## Conditions de travail

Le travail s'organise sur : 2 séances de TP, du travail personnel, et une soutenance.
A réaliser par équipes. Les équipes sont constituées à la base de 3 étudiants. Selon le nombre d'étudiants dans chaque groupe TD/TP, les enseignants vous communiqueront si des équipes de 2 ou 4 sont autorisées, et combien. Tout changement de constitution des équipes après le 1er TP est interdit, sauf accord de l'enseignant. Lui demander avant, et ne pas le mettre devant le fait accompli (qu'il pourra refuser).

Remarque :
Les traces d'exécution fournies dans ce document sont des exemples. Elles ne sont là que pour illustrer le type d'information que l'on vous demande de produire. Vous pouvez présenter ces informations d'une façon différente, à condition bien entendu qu'elles soient compréhensibles.

---

## Résolution de problème d'ordonnancement

L'objectif principal de ce projet est de réaliser un outil de gestion d'ordonnancement. Le but est de construire un graphe à partir de la description d'une suite de tâches et de contraintes, et de calculer les calendriers et marges.

### 1. Lecture d'un tableau de contraintes

Il est question ici de lire en entrée un fichier contenant des contraintes d'ordonnancement telles que vues en cours ou TD.
Des jeux de test vous seront indiqués ultérieurement. Durant votre soutenance, votre enseignant vous indiquera le (ou les) jeu(x) de test à utiliser. Votre programme doit donc demander à l'utilisateur quel jeu de test il souhaite charger en mémoire. Idéalement, votre programme doit permettre de charger et analyser plusieurs fichiers sans être obligé de le relancer.
Un exemple de fichier contenant les contraintes vous est donné en annexe, mais vous avez l'entière liberté pour la structure de ce fichier en entrée. Il doit contenir les informations suivantes :

- la liste des tâches,
- la durée d'exécution de chaque tâche,
- les contraintes d'ordonnancement qui les relient.

Ces contraintes sont du type « la tâche t1 ne peut commencer que lorsque la tâche t2 est terminée ». Dans les exemples fournis ici, nous écrivons cette phrase directement dans les traces d'exécution. Vous pouvez opter pour une présentation plus laconique, que ce soit dans les traces d'exécution ou dans le fichier de description de projet en entrée.

Votre programme sauvegarde en mémoire les données lues sur le fichier en entrée.

A la fin de la phase de lecture, la description du projet est affichée à l'écran.
Ci-dessous un exemple de description, sur la base du tableau « C03 » de vos supports de TD. Ce n'est qu'un exemple : vous pouvez présenter les informations comme vous le souhaitez, pourvu qu'elles soient facilement compréhensibles.

```
-------------------------
Problème d'ordonnancement
-------------------------

Durées des tâches

A  B  C  D  E  F  G  H  I  J  K  L
2  5  4  2  5  5  9  1  5  9  1  1
```

```
Contraintes

C ne peut commencer que lorsque la tâche A est terminée
C ne peut commencer que lorsque la tâche B est terminée
C ne peut commencer que lorsque la tâche D est terminée
D ne peut commencer que lorsque la tâche B est terminée
E ne peut commencer que lorsque la tâche D est terminée
F ne peut commencer que lorsque la tâche E est terminée
G ne peut commencer que lorsque la tâche F est terminée
H ne peut commencer que lorsque la tâche D est terminée
H ne peut commencer que lorsque la tâche F est terminée
H ne peut commencer que lorsque la tâche G est terminée
H ne peut commencer que lorsque la tâche L est terminée
I ne peut commencer que lorsque la tâche E est terminée
I ne peut commencer que lorsque la tâche F est terminée
I ne peut commencer que lorsque la tâche G est terminée
J ne peut commencer que lorsque la tâche K est terminée
K ne peut commencer que lorsque la tâche D est terminée
L ne peut commencer que lorsque la tâche C est terminée
```

## 2. Création du graphe d'ordonnancement

Après la sauvegarde de la description du projet en mémoire, le fichier n'a plus à être utilisé.

