Implémentation de l'algorithme de Dinic et d'Edmonds-Karp

TP Algorithmique, Complexité & Calculabilité (FMIN105)

William Dyce Thibaut Marmin
Clément Sipieter

Université Montpellier 2

15 Décembre 2011

TP Algorithmique, Complexité & Calculabilité

Présentation du sujet

Implémentation

Tests & résultats

Conclusion

Démonstration

TP Algorithmique, Complexité & Calculabilité

Présentation du sujet
Algorithmes

Implémentation

Tests & résultats

Conclusion

Démonstration

Algorithmes

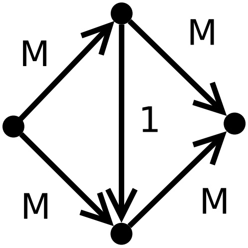
Ford-Fulkerson : $O(mnC^*)$

Idée générale

- Trouver un une chaîne ameillorante.
- Augmenter le flot le long de cette chaîne.

Défauts

- "Pseudo-exponentielle"



- "Diamond maudit" : $2 \times M$ itérations!

Algorithmes

Edmonds-Karp : $O(n^5)$

Idée générale

- Prendre le chemin le plus court en nombre d'arcs

Défauts

- Graphes avec de multiple chemins de plus en plus longs ...

Algorithmes

Dinic : $O(n^4)$

Idée générale

- Rechercher un ensemble de chemins
- Trouver un flot bloquant dans le "graphe de couches"

TP Algorithmique, Complexité & Calculabilité

Présentation du sujet

Implémentation
- Choix Techniques
- Structures
- Mise en oeuvre des algorithmes

Tests & résultats

Conclusion

Démonstration

Choix du langage de programmation

C++

- rapidité d'éxecution
- langage à objets
- connaissance du langage
- langage très répandu

Utilisation d'un gestionnaire de version

git - the stupid content tracker

- sauvegarde
- partage
- mise en commun

Représentation du problème de flot maximum

Réseau de transport

- Graphe orienté pondéré
- une source
- un puits

Graphe d'écarts

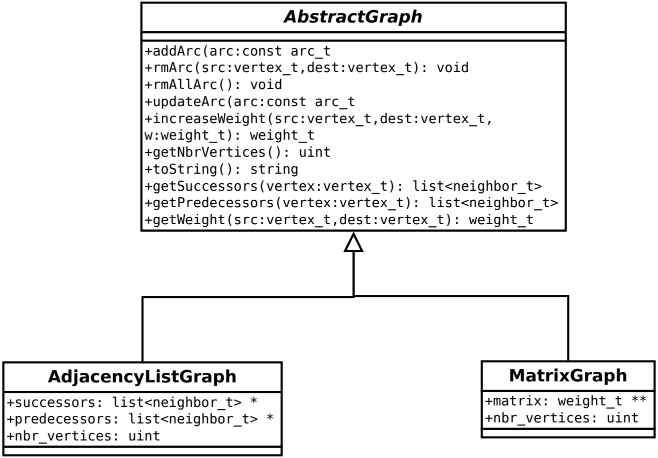
- Graphe orienté pondéré

Représentation du problème de flot maximum

Graphe de couches

- Graphe orienté pondéré
- Représenation des couches par un tableau de listes de sommets.

Diagramme de classes



Edmonds-Karp

Recherche du plus court chemin en nombre d'arcs

Parcours en largeur

- Graphes en listes d'adjacences : O(n+m)
- Graphes en matrice d'adjacences : $O(n^2)$

Mise à jour du graphe d'écart

Parcours du chemin
Décrementation du poids de chaque arc u,v du chemin
Incrémentation du poids de chaque arc v,u

- Graphes en listes d'adjacences : O(n+m)
- Graphes en matrice d'adjacences : O(n)

Dinic

Génération du graphe de couches

Parcours en largeur + stockage d'une liste de parents par sommets

Calcul du flot bloquant

Parcours en largeur + stockage d'une liste de parents par sommets

- Graphes en listes d'adjacences : O(nm)
- Graphes en matrice d'adjacences : O(n^2m)

Mise à jour du graphe d'écart

Pour chaque arcs du flot bloquant
Décrementation du poids de chaque arc u,v du chemin
Incrémentation du poids de chaque arc v,u

