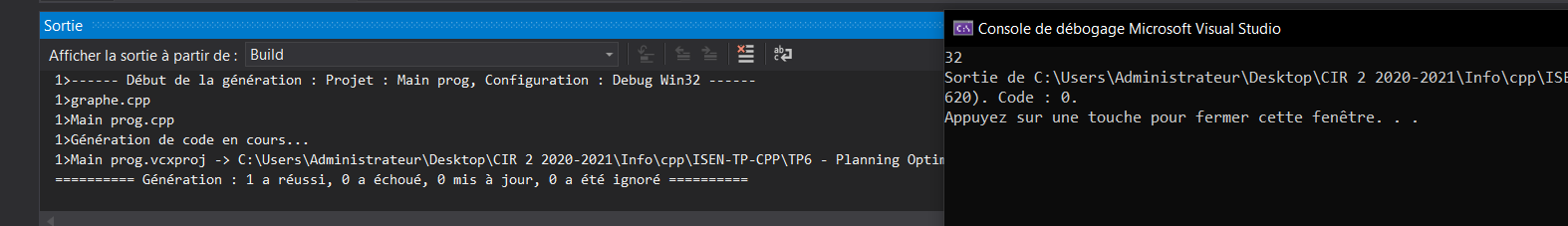
Légende : 0 = Stuttgart ; 1 = Rotterdam ; 2 = Bordeaux ; 3 = Lisbonne ; 4 = New-York ; 5 = New Orleans ; 6 = Los Angeles.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Départ/Destination | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 | 0 | 50 | 70 | 40 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 60 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 40 | 50 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 80 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 70 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

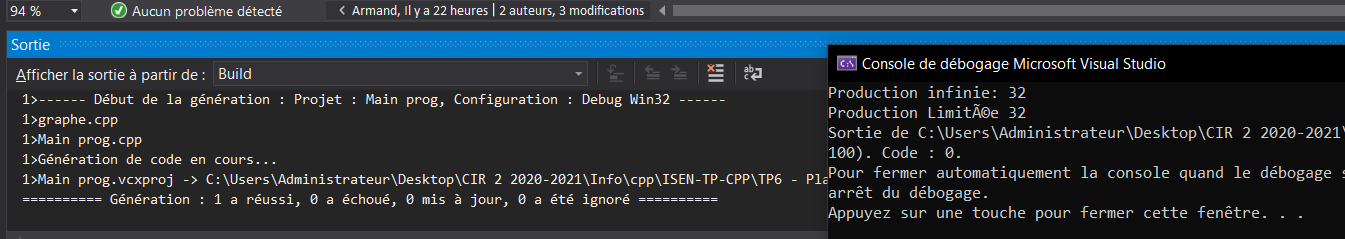
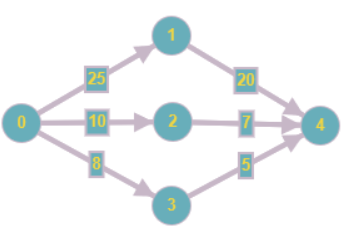
Partie 2.4 :  


Partie 3.1 :