Les informations disponibles en mémoire sont alors utilisées afin de créer le graphe d'ordonnancement (la méthode vue en cours et TD).
Votre programme doit indiquer les différentes étapes suivies, au moyen de traces d'exécutions. Par exemple, sur l'exemple de projet C03 :

```
Création du graphe
----------------------
Propriétés du graphe :
- nombre de sommets : 14
- graphe orienté
- 1 valeur (numérique) pour chaque arc
- maximum 1 arc d'un sommet X donné vers un sommet Y donné
--> Ajout des sommets "début de projet" ('a') et "fin de projet" ('z')
--> Création de la matrice d'adjacence : Madj[x][y] = vrai si arc de x vers y
                                                      initialisé à faux
             de la matrice des valeurs : Mval[x][y] = valeur de l'arc s'il existe
--> Création des arcs associés aux contraintes de type "X ne peut commencer que lorsque Y est
terminée"
        A --[2]--> C
        B --[5]--> C
        D --[2]--> C
        B --[5]--> D
        D --[2]--> E
        E --[5]--> F
        F --[5]--> G
        D --[2]--> H
        F --[5]--> H
        G --[9]--> H
        L --[1]--> H
        E --[5]--> I
        F --[5]--> I
        G --[9]--> I
        K --[1]--> J
        D --[2]--> K
        C --[4]--> L
--> Recherche des points d'entrée et points de sortie - Ajout des arcs incidents au début et à la
fin de projet
        a --[0]--> A
        a --[0]--> B
        H --[1]--> z
```

```
I --[5]--> z
J --[9]--> z
```

### 3. Affichage du graphe

Le contenu du graphe est ensuite affiché, sous forme de tableau (matrice). Par exemple :

```
--------------------------
Graphe d'ordonnancement
--------------------------

12 tâches
14 sommets

Matrice d'adjacence et des valeurs
   A   B   C   D   E   F   G   H   I   J   K   L   a   z
A          2
B          5   5
C                                          4
D          2       2           2           2
E                      5           5
F                          5   5   5
G                              9   9
H                                                  1
I                                                  5
J                                                  9
K                                      1
L                              1
a  0   0
z
```

Remarque :
Dans cet exemple d'affichage (que vous pouvez modifiez si vous le souhaitez, à condition que toutes les informations restent clairement disponibles), on se base sur la structure de la matrice d'adjacence. Les valeurs 'vrai' de la matrice d'adjacence sont remplacés par la valeur de l'arc ; l'absence de valeur en ligne x / colonne y indique que l'arc n'existe pas. Cela permet d'utiliser le chiffre 0 uniquement là où il y a un arc de valeur 0.

### 4. Calcul du calendrier au plus tôt et du calendrier au plus tard

Sur la base du graphe construit, vous devez calculer le calendrier au plus tôt et le calendrier au plus tard.

Vous pouvez utiliser la méthode de votre choix.
Lors de son exécution, votre programme doit cependant indiquer les différentes étapes effectuées et les résultats intermédiaires obtenus.

À titre d'exemple :

(i) Méthode abordée en cours et TD :

-- calcul sur la base des rangs : vous devez afficher les étapes du calcul de ces rangs, les valeurs finales des rangs, puis le déroulement du calcul des calendriers en fonction de ces rangs. Nous ne vous donnons pas d'exemple d'exécution en utilisant cette méthode car elle a été suffisamment travaillée en cours et en TD.

