ENSTA Paris
2022-2023

Résolution de problème algorithmique

Examen

PRÉAMBULE

Consignes générales (A LIRE IMPÉRATIVEMENT)

- **Lisez attentivement les consignes et tout le sujet avant de commencer.**

- Les documents (polys, transparents, TDs, livres ...) sont autorisés.
- Sont **absolument interdits** : le WEB, le courrier électronique, les messageries diverses et variées, le répertoire des camarades, le téléphone (même pour avoir l'heure puisque vous l'avez sur votre ordinateur).

- Votre travail sera (en partie) évalué par un mécanisme automatique. Vous **devez** respecter les règles de **nommage** des fichiers et autres **consignes** qui vous sont données.
- La connaissance de C est un pré-requis d'IN103. Ainsi, la présence d'avertissement(s) ou d'erreur(s) à la compilation réduira mécaniquement la note de l'exercice considéré.
- Lorsqu'il vous est demandé que votre programme réponde en affichant « **Yes** » ou « **No** », il ne doit **rien** afficher d'autre, et **pas** « Oui » ou « Yes. » ou « no » ou « La réponse est : no ». Donc, pensez à retirer vos affichages de test / debug.
- La **lisibilité** et l'efficacité / simplicité / complexité de vos programmes seront **prises en compte** dans l'évaluation.

- **À la fin de l'examen**, vous devrez créer une **archive** contenant **tous** les fichiers **sources** que vous avez écrits (`.c`, `.h`). **N'incluez pas** d'exécutables dans l'archive, le mail pourrait la considérer comme un attachement dangereux et le supprimer. Le nom de cette archive devra avoir la structure suivante :

  `nom_prenom.zip` ou `.tgz` (selon l'outil d'archivage que vous utilisez).

- Vous devrez **m'envoyer** cette archive par mail. En cas d'envoi incorrect, il vous sera demandé de refaire l'archive et l'envoi. Par contre, vous ne devrez **surtout pas modifier** les fichiers : leurs dates de dernière modification ne devra pas être ultérieure à l'heure de fin de l'épreuve sous peine d'être considérés comme nuls.
- **N'oubliez pas** d'effectuer cet envoi sinon nous devrons considérer que vous n'avez rien rendu sur la partie programmation !
- **Il y aura également des questions dont les réponses sont à rendre sur une copie papier**.

- Le sujet comporte 16 pages et l'examen dure **3h00**.
- Le barème est donné à titre indicatif.

Nouvelle version de la bibliothèque `libin103`

La version **1.4** est la nouvelle version de la bibliothèque `libin103` qui corrige quelques bugs. Il faut donc mettre à jour votre environnement de travail en suivant les étapes données ci-dessous.

1. A la racine de votre compte, créez un répertoire nommé `Library` s'il n'a pas déjà été créé, puis placez vous dans ce répertoire.

   ```
   mkdir ~/Library; cd ~/Library
   ```

2. Téléchargez l'archive `libin103-1.4.tar.gz` sur le site du cours

   wget `https://perso.ensta-paris.fr/~chapoutot/teaching/in103/practical-work/libin103-1.4.tar.gz`

3. Désarchivez l'archive

   ```
   tar -xvzf libin103-1.4.tar.gz
   ```

4. Allez dans le répertoire `libin103-1.4` et compilez la bibliothèque.

   - Il faut utiliser la commande `make`. A la fin de la compilation vérifier la présence du fichier `libin103.a` dans le répertoire `source`.
   - Également, vous pouvez exécuter la commande `make check` pour compiler et exécuter les programmes de tests.

Matériel pour l'examen

Récupérez l'archive associé à cette séance de TP à l'adresse :

```
https://perso.ensta-paris.fr/~chapoutot/teaching/in103/practical-work/
                    in103-examen-material.tar.gz
```

Comme pour les TP, pour la plus part des exercices, un répertoire est associé à chaque exercice dans lequel se trouve un fichier Makefile et un ou plusieurs fichiers code source. Ce fichier Makefile a été configuré pour utiliser la dernière version de la bibliothèque `libin103` (version 1.4).

PARTIE À RENDRE

Exercice 1 – Pourquoi cela ne marche pas ? (2 pts)

Le matériel pour cet exercice est donné dans le répertoire `exo1`.

Dans le répertoire, un code qui permet d'évaluer une expression arithmétique en format postfixe est fourni. Cette évaluation utilise une structure de données de type pile. L'algorithme est le suivant :

- On lit une chaîne de caractères représentant une expression arithmétique au format postfixe (p. ex., "3 2 +").
  - Quand on lit un nombre on l'empile
  - Quand on lit un symbole d'opération, on dépile les arguments nécessaires pour réaliser l'opération, on fait le calcul et on empile le résultat
- Quand on a fini de lire l'expression arithmétique, le résultat est sur le haut de la pile.

La mise en œhure informatique de cet algorithme est entachée d'erreurs qui génère un "segmentation fault"

Question 1

Corrigez le code donné pour que le programme se comporte correctement.

