

STRUCTURES DE DONNÉES AVANCÉES ET THÉORIE

DES GRAPHS

RAPPORT SUR LE GOOGLE'S PAGERANK



PageRank



Encadré par : MME OUAFAE BAIDA

Réalisé par : Nada Lazreq

Anas Azizi

Saad Laazrak

Table des matières

1.INTRODUCTION :	3
1.1. DÉFINITION DU PAGERANK :	3
1.2. HISTORIQUE DU PAGERANK DE GOOGLE :	3
1.3. FONCTIONNEMENT :	3
3.DÉFINITION MATHÉMATIQUE DES CALCULS :	5
3.3. LE MODÈLE UTILISÉ PAR GOOGLE :	6
4.PROGRAMME C ASSOCIÉ À L'ALGORITHME :	7
5.1TEST DE L'ALGORITHME SUR LA BASE DE DONNÉE ET EXPLICATION DES RÉSULTATS :	12
5.2.APPLICATION DE L'ALGORITHME SUR UNE BASE DE DONNEE	14
Représentation sur machine du graphe :	14
6.CONCLUSION :	17

1.INTRODUCTION :

1.1. DÉFINITION DU PAGERANK :

L'algorithme PageRank permet de classer les pages web les unes par rapport aux autres en fonction de leur importance relative dans le graphe du web. Pour cela, il affecte un rang à chaque page, de sorte que le rang d'une page est d'autant plus élevé que cette page est importante. Ce rang est calculé régulièrement, au fur et à mesure que des pages apparaissent et disparaissent sur le web. Lorsqu'un utilisateur saisit un ou plusieurs mots-clés dans un moteur de recherche, d'autres algorithmes sont utilisés (en temps réel) pour sélectionner les pages qui sont à la fois pertinentes et importantes.

1.2. HISTORIQUE DU PAGERANK DE GOOGLE :

PageRank a été introduite dans le papier coécrit par Sergey Brin et Lawrence Page en 1998 et dans le brevet déposé par Lawrence Page en 1998. Ce travail s'inscrit dans la continuité d'autres travaux, comme l'algorithme HITS proposé par Jon Kleinberg en 1999 et puis en :

11 décembre 2000 : Google lance la barre d'outils Google

17 juin 2004 : Google dépose le brevet du surfeur raisonnable

12 octobre 2006 : Google dépose son brevet « seed sets »

8 mars 2016 : Google annonce le retrait de la barre d'outils Google

Trouvez plus de détails ci-dessous.

1.3. FONCTIONNEMENT :

Cet algorithme d'analyse repose sur deux principes fondamentaux pour vérifier la qualité d'une page web :

1. La quantité de liens pointant vers une page web.
2. La qualité des pages qui réalisent des liens vers cette page.

Les critères complémentaires sont pris en compte pour définir ce score :

- Liens externes entrants.
- Liens internes - Liens sortants.

2.EXEMPLES D'APPLICATION DE L'ALGORITHME :

L'algorithme PageRank est couramment utilisé dans divers domaines pour classer et ordonner des éléments en fonction de leur importance dans un réseau.

Moteurs de recherche (Recherche Web) :

Google utilise l'algorithme PageRank pour évaluer et classer les pages web en fonction de leur pertinence, influant sur l'ordre des résultats de recherche.

Réseaux sociaux :

Dans les réseaux sociaux comme Facebook et Twitter, PageRank peut être utilisé pour évaluer l'influence des utilisateurs et classer les contenus en fonction de leur popularité.

Analyse de réseaux :

PageRank est utilisé pour identifier les nœuds importants dans divers réseaux, tels que les réseaux sociaux, les réseaux biologiques, et les réseaux informatiques.

Recommandation de produits :

Dans le commerce électronique, PageRank peut être appliqué pour recommander des produits en se basant sur la popularité et la relation entre les produits, aidant les utilisateurs à découvrir des articles pertinents.