Autres méthodes (à vous de faire vos propres recherches si une de ces méthodes vous intéresse) :

(ii) Calcul selon une méthode récursive (pour le calendrier au plus tôt, l'appel du calcul est lancé sur le sommet « fin de projet », et les appels sont propagés jusqu'au sommet « début de projet » pour lequel la valeur est fixée à 0 ; la méthode est symétrique pour le calcul du calendrier au plus tard) : vous devez afficher les différents appels effectués de votre fonction récursive, ainsi que les valeurs retournées ;

(iii) Utilisation de la méthode de Bellman adaptée pour les valeurs maximales des chemins
Avec une éventuelle amélioration : les valeurs sont calculées en fonction des dernières valeurs de 'pi' calculées pour les prédécesseurs, et donc pas nécessairement celles disponibles à l'itération précédente.
- Condition : pas de circuit à valeur strictement positive

Les tracés d'exécution de ces deux dernières méthodes, à titre d'exemple, sont données en Annexe 2.

À l'issue des calculs, les calendriers sont bien entendu affichés sous la forme d'un tableau :

```
-----------
Calendriers
-----------

                  a   A   B   C   D   E   F   G   H   I   J   K   L   z
Date au plus tôt  0   0   0   7   5   7   12  17  26  26  8   7   11  31
Date au plus tard 0   23  0   25  5   7   12  17  30  26  22  21  29  31
```

## 5. Validation du graphe

Revenons à la phase de création du graphe. Il se peut qu'une erreur se soit glissée dans le fichier de description des contraintes. Par exemple : X démarre après Y, qui démarre après Z, qui lui-même démarre après X…
Cela entraîne alors un circuit !

Vous devez donc vérifier, à chaque création d'arc, que cela ne créé pas de circuit dans le graphe.
En cas de circuit, vous déciderez de la suite à donner : abandonner ou ignorer la contrainte.
Dans les deux cas, cela doit bien évidemment être indiqué à l'utilisateur de votre programme.

Par exemple, si l'on ignore simplement les contraintes menant à la création d'un circuit (l'exemple utilisé dans l'illustration ci-dessous n'est pas dans vos supports de TD !) :

```
-------------------------
Problème d'ordonnancement
-------------------------

Durées des tâches

A  B  C  D  E  F  G  H  I
2  4  4  2  3  3  1  1  2

Contraintes

A ne peut commencer que lorsque la tâche D est terminée
B ne peut commencer que lorsque la tâche A est terminée
D ne peut commencer que lorsque la tâche A est terminée
D ne peut commencer que lorsque la tâche B est terminée
D ne peut commencer que lorsque la tâche C est terminée
D ne peut commencer que lorsque la tâche H est terminée
E ne peut commencer que lorsque la tâche D est terminée
F ne peut commencer que lorsque la tâche H est terminée
F ne peut commencer que lorsque la tâche I est terminée
F ne peut commencer que lorsque la tâche E est terminée
F ne peut commencer que lorsque la tâche D est terminée
G ne peut commencer que lorsque la tâche E est terminée
G ne peut commencer que lorsque la tâche D est terminée
G ne peut commencer que lorsque la tâche H est terminée
H ne peut commencer que lorsque la tâche D est terminée
I ne peut commencer que lorsque la tâche D est terminée
I ne peut commencer que lorsque la tâche E est terminée
I ne peut commencer que lorsque la tâche F est terminée


Création du graphe
----------------------
[…]
--> Création des arcs associés aux contraintes de type "X ne peut commencer que lorsque Y est terminée"
        D --[2]--> A
        A --[2]--> B
        A --[2]--> D
```

```
!!! Création de circuit avec valeur potentiellement positive. Contrainte ignorée et arc supprimé.
        B --[4]--> D
!!! Création de circuit avec valeur potentiellement positive. Contrainte ignorée et arc supprimé.
        C --[4]--> D
        H --[1]--> D
        D --[2]--> E
        H --[1]--> F
        I --[2]--> F
        E --[3]--> F
        D --[2]--> F
        E --[3]--> G
        D --[2]--> G
        H --[1]--> G
        D --[2]--> H
!!! Création de circuit avec valeur potentiellement positive. Contrainte ignorée et arc supprimé.
        D --[2]--> I
        E --[3]--> I
        F --[3]--> I
!!! Création de circuit avec valeur potentiellement positive. Contrainte ignorée et arc supprimé.
--> Recherche des points d'entrée et points de sortie - Ajout des arcs incidents au début et à la fin de
projet
        a --[0]--> C
        a --[0]--> H
        B --[4]--> z
        F --[3]--> z
        G --[1]--> z
```

Pour faire cela, vous devez bien entendu mettre en œuvre un algorithme de détection de circuit. Vous avez le choix de la méthode.

