#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#define MAX\_STRING 40

#define SIZE\_BUFFER 1000000

#define SIZE\_BUFFER2 150

void lecture\_fichier(FILE\* fp, char\*\* node, double\*\* matrix, int lines) {

char buffer[SIZE\_BUFFER];

char \*token;

int i = 0;

int j = 0 ;

fgets(buffer, sizeof(buffer), fp); //1ere ligne, noms des noeuds

token = strtok(buffer, " \n");

j=0;

while (token) {

strcpy(node[j], token) ;

++j;

token = strtok(NULL, " \n");

}

while (fgets(buffer, sizeof(buffer), fp) != NULL) {

j = 0;

token = strtok(buffer, " \n");

while (token) {

matrix[i][j] = atof(token);

++j;

token = strtok(NULL, " \n");

}

++i;

}

}

int size\_matrix(FILE\* fp) {

int c, lines=-1; //On ne compte pas la ligne contenant les noms des noeuds

while((c = fgetc(fp)) != EOF)

{

if(c == '\n')

lines++;

}

rewind(fp);

return lines ;

}

void build\_matrix(double\*\* matrix, int lines) {

for(int j=0;j<lines;j++)

matrix[j]=(double\*)malloc(lines\*sizeof(double));

}

void clear\_matrix(double\*\* matrix, int lines) {

for(int j=0;j<lines;j++)

free(matrix[j]);

free(matrix) ;

}

void affichage\_matrix (double\*\* matrix, int taille) {

for (int i = 0; i < taille; i++) {

for (int j = 0; j < taille; j++) {

printf("%.3lf ", matrix[i][j]);

}

putchar('\n');

}

putchar('\n');

}

void build\_node (char\*\* node, int lines) {

for(int j=0;j<lines;j++)

node[j]=(char\*)malloc(MAX\_STRING);

}

void clear\_node (char\*\* node, int lines) {

for(int j=0;j<lines;j++)

free(node[j]);

free(node);

}

void affichage\_node (char\*\* node, int taille) {

for (int i = 0; i < taille ; i++) {

printf("%d :: %s \n", i, node[i]);

}

}

double\* prod\_scal\_vec (double a, double\*x, int size\_x){

for(int i=0;i<size\_x;i++){

x[i]=a\*x[i];

}

return (x);

}

double\* prod\_mat\_vec(double\*\* m, double\*x,int N){

double\* temp=(double\*)malloc(N\*sizeof(double));

for(int j=0;j<N;j++){

temp[j]=0;

}

for(int j=0;j<N;j++){

temp[0]=temp[0]+m[0][j]\*x[j];

}

for(int i=1;i<N;i++){

for(int j=0;j<N;j++){

temp[i]=temp[i]+m[i][j]\*x[j];

}

}

return(temp);

}

double\* normalize\_vec(double\*x, int N){

double somme = x[0];

for(int i=1;i<N;i++){

somme = somme + x[i];

}

for(int i=0;i<N;i++){

x[i]=x[i]/somme;

}

return(x);

}

void trier\_indice(int\* ind, double\* vec, int size) {

int ind\_deja\_tire[size] ;

double max;

int indice;

for (int i=0 ; i<size ; i++) {

ind\_deja\_tire[i] = 0 ;

}

for (int i=0 ; i<size ; i++) {

indice = -1 ;

max=-1.0;

for (int j=0 ; j<size ; j++) {

if (ind\_deja\_tire[j]==0) {

if (vec[j]>max) {

indice=j ;

max=vec[j] ;

}

}

}

if(indice != -1)

{

ind\_deja\_tire[indice]=1 ;

ind[i]=indice;

}

}

}

double diff\_vec\_vec(double\*a,double\*b,int lines){

double diff=fabs(a[0]-b[0]);

for(int i=1;i<lines;i++){

diff = diff + fabs(a[i]-b[i]);

}

printf("Indice de convergence : %.10lf\n\n", diff);

return(diff);

}

int main(int argc, char\* argv[])

{

/\* Lecture de fichier\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

if (argc < 2) {

printf("Erreur, format attendu: %s fichier.txt\n", argv[0]); // ?

exit(EXIT\_FAILURE);

}

FILE \*fp;

fp = fopen(argv[1],"r");

if ( fp == NULL ){

puts ("Acces au fichier impossible");