Systèmes de recommandation de contenu :

Dans les plateformes de streaming vidéo comme Netflix, PageRank peut être utilisé pour recommander des films et des séries en se basant sur la similarité des préférences entre les utilisateurs.

Évaluation des compétences professionnelles :

Dans les réseaux professionnels en ligne tels que LinkedIn, PageRank peut être utilisé pour évaluer l'influence et l'expertise des professionnels, aidant ainsi les recruteurs à identifier les candidats les plus pertinents.

3.DÉFINITION MATHÉMATIQUE DES CALCULS :

3.1. Définition mathématique du graphe :

Le graphe du web Il s'agit d'un **graphe orienté** dont **les sommets sont les indices des pages web** et les arcs représentent les **hyperliens entre les pages**. Autrement dit, les sommets du graphe sont des entiers de 1 à N, ou N est le nombre total de pages web ; pour chaque $i, j = 1, \dots, N$, le graphe contient un arc de i vers

j si la page i contient au moins un hyperlien vers la page j, auquel cas on écrira

$i \rightarrow j$. Si une page i ne contient aucun hyperlien sortant, on ajoute une boucle, c'est-à-dire un arc de i vers i. Cet ajout simplifiera certains raisonnements dans la suite du cours ; à noter cependant qu'il existe d'autres moyens de gérer le cas des pages n'ayant pas d'hyperlien sortant.

La **matrice d'adjacence** $A = (a(i, j))_{i, j=1, \dots, N}$ du graphe du web est définie par

$$a(i, j) = \mathbb{1}_{i \rightarrow j} = \begin{cases} 1 & \text{si } i \rightarrow j, \\ 0 & \text{sinon,} \end{cases} \quad \forall i, j = 1, \dots, N.$$

3.2. Calcul itératif du PageRank :

On définit le vecteur PR de taille N, où PR[i] est le PageRank de la page

Initialement, on peut attribuer à chaque page un PageRank égal $1/N$.

$$PR[i+1] = M * PR[i]$$

PR est un vecteur propre de la matrice M

-on part d'un vecteur **PR [0]** aléatoire

-à chaque itération, on multiplie **PR[i]** par la matrice : **PR[i+1] = M*PR[i]**

3.3. LE MODÈLE UTILISÉ PAR GOOGLE :

MATRICE DE GOOGLE :

Supposons qu'on peut plus aller à aucune page donc il y a aucun lien associé à cette page et donc notre vecteur PR va converger vers 0 il n'y aura plus de rang possible entre les différents liens de la même manière imaginons qu'on ajoute 2 pages étant donné que les deux graphes ne sont pas reliés entre eux graphe notre algorithme va également converger vers 0 et on ne pourra pas faire de rang entre les différentes pages donc pour résoudre ce problème Larry page et son équipe ont l'idée d'introduire un facteur d appelé le damping factor ou le facteur d'amortissement , il va permettre de lier toutes les pages entre elles et donc de permettre la convergence de l'algorithme.

Pour échapper aux trous noirs, Google utilise un modèle plus raffiné : avec une probabilité fixée d le surfeur abandonne sa page actuelle PJ et recommence sur une des n pages du web, choisie de manière équiprobable ; sinon, avec probabilité 1 – d, le surfeur suit un des liens de la page PJ, choisi de manière équiprobable. Cette astuce de « téléportation » évite de se faire piéger par une page sans issue, et garantit d'arriver n'importe où dans le graphe, indépendamment des questions de connexité.

DAMPING FACTOR (d) : Dans l'algorithme PageRank, généralement fixé à 0.85, représente la probabilité qu'un utilisateur suive un lien plutôt que de choisir une page au hasard lors de la navigation sur le web. Il modifie la formule de calcul du PageRank en
Introduisant cette probabilité d'amortissement

$$\hat{M} = d * M + \frac{(1 - d)}{N} * E$$

L'équation du vecteur propre devient la suivante :

$$PR[i+1] = \hat{M} * PR[i]$$

4.PROGRAMME C ASSOCIÉ À L'ALGORITHME :

```
1 void initializeMatrix(int Matrice_ad[num_noeud][num_noeud], int edges[][2], int num_edges) {
2     for (int i = 0; i < num_edges; i++) {
3         int from = edges[i][0];
4         int to = edges[i][1];
5         Matrice_ad[from - 1][to - 1] = 1;
6     }
7 }
```

Cette fonction initialise la matrice d'adjacence par rapport aux nombres de liens liés à chaque page, on pose comme condition que la page ne peut pas avoir un lien vers elle-même.

```
1 void printAdjacencyMatrix(int Matrice_ad[num_noeud][num_noeud]) {
2     printf("Matrice d'adjacence:\n");
3     for (int i = 0; i < num_noeud; i++) {
4         for (int j = 0; j < num_noeud; j++) {
5             printf("%d ", Matrice_ad[i][j]);
6         }
7         printf("\n");
8     }
9 }
```

Affichage de la matrice d'adjacence

```

1
2 void initializePageRank(double pageRank[NUM_NODES]) {
3     for (int i = 0; i < NUM_NODES; i++) {
4         pageRank[i] = 1.0 / NUM_NODES;
5     }
6 }

```

Cette fonction initialise les valeurs de PageRank pour chaque nœud du graphe avec une probabilité uniforme initiale, assurant ainsi une distribution égale du PageRank initial entre tous les nœuds.

```

1 void calculatePageRank(int Matrice_ad[num_noeud][num_noeud], double pageRank[NUM_NODES], double newPageRank[NUM_NODES]) {
2     for (int i = 0; i < NUM_NODES; i++) {
3         for (int j = 0; j < NUM_NODES; j++) {
4             if (Matrice_ad[j][i] == 1) {
5                 int outgoingLinks = 0;
6                 for (int k = 0; k < NUM_NODES; k++) {
7                     outgoingLinks += Matrice_ad[j][k];
8                 }
9                 newPageRank[i] += DAMPING_FACTOR * pageRank[j] / outgoingLinks;
10            }
11        }
12        newPageRank[i] += (1 - DAMPING_FACTOR) / NUM_NODES;
13    }
14 }

```

Cette fonction implémente une itération de l'algorithme du PageRank pour mettre à jour les valeurs de PageRank des nœuds du graphe en fonction de la structure de la matrice d'adjacence et des valeurs de PageRank actuelles.



```
1 newPageRank[i] += DAMPING_FACTOR * pageRank[j] / outgoingLinks;
```

Calcul de la nouvelle valeur de PageRank pour le nœud i en fonction du PageRank du nœud j, du facteur d'amortissement (DAMPING_FACTOR), et du nombre de liens sortants du nœud j.




```
1 newPageRank[i] += (1 - DAMPING_FACTOR) / NUM_NODES;
```

Ajout d'une composante supplémentaire pour prendre en compte le facteur de non-amortissement (1 - DAMPING_FACTOR) et garantir une certaine redistribution du PageRank à tous les nœuds, même ceux sans liens entrants



```
1 void updatePageRank(double pageRank[NUM_NODES], double newPageRank[NUM_NODES]) {  
2     for (int i = 0; i < NUM_NODES; i++) {  
3         pageRank[i] = newPageRank[i];  
4     }  
5 }  
6
```




```

1
2 void normalizePageRank(double pageRank[NUM_NODES]) {
3     double sum = 0;
4     for (int i = 0; i < NUM_NODES; i++) {
5         sum += pageRank[i];
6     }
7
8     for (int i = 0; i < NUM_NODES; i++) {
9         pageRank[i] /= sum;
10    }
11 }

```

La normalisation est une étape souvent effectuée dans l'algorithme du PageRank pour garantir que la somme des valeurs de PageRank reste égale à 1, ce qui représente une distribution de probabilité valide.




```

1 void printFinalPageRank(double pageRank[NUM_NODES]) {
2     printf("Final PageRank scores:\n");
3     for (int i = 0; i < NUM_NODES; i++) {
4         printf("Node %d: %f\n", i + 1, pageRank[i]);
5     }
6 }

```

Affichage du PageRank de chaque page




```

1 void rank(double pageRank[NUM_NODES], int classement[NUM_NODES]) {
2     for (int i = 0; i < NUM_NODES; i++) {
3         classement[i] = i;
4     }
5     for (int i = 0; i < NUM_NODES - 1; i++) {
6         for (int j = i + 1; j < NUM_NODES; j++) {
7             if (pageRank[classement[i]] < pageRank[classement[j]]) {
8                 int temp = classement[i];
9                 classement[i] = classement[j];
10                classement[j] = temp;
11            }
12        }
13    }
14 }

```

Le rôle de cette fonction est de trier les valeurs du PageRank de chaque page dans un ordre décroissant



```

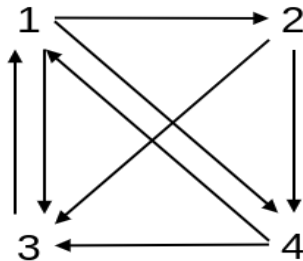
1
2 void printRankedPageRank(double pageRank[NUM_NODES], int classement[NUM_NODES]) {
3     printf("Classement des pages par PageRank:\n");
4     for (int i = 0; i < NUM_NODES; i++) {
5         int n = classement[i];
6         printf("Page %d: %f\n", n + 1, pageRank[n]);
7     }
8 }
9

```

5.1.TEST DE L'ALGORITHME SUR LA BASE DE DONNÉE ET

EXPLICATION DES RÉSULTATS :

Supposons que nous ayons le site Web suivant composé de quatre pages reliées entre elles :




Chaque page Web doit relier au moins une autre page et aucune page Web ne peut se lier elle-même.

Le score d'une page donnée est dérivé des liens créés vers cette page Web à partir d'autres pages. Le web devient ainsi une démocratie où les pages votent pour l'importance des autres pages en créant des liens vers elles.

Exécution du code :

```
Matrice d'adjacence:
0 1 1 1
0 0 1 1
1 0 0 0
1 0 1 0
Classement des pages par PageRank:
Page 1: 0.368151
Page 3: 0.287962
Page 4: 0.202078
Page 2: 0.141809
PS C:\Users\NADA\OneDrive\Bureau\c project>
```



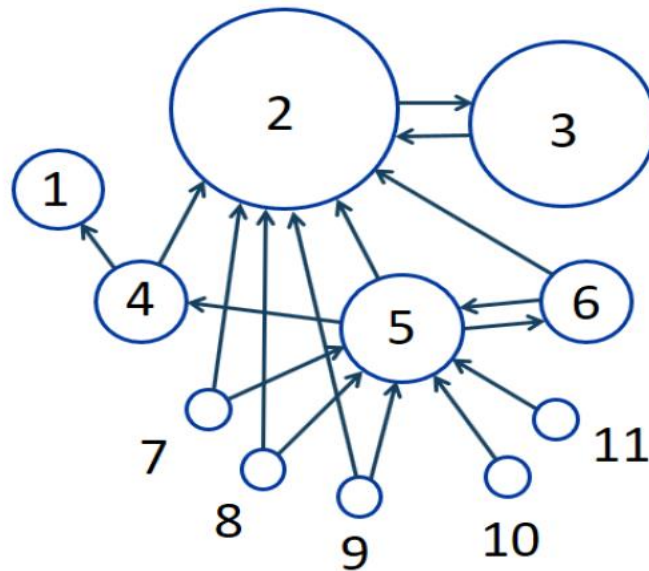
```

1  int main() {
2      int Matrice_ad[num_noeud][num_noeud] = {0};
3      int edges[][2] = {
4          {1, 2},{1, 4},{1, 3},
5          {2, 3},{2, 4},
6          {3, 1},
7          {4, 3},{4, 1},
8      };
9
10     initializeMatrix(Matrice_ad, edges, 8);
11     printAdjacencyMatrix(Matrice_ad);
12
13     double pageRank[NUM_NODES];
14     initializePageRank(pageRank);
15
16     for (int iter = 0; iter < MAX_ITERATIONS; iter++) {
17         double newPageRank[NUM_NODES] = {0};
18         calculatePageRank(Matrice_ad, pageRank, newPageRank);
19         updatePageRank(pageRank, newPageRank);
20     }
21
22     normalizePageRank(pageRank);
23     int classement[NUM_NODES];
24     rank(pageRank, classement);
25     printRankedPageRank(pageRank, classement);
26
27     return 0;
28 }