## 6. Affichage dans le style d'un diagramme de Gantt

Afficher un graphe sous forme d'une matrice d'adjacence et des valeurs n'est pas ce qui est le plus pratique pour visualiser un projet. Le tableau des calendriers n'est pas très « visuel » non plus.
Un diagramme de Gantt est bien plus intéressant. Son contenu devient immédiatement clair en utilisant l'exemple ci-dessous.
A vous d'en construire un, simple, pouvant être affiché dans une « fenêtre windows/dos ». Bien entendu, les « flèches » (contraintes) visibles dans un vrai diagramme de Gantt ne peuvent pas être affichées dans une fenêtre DOS…

Dans les exemples ci-dessous, on montre 2 diagrammes basés sur l'exemple C03. Dans le premier, les tâches sont placées sur le calendrier en fonction de leurs dates au plus tôt. Dans le second, en fonction de leurs dates au plus tard.

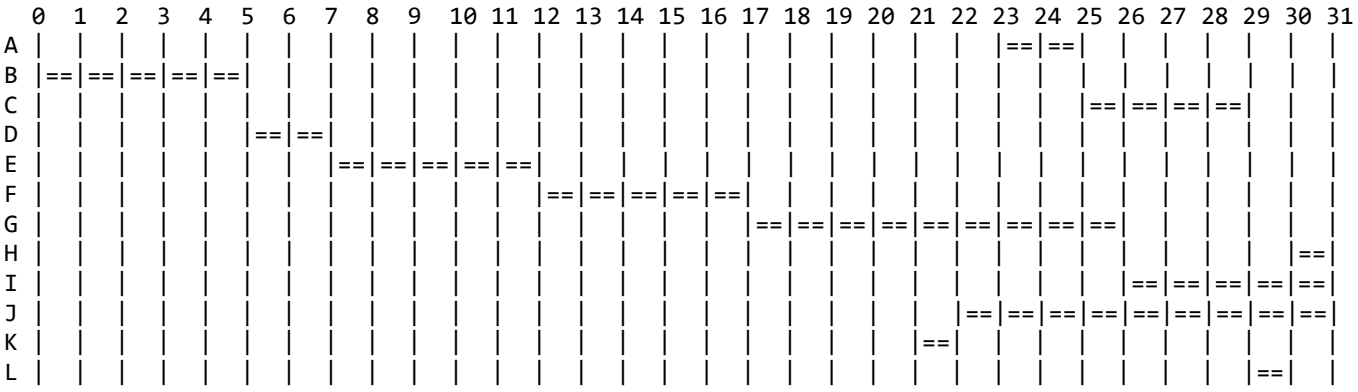
```
--------------------------------
Diagramme "au plus tôt"
--------------------------------

(chaque tâche débute à sa date au plus tôt)

  0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
A |==|==|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
B |==|==|==|==|==|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
C |  |  |  |  |  |  |  |==|==|==|==|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
D |  |  |  |  |  |==|==|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
E |  |  |  |  |  |  |  |==|==|==|==|==|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
F |  |  |  |  |  |  |  |  |  |  |  |  |==|==|==|==|==|  |  |  |  |  |  |  |  |  |  |  |  |  |
G |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |==|==|==|==|==|==|==|==|==|  |  |  |  |
H |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |==|  |  |  |
I |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |==|==|==|==|==|
J |  |  |  |  |  |  |  |  |==|==|==|==|==|==|==|==|==|  |  |  |  |  |  |  |  |  |  |  |  |  |
K |  |  |  |  |  |  |  |==|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
L |  |  |  |  |  |  |  |  |  |  |  |==|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

--------------------------------
Diagramme "au plus tard"
--------------------------------

(chaque tâche débute à sa date au plus tard)
```




## 7. Calcul der rangs (au cas d'utilisation d'une méthode non basée sur les rangs)

Si vous avez choisi d'utiliser une méthode non basée sur les rangs pour le calcul des calendriers, vous devez néanmoins trouver les rangs de chaque sommet et les afficher, avec toutes les étapes.

