Algorithme de Dinic pour le problème du flot maximum

On se propose de résoudre un problème de flot maximum à l'aide de l'algorithme de **DINIC**. Nous partirons d'un **flot initial nul**.

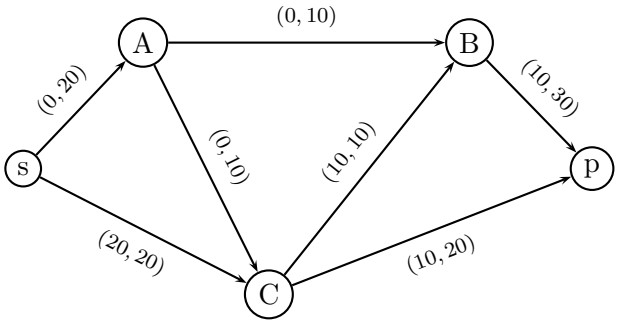
1 Principe de l'algorithme et notion de graphe d'écart

Cet algorithme permet, à partir d'un flot initial, de rechercher une chaîne améliorante dans un réseau de façon à construire un flot de valeur supérieure et ainsi obtenir (après plusieurs itérations) le flot maximum. Cet algorithme se base sur la notion de graphe d'écart (*Residual Graph (RG) en anglais*).

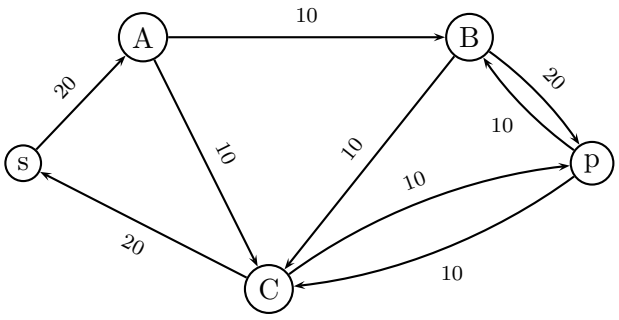
Définition : Soit le réseau $R = (X, A, c)$ et un flot f dans ce réseau.
Au couple (R, f) on associe un graphe d'écart $R^e(f) = (X, A^e_f, c^e)$ où :

- R^e a le même ensemble de sommets que R,
- à tout arc $(xy) \in A$, de capacité $c(xy)$ et de flux $f(xy)$, on associe dans $R^e(f)$
  - un arc (xy) de capacité $c^e(xy) = c(xy) - f(xy)$ si l'arc n'est pas saturé (*ie.* si $f(xy) < c(xy)$),
  - un arc (yx) de capacité $c^e(yx) = f(xy)$ si le flux passant par (xy) n'est pas nul.

Ci-dessous une figure illustrant cette définition. Le premier graphe représente un réseau par lequel circule du sommet source s au sommet puits p un flot d'une valeur 20 et indique pour chaque arc, le flux courant et sa capacité.



Le second graphe représente le graphe d'écart associé au réseau précédent et au flot de valeur 20.



2 Description de l'algorithme de DINIC

```
Procédure DINIC
    Donnée : Réseau R = (X, A, c)
    Donnée/Résultat : flot f

Début
/* Initialisation */
Construire le graphe d'écart  R^e(f)
/* Programme */
TantQue (il existe un chemin de s à p dans R^e(f)) Faire
    Chercher le plus court chemin (nombre d'arcs) de s à p dans R^e(f)
    Déterminez l'amélioration possible de flot sur ce chemin
    Mettre à jour le graphe d'écart R^e(f)
FinTantQue
Mettre à jour le flot f
Fin
```

- Proposer un algorithme pour chercher le plus court chemin en nombre d'arcs de s à p.
- Expliquer le principe de cet algorithme.
- Dérouler l'algorithme de DINIC sur le réseau R_1 (voir section 4.2) à partir d'un flot initial nul et illustrer chaque itération en dessinant le graphe d'écart mis à jour.
  Représenter R_1 après l'exécution de l'algorithme de DINIC en précisant le flot maximal et le flux associé pour chacun des arcs.

3 Analyse des structures de données pour implémenter le réseau et le graphe d'écart associé

Dans ce projet, un réseau est représenté par un graphe orienté auquel on ajoute deux informations pour chaque arc : son flux courant et sa capacité. Dans le cours, plusieurs structures de données ont été vues pour représenter un graphe orienté :

- représentations matricielles :
  - matrice d'incidence
  - matrice d'adjacence
- représentations par tableaux :
  - table des arcs
  - tableaux sommets - successeurs
- représentation par listes de successeurs : un tableau associe à chaque sommet, la liste chaînée de ses successeurs.