Solution:

```
int main (){

  int size = 7;
  char tab[] = { '3', '4', '-', '5', '+', '9', '*' };
  integer_stack_t stack;

  integer_stack_init(&stack);

  for (int i = 0; i < size; i++) {
    printf ("Element = %c\n", tab[i]);
    switch (tab[i]) {
    case '0':
    case '1':
    case '2':
    case '3':
    case '4':
    case '5':
    case '6':
    case '7':
    case '8':
    case '9':
      integer_stack_push (&stack, atoi(char2str(tab[i])));
      break;

    case '+': {
      int a;
      integer_stack_pop (&stack, &a);
      int b;
      integer_stack_pop (&stack, &b);
      integer_stack_push (&stack, b+a);
      break;
    }

    case '-': {
      int a;
      integer_stack_pop (&stack, &a);
      int b;
      integer_stack_pop (&stack, &b);
      integer_stack_push (&stack, b-a);
      break;
    }

    case '*': {
      int a;
      integer_stack_pop (&stack, &a);
      int b;
      integer_stack_pop (&stack, &b);
      integer_stack_push (&stack, b*a);
      break;
    }

    case '/': {
      int a;
      integer_stack_pop (&stack, &a);
      int b;
      integer_stack_pop (&stack, &b);
      integer_stack_push (&stack, b/a);
      break;
    }

    default:
      fprintf (stderr, "Should not happen %c!\n", tab[i]);
      return EXIT_FAILURE;
      break;

    }
  }

  int result;
  integer_stack_pop (&stack, &result);
  printf ("result = %d\n", result);

  integer_stack_destroy(&stack);

  return EXIT_SUCCESS;
}
```

Exercice 2 – Et si on automatisait la compilation avec un `Makefile` ? (3 pts)

Le matériel pour cet exercice est donné dans le répertoire `exo2`.

Nous considérons dans cet exercice le seul exercice dont le code vous est fourni sans fichier `Makefile` pour générer le programme exécutable. L'objectif donc est décrire un fichier `Makefile` pour automatiser la compilation du programme `principal.x` qui dépend de :

- `interval.h` / `interval.c` contiennent la déclaration et définition du type de données pour représenter des intervalles fermés et les opérations arithmétiques d'addition, soustraction et multiplication sur les intervalles fermés.
- `display.h` / `display.c` contiennent la déclaration et définition d'une fonction `interval2string` qui permet de représenter sous la forme d'une chaîne de caractères un intervalle fermé

- `principal.c` qui contient la fonction principale du programme.

Le matériel pour cet exercice est donné dans le répertoire `exo2`.

Un exemple d'exécution du programme compilé est

```
alex@MacAlex exo2 % ./principal.x
x = [1.00; 2.00]
y = [-2.00; -1.00]
add(x, y) = [-1.00; 1.00]
sub(x, y) = [2.00; 4.00]
mul(x, y) = [-4.00; -1.00]
```

Question 1

Donnez les lignes (sur la feuille) de compilation pour générer les fichiers `.o` à partir des fichiers `.c` donnés dans le répertoire `exo2`. Nous utiliserons les options `-Wall` et `-Werror` de la commande `gcc`.

Solution:

- `gcc -Wall -Werror -c interval.c`
- `gcc -Wall -Werror -c display.c`
- `gcc -Wall -Werror -c principal.c`
- `gcc -Wall -Werror -o principal.x interval.o display.o principal.o`

Question 2

Écrivez le fichier `Makefile` pour automatiser la génération du fichier exécutable `principal.x`. On ne cherchera pas forcément à simplifier les règles.

Solution:

```
all: principal.x

interval.o: interval.c
	gcc -Wall -Werror -c $^

display.o: display.c
	gcc -Wall -Werror -c $^

principal.o: principal.c
	gcc -Wall -Werror -c $^

principal.x: interval.o display.o principal.o
	gcc -Wall -Werror -o $@ $^ -lm
```

Question 3

Ajoutez les règles `clean` et `realclean` qui permettent respectivement de supprimer les fichiers `.o` générés et de supprimer le fichier exécutable `principal.x` (en plus de supprimer les fichiers `.o`).

Solution:

```
clean:
	rm -f *.o *~

realclean: clean
	rm -f principal.x
```

Exercice 3 – Détecteur de palindromes (4 pts)

Le matériel pour cet exercice est donné dans le répertoire `exo3`.

Dans la feuille 2 des TPs a été donné un exercice pour détecter les chaînes de caractères palindromiques. Cet exercice est une variation de ce problème dans lequel il faut détecter des palindromes dans des listes chaînées de caractères.