## Complément d'information -- Aide à la lecture de tâches et contraintes sur fichier

Le code ci-dessous va vous permettre de démarrer votre TP / mini-projet.
Il est basé sur la structure suivante du fichier en entrée représentant le même tableau de contraintes (attention, c'est une troisième variante, différente des structures proposées au début de ce document ! On ne dit pas que cette structure est optimale, elle est proposée à titre d'exemple) :

| | |
|---|---|
| 7 | Ligne 1 : nombre de tâches |
| A 2 | Ligne 2 : libellé (un caractère) de la tâche n° 1 et sa durée |
| B 1 | Ligne 3 : même chose pour tâche n°2 |
| C 4 | … |
| D 3 | … |
| E 1 | … |
| F 8 | … |
| G 3 | Ligne n° (nombre tâches +1) : libellé (un caractère) de la dernière tâche et sa durée |
| A. | Lignes suivantes : libellé d'une tâche suivi par la liste des contraintes terminée par un point |
| B. | Libellé d'une tâche suivi immédiatement du point si pas de contrainte |
| CAD. | Par exemple : A. veut dire que A n'a pas de contrainte |
| DAB. | DAB. veut dire que D a A et B comme contraintes |
| ECD. | |
| FC. | |
| GBE. | |

Le code ci-dessous nécessite l'instruction « #include <fstream> » pour lier la librairie permettant de lire dans un fichier. Il contient une structure de données non précisée (à vous de l'élaborer) et une fonction que vous devez construire vous-mêmes.

```
// Ebauche de code pour démarrer votre TP / mini-projet

// Stockage des durées et contraintes
                // Il s'agit de la structure de données initialisée et remplie lors de
                // l'étape 1 « Lecture d'un tableau de contraintes » décrite précédemment.
typedef struct {
       int nbTaches;        // Les tâches seront numérotées de 1 à nbTaches
       char* nomTaches;     // Chaque tâche est représentée par un caractère
                            // Un numéro est associé à chaque tâche : c'est son indice
                            // dans ce tableau
       int* durees;         // Chaque durée de tâche est stockée dans ce tableau
       // ???? à élaborer   // Stockage des contraintes pour chaque tâche
} tt_contraintes;

// Stockage du graphe construit à partir des contraintes
                // Il s'agit de la structure de données initialisée et remplie lors de
                // l'étape 2 « Lecture d'un tableau de contraintes » décrite précédemment.
typedef struct {
       int nbSommets;       // Nombre de tâches + 2
       bool** adj;          // Matrice d'adjacence
       int** val;           // Matrice des valeurs
                            // Valeur val[x][y] significative uniquement lorsque adj[x][y]=true
} tt_graphe;

// ------------------------------------------
// Lecture des tâches et contraintes sur fichier

// Attention :
// Il n'est fait ici aucune vérification de syntaxe dans le fichier.
// Toute erreur entraînera un dysfonctionnement du programme
printf("Lecture de tableau de contraintes\n");
ifstream F("monFichier.txt");
tt_contraintes* lesContraintes = new tt_contraintes;
char tacheCourante, contrainteCourante;
int indiceTacheCourante, dureeTache;

// Ligne 1 : nombre de tâches
F >> lesContraintes->nbTaches;
```

```
printf("Nombre de tâches : %2d\n", lesContraintes->nbTaches);
lesContraintes->nomTaches = new char[lesContraintes->nbTaches + 1];
              // Indices 1 à nbTaches utilisés. Case 0 non utilisée
lesContraintes->durees = new int[lesContraintes->nbTaches + 1];
              // Indices 1 à nbTaches utilisés. Case 0 non utilisée

// Lignes 2 à nbTaches+1 : durées des tâches
printf("Durées : \n");