}

int lines = size\_matrix(fp) ;

double\*\* matrix = (double \*\*)malloc(lines\*sizeof(double\*));

build\_matrix(matrix, lines) ;

char\*\* node = (char \*\*)malloc(lines\*sizeof(char\*)) ;

build\_node(node, lines) ;

lecture\_fichier(fp, node, matrix, lines) ;

fclose(fp);

// affichage\_node(node,10) ;

//affichage\_matrix(matrix,10) ;

/\*Fin de lecture du fichier\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

/\*Algorithme Pagerank\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

//Initialisation du paramÃƒÂ©trage pagerank

double beta;

printf("Entrer la valeur du parametre beta, entre 0 et 1 : ") ;

scanf("%lf", &beta) ;

printf("%.3lf \n",beta);

double diff;

int diffp ;

printf("\nEntrer la puissance p de l'indice de convergence seuil 10e-p : ") ;

scanf("%d", &diffp) ;

diff = pow(10,(-1)\*diffp);

printf("%.10lf \n", diff) ;

//Initialisation du vecteur initial

double\* q=(double\*)malloc(lines\*sizeof(double));

int value ;

printf("\nPreciser pageranking classique (0) ou personnalise (1) : ") ;

scanf("%d", &value) ;

int nb ;

int\* perso;

if (value==0) {

for (int i=0;i<lines;i++){

q[i] = 1.0/lines;

}

nb=lines;

}

else if (value==1) {

printf("\nPreciser le nombre de pages a personnaliser : ") ;

scanf("%d", &nb) ;

perso=(int\*)malloc(nb\*sizeof(int));

for(int i=0;i<nb;i++){

printf("\nDonner l'indice de la page %d a personnaliser : ",i+1) ;

scanf("%d", &perso[i]) ;

}

printf("%d \n",perso[0]);

for(int i=0;i<lines;i++){

q[i] = 0;

}

for(int i=0;i<nb;i++){

q[perso[i]] = 1.0/nb ;

}

fflush( stdout );

printf("%.3lf \n",q[36]);

}

printf("\n") ;

//Initialisation des derniers paramÃƒÂ¨tres

double constante = (1.0-beta)/nb; // (1-beta / N)

double\* q\_prec=(double\*)malloc(lines\*sizeof(double)); //q\_precedent donc q(k-1)

memmove (q\_prec, q, lines\*sizeof(double)); // Copie de q dans q\_prec (donc q(k-1) = q(k))

//Algorithme pagerank

double somme ;

int iter = 0;

do{

memmove (q\_prec, q, lines\*sizeof(double)); // Copie de q dans q\_prec (donc q(k-1) = q(k))

//Ajout du premier terme de q

q = prod\_scal\_vec(beta,prod\_mat\_vec(matrix,q\_prec, lines), lines) ;

// Calcul somme q\_prec[i]

somme = 0;

for(int j=0;j<lines;j++){

somme = somme + q\_prec[j];

}

somme = somme\*constante;

//Ajout du second terme de q

if (value==1) {

for (int k=0 ; k<nb ; k++) {

q[perso[k]] = q[perso[k]] + somme ;

}

}

else{

for (int k=0 ; k<lines ; k++) {

q[k] = q[k] + somme ;

}

}

// Normalisation q

q=normalize\_vec(q,lines);

iter++;

printf("Iteration : %d\n",iter);

}while(diff\_vec\_vec(q,q\_prec,lines) > diff );

printf("\n\n") ;

//Affichage des sites ayant les plus gros scores

int\* ind\_trier=(int\*)malloc(lines\*sizeof(int));

trier\_indice(ind\_trier, q, lines) ;

printf("Les 20 sites ayant les meilleurs rangs :\n\n") ;

for(int i=0 ; i<20 ; i++) {

printf("\t%s (%d) :: %.6lf\n", node[ind\_trier[i]], ind\_trier[i], q[ind\_trier[i]]);

}

printf("\n\n") ;

/\*Fin de l'algorithme Pagerank\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

FILE \* fsortie;

fsortie = fopen("vecteur.txt", "w");

if(fsortie == NULL){

printf("Creation du fichier impossible\n");

exit(EXIT\_FAILURE);

}

for(int i=0;i<lines;i++){

fprintf(fsortie, "%s \"%.15lf\n\"",node[i],q[i]);

}

fclose(fsortie);

if (value==1)

free(perso) ;

free(ind\_trier) ;

free(q) ;

free(q\_prec) ;

//clear\_matrix(matrix, lines) ;

//clear\_node(node,lines) ;

return(0);

}