Le choix de la structure de données doit être réfléchi en fonction de son utilisation et de son coût en mémoire. Pour ce projet, vous devez utilisez des structures de données similaires pour implémenter le réseau et son graphe d'écart associé. Dans la suite, seules les 3 structures de données (SD) soulignées doivent être analysées.

3.1 Coût mémoire

Soit n le nombre de sommets et m le nombre d'arcs d'un graphe. De plus, les graphes modélisant des réseaux sont généralement "creux" (peu d'arcs entre les sommets).

- Comparer les coûts mémoire des trois SD rappelées ci-dessus.
- Quelles sont les meilleures SD du point de vue du coût mémoire ?

3.2 Coût de traitement

Le problème de recherche du flot maximum dans un réseau requiert un accès facile aux successeurs d'un sommet afin de trouver l'augmentation du flot de s à p à chaque itération de l'algorithme.

- Comparer les coûts de traitement liés à l'accès des successeurs d'un sommet des 3 SD considérées.
- Quelles SD sont les plus adaptées ?

Le graphe d'écart est associé à un réseau et à un flot, des arcs sont donc ajoutés ou supprimés au cours de la résolution par l'algorithme de DINIC. Ceci implique un coût de traitement non négligeable sur la performance de l'implémentation.

- Analyser l'impact des ajouts/suppressions d'arcs pour les 3 SD considérées.
- Adapter la définition du graphe d'écart afin de s'affranchir de cette contrainte d'implémentation.

3.3 Choix de la structure de données

Les analyses précédentes sur les coûts mémoire et de traitement ont mis en évidence les avantages/inconvénients des 3 structures de données considérées. Pour la suite, nous retenons la représentation par listes de successeurs pour représenter le réseau et le graphe d'écart associé.

- Décrire précisément l'implémentation de cette SD pour le réseau de la section 1 et son graphe d'écart associé. Illustrer par deux figures.

4 Description d'un réseau par fichier texte

Nous nous proposons d'utiliser un même format de fichiers pour décrire un réseau (graphe orienté, ayant une source et un puits et dans lequel les arcs ont des capacités maximales). Ce format est inspiré de DIMACS - Center for Discrete Mathematics and Theoretical Computer Science. Ici, nous rappelons que le flot est initialisé avec un flot nul.

4.1 Détails sur le format de fichier pour un problème de recherche de flot maximum

[Extrait de l'explication sur le site de DIMACS...]

The maximum flow problem is structured on a network. Here the arc capacities, or upper bounds, are the only relevant parameters. The problem is to find the maximum flow possible from a given source node to a given sink node. Applications of this problem include finding the maximum flow of orders through a job shop, the maximum flow of water through a storm sewer system, and the maximum flow of product through a product distribution system, among others.

Using the DIMACS format, information is collected into lines, which begin with one-character designators. We describe each type of information line in turn.

— **Comment Lines** : Comment lines give human-readable information about the file and are ignored by programs. Comment lines can appear anywhere in the file. Each comment line begins with a lower-case character `c`.

```
c This is an example of a comment line.
```

— **Problem Line** : There is one problem line per input file. The problem line must appear before any node or arc descriptor lines. For maximum flow network instances the problem line has the following format :

```
p NODES ARCS
```

The lower-case character `p` signifies that this is a problem line. The `NODES` field contains an integer value specifying n, the number of nodes in the network. The `ARCS` field contains an integer value specifying m, the number of arcs in the network.

— **Node Descriptors** : All node descriptor lines must appear before all arc descriptor lines. Node descriptors are of the form :

```
n ID WHICH
```

where `ID` is the node id and `WHICH` is s for the source and t for the sink. Two node descriptors, one for the source and one for the sink, must appear between the problem line and the arc descriptor lines.

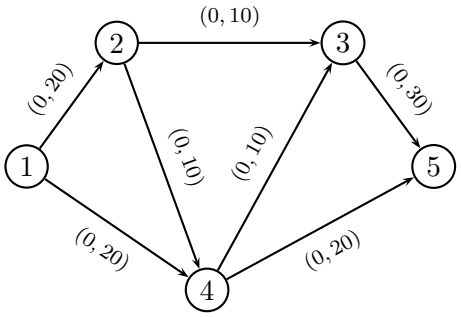
— **Arc Descriptors** : There is one arc descriptor line for each arc in the network. For a maximum flow instance, arc descriptor lines are of the following form :

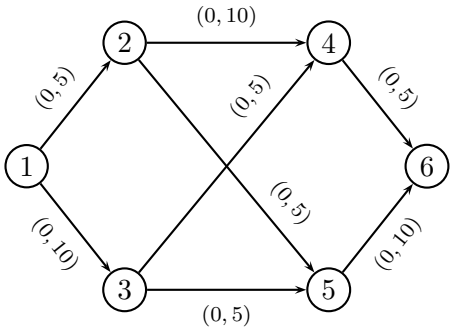
```
a SRC DST CAP
```

The lower-case character `a` signifies that this is an arc descriptor line. For a directed arc (v, w) the `SRC` field gives the identification number for the source vertex v, and the `DST` field gives the destination vertex w. Identification numbers are integers between 1 and n. The `CAP` field gives the arc capacity.