for (int t = 1; t <= lesContraintes->nbTaches; t++) {
       F >> tacheCourante;
       lesContraintes->nomTaches[t] = tacheCourante;
       F >> dureeTache;
       lesContraintes->durees[t] = dureeTache;
       printf("%c %2d\n", tacheCourante, lesContraintes->durees[t]);
};

// Ligne suivantes : contraintes
// Une ligne par tâche
printf("Contraintes :\n");
for (int t = 1; t <= lesContraintes->nbTaches; t++) {
       F >> tacheCourante;
       printf("%c\t", tacheCourante);
       F >> contrainteCourante;
       while (contrainteCourante != '.') {
              // Ajout de la contrainte tacheCourante ne peut démarrer que lorsque
              // contrainteCourante est terminée"
              printf("%c", contrainteCourante);
              // ===> ajouterContrainte(lesContraintes, tacheCourante, contrainteCourante);
              //      A vous de coder cette fonction qui ajoute la contrainte à
              //      votre structure de données
              F >> contrainteCourante;
       };
       printf("\n");
};

// ------------------
// Création du graphe

// Initialisation des structures de données
tt_graphe* leGraphe = new tt_graphe;
leGraphe->nbSommets = lesContraintes->nbTaches + 2;
       // début = lesContraintes->nbTaches+1
       // fin = lesContraintes->nbTaches+2
leGraphe->adj = new bool*[leGraphe->nbSommets + 1];  // ligne 0 non utilisée
leGraphe->val = new int*[leGraphe->nbSommets + 1];   // ligne 0 non utilisée
for (int ligne = 1; ligne <= leGraphe->nbSommets; ligne++) {
       leGraphe->adj[ligne] = new bool[leGraphe->nbSommets + 1];  // colonne 0 non utilisée
       leGraphe->val[ligne] = new int[leGraphe->nbSommets + 1];   // colonne 0 non utilisée
       // Initialisation : aucun arc
       for (int colonne = 1; colonne <= leGraphe->nbSommets; colonne++) {
              leGraphe->adj[ligne][colonne] = false;
       };
};

// Transformation contraintes et durées --> graphe
// ===> ajouter ici votre code pour créer réellement le graphe

// Impression du graphe
// Le code proposé ici imprime le contenu de la matrice d'adjacence.
// A vous de l'améliorer pour afficher les valeurs des arcs, ainsi que les noms (caractères)
// de chaque tâche en en-tête de ligne et colonne.
// Et aussi, comme indiqué dans le sujet du TP, remplacer les "1" par la valeur de l'arc,
// et les "0" par rien
// Attention : les sommets 'début' (nbTaches+1) et 'fin' (nbTaches+1) n'ont pas de libelle
// dans la structure "contraintes"
printf("Graphe généré\n");
for (int ligne = 1; ligne <= leGraphe->nbSommets; ligne++) {
       printf("%-2d ", ligne);
       for (int colonne = 1; colonne <= leGraphe->nbSommets; colonne++) {
              if (leGraphe->adj[ligne][colonne]) {
                     printf("1 ");
              }
              else {
                     printf("0 ");
              };
```

```
        };
        printf("\n");
};

printf("Fin de lecture des tâches et contraintes\n"); system("PAUSE");
```

L'exécution de ce code (complété, évidemment, là où il est suggéré que vous le complétiez) sur le fichier en entrée provoque l'affichage suivant dans la console :

```
Lecture de tableau de contraintes
Nombre de tâches :  7
Durées :
A  2
B  1
C  4
D  3
E  1
F  8
G  3
Contraintes :
A
B
C       AD
D       AB
E       CD
F       C
G       BE
Graphe généré
1  0 1 1 0 0 0 0 0 0
2  0 0 0 1 1 0 0 0 0
3  0 0 0 0 1 0 0 1 0
4  0 0 0 0 0 1 1 0 0
5  0 0 0 1 0 1 0 0 0
6  0 0 0 0 0 0 0 1 0
7  0 0 0 0 0 0 0 0 1
8  0 0 0 0 0 0 0 0 1
9  0 0 0 0 0 0 0 0 0
Fin de lecture des tâches et contraintes
Appuyez sur une touche pour continuer...
```

