Julie GASPAR
Max HOFFER
Laureline PARTONNAUD
Simon PEREIRA

Projet pour le 04/01/2021

Outils pour la conception d'algorithmes

Projet - Les graphes du Web

Type de graphe	Nombre sommets	Nombre d'arêtes	Degré maximal	Degré moyen
Edgar Gilbert	10	23	7	4.6
Edgar Gilbert	100	2401	62	48.02
Edgar Gilbert	1000	249714	573	499.428
Edgar Gilbert	2000	1000191	1072	1000.191
Edgar Gilbert	5000	Temps de calcul trop long		
Barabasi-Albert	10	15	6	3.0
Barabasi-Albert	100	124	19	2.48
Barabasi-Albert	1000	1373	44	2.746
Barabasi-Albert	10000	13762	82	2.752
Barabasi-Albert	15000	20533	101	2.738
FacebookSites	22470	170823	709	15.22
GitHub	37700	289003	9458	15.332
RoadNetwork	1965206	Temps de calcul trop long	12	2.816
twitchDE	9498	153138	4259	32.246
Wikipedia1	2277	31371	732	27.599
Wikipedia2	11631	170773	3546	29.39

Au-delà de 5000 sommets pour les graphes Edgar Gilbert et de 15000 sommets pour les graphes Barabasi-Albert, les temps de calcul (notamment le nombre d'arêtes) sont trop long (plus de 10 min).
En se basant sur cette limite, nous avons trouvé que la taille maximum des graphes de Edgar Gilbert est de 10000 sommets et de 20000 sommets pour les graphes de Barabasi-Albert. Avec ces tailles, il est impossible de calculer les paramètres.

Pour la lecture des grands graphes de Stanford, nous avons modifié les fichiers en retirant les lignes comprenant du texte pour ne garder que les voisins des différents sommets.

L'algorithme de diamètre et sa complexité :

Le calcul du diamètre se fait en plusieurs étapes.

Premièrement, on transforme notre graphe représenté par "sommet, liste de voisin" en une matrice des poids, nos graphes n'étant pas pondérés, il s'agit d'une matrice d'adjacence avec les zéros remplacés par "infini" (poids d'une arête qui n'existe pas), on utilise une fonction math.infini pour pouvoir comparer par la suite.

Ensuite, on applique l'algorithme de Floyd-Warshall à cette matrice, on obtient alors la matrice des distances : ici le nombre d'arêtes du plus court chemin entre les sommets.

Enfin, on parcourt cette matrice pour y trouver la distance maximale soit le diamètre du graphe.

Complexité :

```
def graphetoMat(graphe):
      M = init matrice de taille nbSommets²

      for s in graphe.sommets:
            for v in graphe.getSommet(s).voisins:
                  M[s-1][v.id-1]=1                    O(1) opérations
      return(M)
```

Complexité des boucles imbriquées :

Pour chaque sommets on visite tous les voisins ⇔ 2 * nbArêtes

→ O(nbArêtes)

```
def floyd_warshall(graphe):
      distance = init matrice avec graphtoMat()
      n = nbSommets
      for k in range(n):
            for i in range(n):
                  for j in range(n):
                        distance[i][j] = min(distance[i][j],
                  distance[i][k] + distance[k][j])   O(1) opérations
      return(distance)
```

Complexité des boucles imbriquées :

Chaque boucle itère les instructions nbSommets fois → O(nbSommets^3)

Complexité de recherche du maximum :

Le parcours de la matrice se fait dans son entièreté → O(nbSommets²)