4.2 Exemples

Voici deux réseaux R_1 (à gauche) et R_2 (à droite), tous deux initialisés avec un flot nul :





Les fichiers relatifs à R_1 et R_2 sont :

```
c   Pb de flot max
c   pb lines (nodes, links)
p   5 7
c   source
n   1 s
c   sink
n   5 t
c   arc (from, to, capa)
a   1 2 20
a   1 4 20
a   2 3 10
a   2 4 10
a   3 5 30
a   4 3 10
a   4 5 20
```

```
c   Pb de flot max
c   pb lines (nodes, links)
p   6 8
c   source
n   1 s
c   sink
n   6 t
c   arc (from, to, capa)
a   1 2 5
a   1 3 10
a   2 4 10
a   2 5 5
a   3 4 5
a   3 5 5
a   4 6 5
a   5 6 10
```

Ces fichiers (ainsi que d'autres réseaux plus grands) sont disponibles sur moodle.

5 Décomposition de l'algorithme de DINIC

Afin de concevoir l'algorithme de DINIC sur la structure de données choisie, plusieurs procédures ont été identifiées :

- procédure **buildGraph** qui, à partir d'un graphe décrit par un fichier au format DIMACS, construit votre SD.
- procédure **buildRG** qui, à partir d'un réseau et d'un **flot nul**, construit le graphe d'écart associé (Residual Graph).
- procédure **shortestPath** qui, à partir d'un graphe orienté (ici le graphe d'écart), de 2 sommets s et p, détermine le plus court chemin **en nombre d'arcs**, permettant d'aller de s à p.
- procédure **minCapa** qui, étant donnés un graphe et un chemin de s à p, détermine l'arc de plus petite capacité le long du chemin donné.
- procédure **updateFlowInRG** qui, étant donnés un chemin et un nombre k représentant l'augmentation de flot, met à jour le graphe d'écart.
- procédure **updateFlowInNet** qui, étant donnés un graphe d'écart, met à jour le flot dans le réseau et donc les flux courant de chacun des arcs.

- Définir et décrire la structure de donnée choisie pour représenter un chemin de s à p.
- Ecrire la description (en pseudo-langage) de ces procédures (excepté **buildGraph**) en les instanciant sur la SD choisie (voir section 3.3).
- Ecrire l'algorithme principal (équivalent du *main* en C) qui appelle l'ensemble de ces procédures pour résoudre le problème de flot maximum. Le résultat attendu est un fichier texte contenant le flot optimal et le flux courant de chaque arc du réseau.

6 Cours Moodle

La plateforme Moodle est utilisée pour mettre à disposition des instances du problème et pour rendre les différents fichiers demandés.

6.1 Inscription

Aller sur moodle de l'université puis entrer la clé d'inscription : `mz3i5e`.

6.2 Instances disponibles

Dans les instances de test, vous trouverez les fichiers au format DIMACS (cf. section 4.1) :

1. les exemples R_1 et R_2 présenté dans ce sujet (cf. section 4.2) afin de valider le bon fonctionnement de votre algorithme et donc vous permettre de valider les différentes étapes nécessaires à la résolution du problème de flot maximum cible
2. trois instances avec un nombre de sommets croissant afin d'éprouver l'efficacité de vos algorithmes et de votre implémentation. Pour ces instances, on vous rappelle qu'il est nécessaire d'indiquer dans votre rapport final le résultat obtenu et le temps CPU de l'exécution.

7 Evaluation

Lundi 28 avril 2025 avant 13h50 : déposer sur moodle (en respectant les consignes de nommage du fichier) un rapport d'analyse et de conception, **répondant aux consignes encadrées** du sujet.

Lundi 26 mai 2025 avant 16h : déposer sur moodle (en respectant les consignes de nommage des fichiers) un rapport final (pdf) contenant :

- Contexte du projet et description du sujet
- Partie analyse et conception du précédent rapport avec des compléments éventuels,
- Mode d'emploi (commandes de compilation et de lancement, description des fichiers d'entrées et sorties),
- Description des exemples traités et résultats obtenus,
- Conclusion (point sur ce qui a été fait / non fait), améliorations possibles,
- Bilan personnel sur le projet.

ainsi que les sources (uniquement les .c et .h) dans une archive en **tar exclusivement**.