## Annexe 1 -- Rendu du travail

* Rendu en fin de séance de TP

En fin de séance de TP, ou dans un délai qui vous sera communiqué par votre enseignant, vous devez envoyer un bilan de votre avancée dans votre travail.
Ceci se fait par un email simple :

- adresse : velikson.efrei@gmail.com (pour les groupes L'3)
  herve.barbot@efrei.fr (pour les groupes L3)
- titre (sujet) : L'3-X-TG-TPY nom1 / nom2 / nom3 (pour L'3)
  L3-X-TG-TPY nom1 / nom2 / nom3 (pour L3)
  'X' est le numéro de votre groupe TD/TP
  'Y' est le numéro de la séance TP ('1' ou '2'
  Cette structure est IMPERATIVE. Vos enseignants utilisent des filtres automatiques lors de la réception de vos emails. Un mauvais titre de message pourra entraîner sa perte.
- Corps (texte) : bilan d'avancement, sous la forme d'un tableau simple et immédiatement compréhensible :

| | 1 | 2 | 3 … |
|---|---|---|---|
| TP1 | ? | ? | ? … |
| TP2 | ? | ? | ? ... |

  Les ' ?' doivent être remplacés par 'ok', 'ok mais incomplet', 'bug'.
  laisser vide si vous n'avez encore rien fait.
- Pièce jointe : AUCUNE

* Rendu final

Les graphes et tableaux de contraintes à utiliser pour tester votre programme vous seront fournis en séance. Vous devrez les saisir dans un fichier d'entrée pour qu'ils puissent être lus par votre programme.

Vous devez rendre votre travail à la date qui sera fixée par votre enseignant.
Le rendu s'effectue par email :

- adresse : mêmes adresses que pour les rendus en fin de séance de TP
- titre (sujet) : L'3-X-TG-PRJ nom1 / nom2 / nom3 (pour L'3)
  L3-X-TG-PRJ nom1 / nom2 / nom3 (pour L3)
  Cette structure est IMPERATIVE. Vos enseignants utilisent des filtres automatiques lors de la réception de vos emails. Un mauvais titre de message pourra entraîner sa perte.
- corps (texte) du message : reprenez la même forme que pour les emails en fin de séance de TP, et ajoutez une ligne 'PRJ' où vous indiquez le statut final de votre travail
- pièces jointes :
  - votre code source (fichiers de type .c, .cpp ou .h) ;
  - vos fichiers contenant les graphes de test utilisés en entrée de votre programme (fichiers de type .txt) ;
  - les traces d'exécution de votre programme (fichiers de type .txt).

  Remarques :
  - aucun fichier exécutable,
  - aucun fichier « projet » lié à CodeBlocks, Visual Studio ou tout autre outil de développement (.cbp, .sln, …),
  - pas de copie d'écran au format image (.jpeg, …) pour les traces d'exécution.

Tous vos fichiers doivent être préfixés par vos noms (par exemple « dupont-durant-durand-tp.cpp », « dupon-durant-durand-traces.txt », « dupont-durant-durand-g01.txt »). Vos noms doivent toujours être dans le même ordre. Pas de répertoire dans la structure de vos fichiers. Votre enseignant doit être capable de mettre toutes les pièces jointes provenant d'équipes différentes dans un même répertoire, sélectionner facilement les fichiers vous concernant, et compiler puis exécuter correctement votre programme.

Le déroulement des soutenances vous sera précisé ultérieurement.