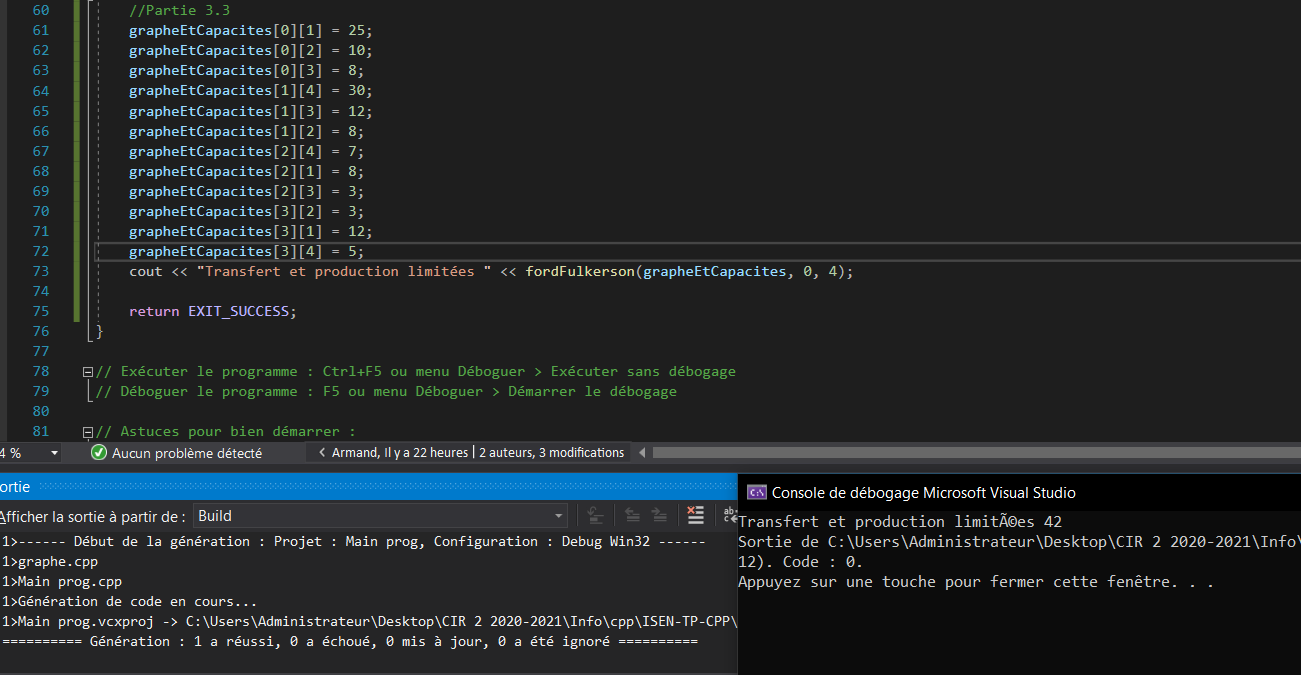
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Départ/Destination | 0 | 1 | 2 | 3 | 4 |
| 0 | 0 | INFINI | INFINI | INFINI | 0 |
| 1 | 0 | 0 | 0 | 0 | 20 |
| 2 | 0 | 0 | 0 | 0 | 7 |
| 3 | 0 | 0 | 0 | 0 | 5 |
| 4 | 0 | 0 | 0 | 0 | 0 |



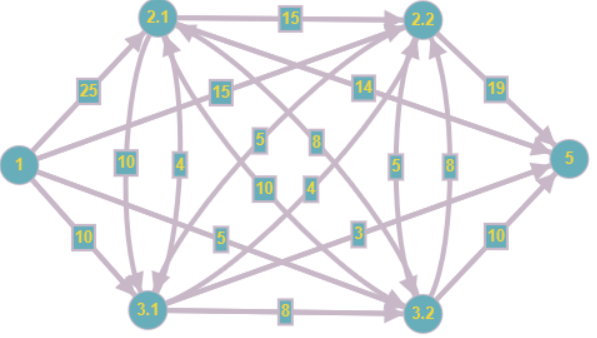
Partie 3.2 :



Partie 3.3 :



Partie 3.4 :



Je représente aussi la matrice pour m’aider dans l’implémentation en C++.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| De / à | 0 (1) | 1 (2.1) | 2 (2.2) | 3 (3.1) | 4 (3.2) | 5 (4.1) | 6 (4.2) | 7 (5) |
| 0 (1) | 0 | 25 | 15 | 10 | 5 | 5 | 8 | 0 |
| 1 (2.1) | 0 | 0 | 15 | 10 | 10 | 10 | 10 | 14 |
| 2 (2.2) | 0 | 0 | 0 | 5 | 5 | 5 | 5 | 19 |
| 3 (3.1) | 0 | 4 | 4 | 0 | 8 | 10 | 10 | 3 |
| 4 (3.2) | 0 | 8 | 8 | 0 | 0 | 5 | 5 | 10 |
| 5 (4.1) | 0 | 4 | 4 | 10 | 10 | 0 | 7 | 7 |
| 6 (4.2) | 0 | 7 | 7 | 5 | 5 | 0 | 0 | 5 |
| 7 (5) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |