TD 1

# Exo 1

recherche.liste(l,k) O(n)

x = l.tete

TANTQUE x != NULL ET x.clé != k

x = x.succ

Retourner x

inserer.liste(l,k) O(1)

x.secc = l.tete

SI l.tete != NULL

l.tete.pred = x

l.tet = x

x.pred = NULL

supprimer.liste(l,x) O(1)

SI x.pred != NULL

x.pred.succ = x.succ

SINON

l.tete = x.succ

SI x.succ != NULL

x.succ.pred = x.pred

# Exo 2

3.a) O(S+A)

3.b) O(S²)

4.a) O(degré(u))

4.b) O(S)

5.a) O(degré(u))

5.b) O(1)

# Exo 3

2)

cellule.h :

typedef struct cellule\_type{

    int id\_sommet;

    struct cellule\_type \*pred;

    struct cellule\_type \*succ;

}cellule\_t;

int initialiser\_cellule(cellule\_t\* c, int id\_sommet);

cellule.c :

int initialiser\_cellule(cellule\_t\* c, int id\_sommet){

    ...

    return 0;

}

liste.h :

typedef struct{

    cellule\_t\* tete;

}liste\_t;

int initialiser\_liste(liste\_t\* l);

int inserer(liste\_t\* l, cellule\_t\* c);

...

liste.c :

progListe.c :

int main(...){

    cellule\_t cl;

    initialiser\_cellule(&cl,4);

    cellule\_t\* iz;

    iz=(cellule\_t\*) malloc(sizeof(cellule\_t));

    initialiser\_cellule(iz,4);

}

graphe.h :

typedef struct{

    int n\_sommets;

    int oriente;

    int value;

    liste\_t\* l\_adj;

    int \*\* m\_adj;

}graphe\_t;

int initialiser\_graphe(graphe\_t\* g, char \* nom\_fichier);

int initialiser\_graphe(graphe\_t \* g, char \* nom\_fichier){

    FILE \* fichier;

    char ch\_temp[30];

    fichier = fopen(nom\_fichier, "r"); //lecture

    fscanf(fichier, "%s", ch\_temp);

    fscanf(fichier, "%d", g->n\_sommets)

    g->m\_stockage = (int\*) malloc(sizeof(int)\*g->n\_sommets\*g->n\_sommets);

    g->m\_adj = (int\*\*)malloc(sizeof(int\*)\*g->n\_sommets);

    f(i...){

        g->m\_adj[i]=&g->ù\_stockage[i\*g->n\_sommet];

    }

}

TD 2

**Attention,** certains NULL peuvent être des NIL jusqu'à exo 3.3, exclu

# Exercice 1

1. Schéma

pile\_vide(p)    O(1)

  si p.sommet == 0

    retourner vrai

  sinon

    retourner faux

empiler(p,x)    O(1)

  p.sommet = p.sommet +1

  p[p.sommet] = x

depiler(p)     O(1)

  si pile\_vide(p)

    erreur("débordement négatif")

  sinon

    p.sommet = p.sommet-1

    retourner p[p.sommet+1]

# Exercice 2

1. Schéma

enfiler(f,x) O(1)

  f[f.queue] = x

  si f.queue == f.longueur

    f.queue = 1

  sinon

    f.queue = f.queue +1

defiler(f) O(1)

  x = f[f.tete]

  si f.tete == f.longueur

    f.tete = 1

  sinon

    f.tete = f.tete +1

  retourner x

# Exercice 3

1. Schéma

parcours\_largeur(G,s) O(S+A)

  Pour chaque sommet u appartenant à G.s

    u.couleur = BLANC

    u.d = infinie (max int)

    u.pere = NULL

  s.couleur = GRIS

  s.d = 0

  s.pere = NULL

  f = NULL

  enfiler(f,s)

  Tant que f != NULL

    u = defiler(f)

    Pour chaque sommet v appartenant à G.adj[u]

      si v.couleur == BLANC

        v.couleur = GRIS

        v.d = u.d +1

        v.pere = u

        enfiler(f.v)

    u.couleur = NOIR

intialisation : O(S)

enfiler/defiler : O(S)

balayage des listes d'adjacences : O(A)

* O(S+A)



afficher\_chemin(G,s,v)  O(S) //Graphe G, Sommet départ S, Sommet D'arrivé V

  si v == s

    afficher s

  sinon

    si u.pere == NIL

      afficher "aucun chemin de s à v"

    sinon

      afficher\_chemin(G,s,v.pere)

      afficher v

# Exercice 4

Schéma

parcours\_profondeur(G)  O(S+A)

  pour chaque sommet u appartenant à G.s

    u.couleur = BLANC

    u.pere = NIL

  date = 0

  pour chaque sommet u appartenant à G.s

    s u.couleur == BLANC

    visiter.pp(G,u)

visiter\_pp(G,u)

  date = date +1

  u.d = date

  u.couleur == GRIS

  pour chaque sommet v appartenant à G.adj[u]

    si v.couleur == BLANC

      v.pere = u

      visiter\_pp(G,v)

  u.couleur = NOIR

  date = date+1

  u.f = date