- Graphes en listes d'adjacences : O(m^2)
- Graphes en matrice d'adjacences : O(n^2)

Génération de réseaux de transport aléatoires

Deux stratégies

- Tirage aléatoire de deux sommets
- Génération de tous les arcs possibles et tirage d'un arc

TP Algorithmique, Complexité & Calculabilité

Présentation du sujet

Implémentation

Tests & résultats

- Méthode de tests
- Résultats

Conclusion

Démonstration

Méthode de tests

Série de tests

Complexité

- Edmonds-Karp : $O(nm^2)$
- Dinic : $O(n^2 m)$

Tests effectués

- Nombre de sommets : 100, 200, 300, ...1000
- Densité du graphe : 20%, 50% et 80%

Méthode de tests

Profiling

GNU gprof

Profiler, analyse du code en fonction du temps passé par chaque fonction à l'exécution.

- Compilation avec l'argument `-pg`
- Exécution du programme, génération du fichier `gmon.out`
- Exportation des statistiques en fichier texte

Tests & résultats

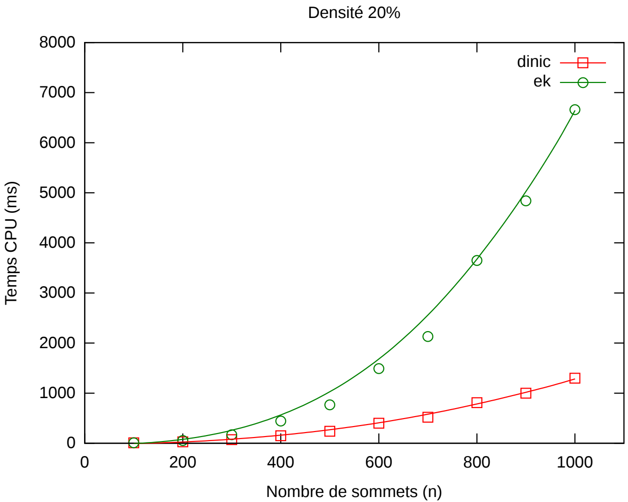
Profiling

GNU gprof

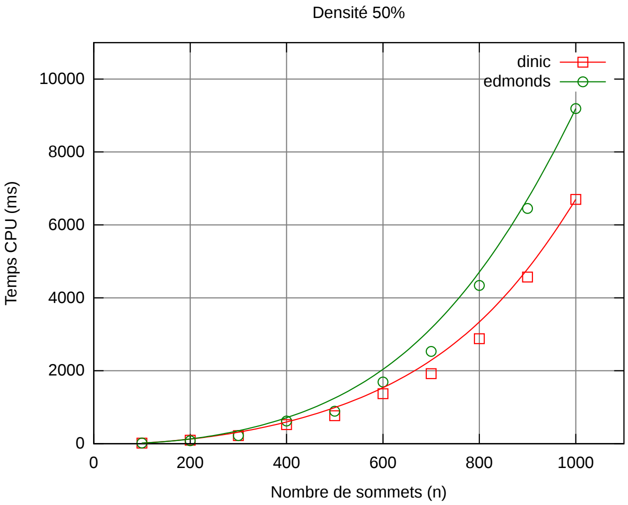
Statistique fournies pour chaque fonction :

- % temps cpu total
- temps cpu
- temps cpu par appel (de manière cumulative ou non)

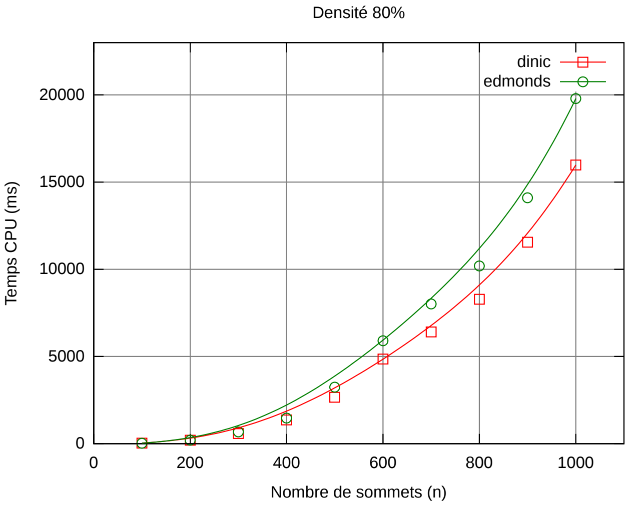
Résultats


Densité 20%
8000
7000
6000
5000
4000
3000
2000
1000
0
Temps CPU (ms)
0
200
400
600
800
1000
Nombre de sommets (n)
dinic
ek

