Ecole Nationale Supérieure de l'Informatique pour l'Industrie et l'Entreprise

ARSE 2023-2024

TD 1 : L'ordonnanceur du noyeau Linux

GOELLER Karim

Pour les deux phases, il faudra prendre soin de se placer dans le répertoire contenant tous les fichiers du projet.

1 Phase 1

S'il n'y a pas l'exécutable nommé `phase1`, il faut exécuter la commande suivante : `make phase1`.
Une fois l'exécutable créé, il faut le lancer avec la commande `./phase1 <nom_de_fichier>`.
Le fichier doit être construit de la manière suivante :

- Ligne 1: un mot (suite de caractères alphanumériques et de "_") désignant la source
- Ligne 2: un mot désignant le puits
- Ligne 3: un entier n désignant le nombre de couloirs
- n Lignes: deux mots désignant respectivement le point de départ et le point d'arrivée pour un couloir à sens unique

```
grigny_c@rimkbinks:~/PRFO/projet/phase_1$ make phase1
ocamlc  -o graph.cmi -c graph.mli
ocamlc  -o graph.cmo -c graph.ml
ocamlc  -o analyse.cmi -c analyse.mli
ocamlc  -o analyse.cmo -c analyse.ml
ocamlc  -o phase1.cmo -c phase1.ml
ocamlc  -o phase1 graph.cmo analyse.cmo phase1.cmo
grigny_c@rimkbinks:~/PRFO/projet/phase_1$ ./phase1 inpute1.txt
```

Figure 1: Création de l'exécutable et utilisation de la phase 1

S'il n'y a pas de chemin de la source au puits, une exception `NoWay` est levée. Sinon, un fichier de sortie est créé où chaque ligne représente un chemin de la source au puits où les sommets sont séparés par des espaces et l'ordre des chemins est arbitraire.

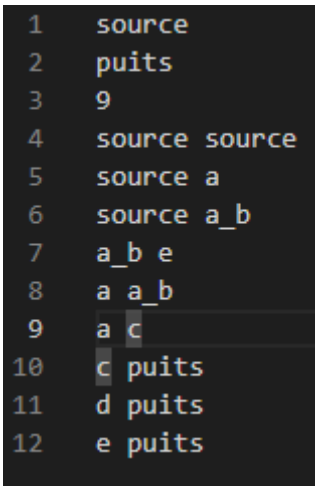

Figure 2: Exemple de fichier d'entrée pour la phase 1

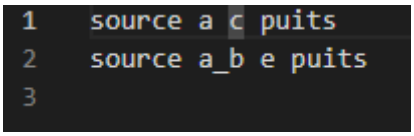

Figure 3: Son fichier de sortie associé

2 Phase 2

S'il n'y a pas l'exécutable nommé `phase2`, il faut exécuter la commande suivante : `make phase2`.
Une fois l'exécutable créé, il faut le lancer avec la commande `./phase2 <nom_de_fichier>`.
Le fichier doit être construit de la manière suivante :

- Ligne 1: un mot (suite de caractères alphanumériques et de "_") désignant la source
- Ligne 2: un mot désignant le puits
- Ligne 3: un entier n désignant le nombre de couloirs
- n Lignes: deux mots et un entier désignant respectivement le point de départ, le point d'arrivée et la capacité (entière) pour un couloir à sens unique.

```
grigny_c@rimkbinks:~/PRFO/projet/phase_1$ make phase2
ocamlc  -o graph.cmi -c graph.mli
ocamlc  -o graph.cmo -c graph.ml
ocamlc  -o analyse.cmi -c analyse.mli
ocamlc  -o analyse.cmo -c analyse.ml
ocamlc  -o phase2.cmo -c phase2.ml
ocamlc  -o phase2 graph.cmo analyse.cmo phase2.cmo
grigny_c@rimkbinks:~/PRFO/projet/phase_1$ ./phase2 input.txt
```

Figure 4: Création de l'exécutable et utilisation de la phase 2

S'il n'y a pas de chemin de la source au puits, une exception `NoWay` est levée. Sinon, un fichier de sortie est créé où chaque ligne représente :

- Ligne 1: le débit maximal entre la source et le puits
- Ligne 2: un entier k désignant le nombre de couloirs utilisés dans le flot maximum trouvé
- k lignes: deux mots et un entier désignant respectivement un point de départ et un point d'arrivée et la capacité utilisée sur ce chemin élémentaire

```
source
puits
9
source source 10
source a 10
source a_b 5
a_b e 7
a a_b 4
a c 3
c puits 7
d puits 12
e puits 28
```

Figure 5: Exemple de fichier d'entrée pour la phase 2

```
10
7
source a_b 5
source a 5
e puits 7
c puits 3
a_b e 7
a c 3
a a_b 2

```

Figure 6: Son fichier de sortie associé