```

5.2.APPLICATION DE L'ALGORITHME SUR UNE BASE DE DONNEE

Supposons que nous ayons le site Web suivant composé de 11 pages reliées entre elles :



Représentation sur machine du graphe :

Matrice d'adjacence :

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



```

1  int main() {
2      int Matrice_ad[num_noeud][num_noeud] = {0};
3      int edges[][2] = {
4          {2, 3},
5          {3, 2},
6          {4, 1}, {4, 2},
7          {5, 4}, {5, 6},
8          {6, 5}, {6, 2},
9          {7, 2}, {7, 5},
10         {8, 5}, {8, 2},
11         {9, 5}, {9, 2},
12         {10, 5},
13         {11, 5},
14     };
15
16     initializeMatrix(Matrice_ad, edges, 15);
17     printAdjacencyMatrix(Matrice_ad);
18
19     double pageRank[NUM_NODES];
20     initializePageRank(pageRank);
21
22     for (int iter = 0; iter < MAX_ITERATIONS; iter++) {
23         double newPageRank[NUM_NODES] = {0};
24         calculatePageRank(Matrice_ad, pageRank, newPageRank);
25         updatePageRank(pageRank, newPageRank);
26     }
27
28     normalizePageRank(pageRank);
29     int classement[NUM_NODES];
30     rank(pageRank, classement);
31     printRankedPageRank(pageRank, classement);
32
33     return 0;
34 }

```

Résultat après compilation du code :

```
Matrice d'adjacence:
0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
Classement des pages par PageRank:
Page 2: 0.361957
Page 3: 0.325793
Page 5: 0.078550
Page 4: 0.051514
Page 6: 0.051514
Page 1: 0.040023
Page 7: 0.018130
Page 8: 0.018130
Page 9: 0.018130
Page 10: 0.018130
Page 11: 0.018130
PS C:\Users\NADA\OneDrive\Bureau\c project>
```

- La **page 2** occupe la première position avec le score de PageRank le plus élevé. Cela est dû à au liens entrants depuis la page 3, qui est elle-même liée à la page 2. La forte connectivité avec d'autres pages importantes contribue à son score élevé.

- La **page 3** est la deuxième page la plus importante. Elle est liée à la page 2, ce qui contribue à son score élevé. Bien que la page 3 ait un lien sortant vers la page 2, cela n'a pas autant d'impact que la connectivité entrante depuis la page 2.

- La **page 5** occupe la troisième position. Elle est liée à plusieurs pages, y compris la page 4 et la page 6. Cependant, la connectivité avec la page 4 semble avoir plus d'importance, ce qui contribue à son score de PageRank relativement élevé.

- Les **pages 4 et 6** ont des scores de PageRank identiques. Elles sont liées à la page 5, mais également à d'autres pages.

- Les **pages 1, 7, 8, 9, 10, 11** ont des scores de PageRank identiques et occupent les positions les moins importantes dans le classement. Elles sont soit peu connectées ou connectées à des pages moins importantes, ce qui contribue à leurs scores relativement bas.

6.CONCLUSION :

Il est important de noter que l'algorithme PageRank n'est pas parfait. Il peut être influencé par des facteurs tels que la qualité des liens et la popularité des pages. De plus, il peut être difficile d'interpréter les résultats de l'algorithme PageRank, car ils sont basés sur une distribution de probabilité.

Malgré ses limites, l'algorithme PageRank reste un outil précieux qui peut être utilisé pour améliorer la qualité des résultats de recherche et la pertinence des recommandations.