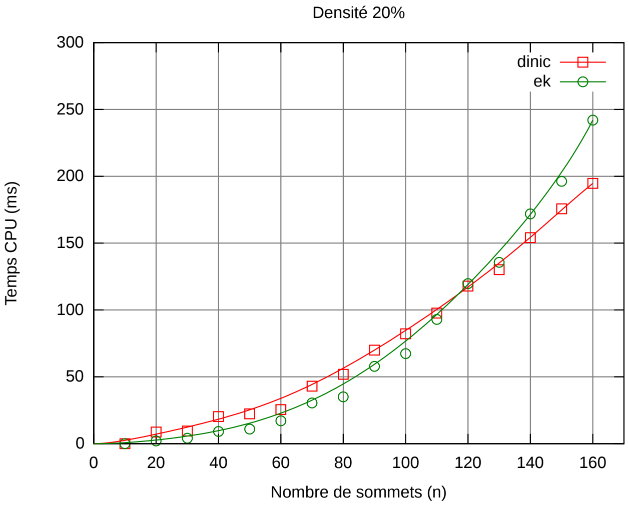
Résultats


Densité 50%
dinic
edmonds
Temps CPU (ms)
Nombre de sommets (n)
0
2000
4000
6000
8000
10000
0
200
400
600
800
1000

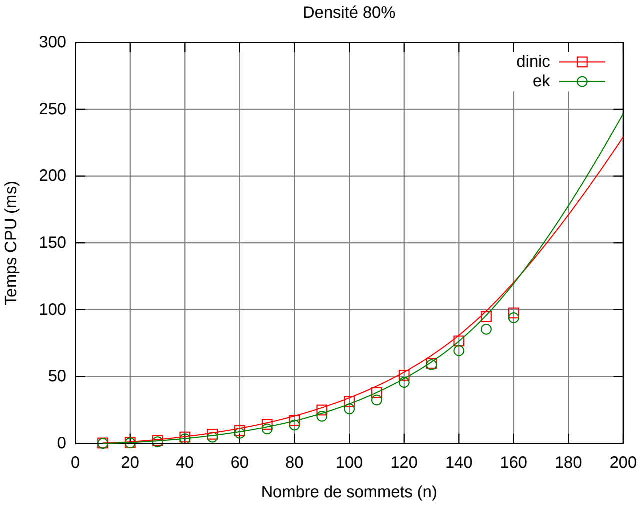
Résultats


Densité 80%
dinic
edmonds
Temps CPU (ms)
0
5000
10000
15000
20000
0
200
400
600
800
1000
Nombre de sommets (n)

Résultats


Densité 20%
Temps CPU (ms)
Nombre de sommets (n)
dinic
ek
0
50
100
150
200
250
300
0
20
40
60
80
100
120
140
160

Résultats


Densité 80%
Temps CPU (ms)
Nombre de sommets (n)
dinic
ek
0
50
100
150
200
250
300
0
20
40
60
80
100
120
140
160
180
200

Résultats

Conclusion

- Dinic globalement plus rapide
- Surtout sur des graphes peu denses
- Edmonds-Karp efficace sur des petits graphes

$\Rightarrow$ Cohérence avec les complexités théoriques

$$\swarrow \qquad \searrow$$

Dinic	Edmonds-Karp
$O(nm^2)$	$O(n^2m)$

TP Algorithmique, Complexité & Calculabilité

Présentation du sujet

Implémentation

Tests & résultats

Conclusion

Conclusion

Démonstration

Conclusion

TP Algorithmique, Complexité & Calculabilité

Présentation du sujet

Implémentation

Tests & résultats

Conclusion

Démonstration

Démonstration

Démonstration