## Annexe 2 -- Exemples des traces d'exécution correspondant à l'utilisation de deux méthodes parmi les trois méthodes évoquées

```
-------------------------------------------------
Calcul du calendrier au plus tôt - Méthode récursive
-------------------------------------------------

Initialisation: date au plus tôt de début de projet = 0
Calcul récursif en partant du sommet "fin de projet"
z ?
        H ?
                D ?
                        B ?
                                a=0
                        B:=0
                D:=5
                F ?
                        E ?
                                D=5
                        E:=7
                F:=12
                G ?
                        F=12
                G:=17
                L ?
                        C ?
                                A ?
                                        a=0
                                A:=0
                                B=0
                                D=5
                        C:=7
                L:=11
        H:=26
        I ?
                E=7
                F=12
                G=17
        I:=26
        J ?
                K ?
                        D=5
                K:=7
        J:=8
z:=31
```

-- si le calcul du calendrier au plus tard est effectué sur la base du calcul des valeurs maximales des chemins entre un sommet et le sommet « fin de projet » :

```
-----------------------------------------------------------------------------
Calcul du calendrier au plus tard - Méthode des chemins de valeur la plus forte
-----------------------------------------------------------------------------

- Utilisation de la méthode de Bellman adaptée pour les valeurs maximales des chemins
- Amélioration de Bellman : les valeurs sont calculées en fonction des dernières valeurs de 'pi'
calculées pour les prédécesseurs, et donc pas nécessairement celles disponibles à l'itération
précédente
- Condition : pas de circuit à valeur strictement positive.
Attention : au cours et dans les TD, on appliquait l'algo de Bellman pour trouver les chemins les
plus courts. Ici, il s'agit de trouver les chemins les plus longs. La modification nécessaire est
évidente. Une des conséquences, c'est que la condition de ne pas avoir de circuits à valeur
strictement négative est remplacée par la condition ci-dessus.

Etape 1 : calcul des chemins de valeur maximale de chaque sommet à celui de fin de projet
```

```
Initialisation du chemin de "fin de projet" à "fin de projet" = 0

A  B  C  D  E  F  G  H  I  J  K  L  a  z
                     1  5  9  10 2     0
      6  12 10 10 14 1  5  9  10 2     0
8  17 6  12 15 19 14 1  5  9  10 2  17 0
8  17 6  17 24 19 14 1  5  9  10 2  17 0
8  22 6  26 24 19 14 1  5  9  10 2  22 0
8  31 6  26 24 19 14 1  5  9  10 2  31 0
8  31 6  26 24 19 14 1  5  9  10 2  31 0
Etape 2 : date au plus tard(x) = date au plus tard(fin) - valeur maximale du chemin (x--fin)
A  B  C  D  E  F  G  H  I  J  K  L  a  z
23 0  25 5  7  12 17 30 26 22 21 29 0  31
```

-- si le calcul du calendrier au plus tard est effectué par une méthode récursive (on lance le calcul à partir du sommet ‘début’, les appels s’enchaînent jusqu’au sommet ‘fin’ pour lequel la date au plus tard est ici fixée égale à la date au plus tôt) :

```
-----------------------------------------------------
Calcul du calendrier au plus tard - Méthode récursive
-----------------------------------------------------

Initialisation: date au plus tard de fin de projet = date au plus tôt de fin de projet = 31
Calcul récursif en partant du sommet "début de projet"
a ?
        A ?
                C ?
                        L ?
                                H ?
                                        z=31
                                H:=30
                        L:=29
                C:=25
        A:=23
        B ?
                C=25
                D ?
                        C=25
                        E ?
                                F ?
                                        G ?
                                                H=30
                                                I ?
                                                        z=31
                                                I:=26
                                        G:=17
                                        H=30
                                        I=26
                                F:=12
                                I=26
                        E:=7
                        H=30
                        K ?
                                J ?
                                        z=31
                                J:=22
                        K:=21
                D:=5
        B:=0
a:=0
```

FIN DE DOCUMENT