Le code source fourni dans le répertoire `exo3` a été légèrement modifié pour lire une chaîne de caractères sur la ligne de commande et pour la transformer en une liste chaînée de caractères. Ci-dessous, quelques exemples de fonctionnement du programme :

```
alex@MacAlex exo3 % ./is-palindrome.x
Usage ./is-palindrome.x string
alex@MacAlex exo3 % ./is-palindrome.x aca
The string "aca" is a palindrome
alex@MacAlex exo3 % ./is-palindrome.x abcd
The string "abcd" is not a palindrome
```

Question 1

Définissez le code de la fonction qui vérifie si la liste chaînée de caractère est palindromique, son prototype est

```
bool is_palindrome (character_list_t* str);
```

Solution:

```
bool is_palindrome (character_list_t* str) {

  int len = character_list_size (str);
  character_stack_t stack;
  character_stack_init (&stack);

  character_list_elmt_t* c = character_list_head (str);
  int i = 0;
  while (i < len / 2) {
    char cd = character_list_data (c);
    character_stack_push (&stack, cd);
    c = character_list_next (c);
    i++;
  }
  if (len % 2 == 1) {
    /* Ne pas considerer le caractere du milieu */
    c = character_list_next (c);
    i++;
  }
  while (i < len) {
    char cs;
    character_stack_pop (&stack, &cs);
    char cd = character_list_data (c);
    if (cs != cd) {
      character_stack_destroy (&stack);
      return false;
    }
    c = character_list_next (c);
    i++;
  }

  character_stack_destroy (&stack);
  return true;
}
```

Exercice 4 – Variation sur les piles (6 pts)

Le matériel pour cet exercice est donné dans le répertoire `exo4`.

En avant propos, un petit exercice de réflexion sur les piles. On souhaite définir une structure de données de type pile (ou FILO, first-in last-out) avec les fonctions classiques `push` et `pop` et en plus, un accès (avec une complexité en $O(1)$ à la moyenne des éléments de la pile. On supposera dans cet exercice que nous nous ne considérerons que des piles d'entiers.

Question 1
Expliquez brièvement comment vous allez procéder pour répondre à ce problème. Quel type de structure de données ? Quels sont les principaux comportements à gérer ?

Question 2
Définissez une structure de données qui regroupe toutes les informations nécessaires pour répondre à la spécification. Pour être conforme à l'exemple d'utilisation donné dans la fonction `main`, nous appellerons ce nouveau type `MPile_t`.

Solution:
```
typedef struct MPile_ {
  int somme;
  integer_stack_t stack;
} MPile_t;
```

Question 3
Définissez les fonctions `init` et `destroy` qui agissent sur la structure définie à la question précédente et qui respectivement l'initialise et la détruit. Les prototypes de ces fonction sont :

- `void init (MPile_t *p);`
- `void destroy (MPile_t *p);`

Solution:
```
void init (MPile_t *p) {
  p->somme = 0;
  integer_stack_init (&(p->stack));
}

void destroy (MPile_t *p) {
  p->somme = 0;
  integer_stack_destroy (&(p->stack));
}
```

Question 4
Définissez les fonctions `push` et `pop` qui permettent respectivement d'ajouter un élément sur la pile et de supprimer l'élément en sommet de pile. Les prototypes de ces fonction sont :

- `void push (MPile_t *p, int elem);`
- `int pop (MPile_t *p);` Cette fonction retourne l'élément dépilé.

Solution:
```
void push (MPile_t *p, int elt) {
  p->somme += elt;
  integer_stack_push (&(p->stack), elt);
}

int pop (MPile_t *p) {
  int temp;
  integer_stack_pop (&(p->stack), &temp);
  p->somme -= temp;
  return temp;
}
```

Question 5

Définissez les fonctions `mean` et `size` qui permettent respectivement d'accéder à la valeur moyenne des éléments de la pile et la taille de la pile (son nombre d'éléments). Les prototypes de ces fonction sont :

- `double mean (MPile_t *p);`
- `int size (MPile_t *p);`

Solution:

```
double mean (MPile_t *p) {
  if (integer_stack_size (&(p->stack)) > 0) {
    return ((double)p->somme) / integer_stack_size (&(p->stack));
  }
  return 0.0;
}

int size (MPile_t *p) {
  return integer_stack_size (&(p->stack));
}
```

Exercice 5 – Variation sur les acrostiches (6 pts)

Le matériel pour cet exercice est donné dans le répertoire `exo5`.

Un acrostiche est d'après Wikipédia[1]

> un poème, une strophe ou une série de strophes fondés sur une forme poétique consistant en ce que, lues verticalement de haut en bas, la première lettre ou, parfois, les premiers mots d'une suite de vers composent un mot ou une expression en lien avec le poème.

Dans cet exercice, nous allons considérer un petit texte donné sous la forme d'une matrice de chaînes de caractères :

```
char* strings[5][5] = { {"wize", "students", "work", "very", "tremendously"},
                        {"beautiful", "blue", "flowers", "of", "Earth"},
                        {"great", "place", "is", "ENSTA", "Paris"},
                        {"work", "happily", "to", "be", "here"},
                        {"hard", "time", "to", "spend", "here"}
};
```

Dans ces données est caché un message suivant le codage :

- pour les lignes de numéros impairs, il faut afficher le mot le plus grand selon l'ordre lexicographique (ordre du dictionnaire).
- pour les lignes de numéros pairs, il faut afficher le mot le plus petit selon l'ordre lexicographique.

Remarque : on compte les lignes à partir de 0 et les lettres majuscules apparaissent avant les lettres minuscules dans l'ordre lexicographique.

Question 1

Essayez de décoder le message à la main et inscrivez la réponse sur votre copie.

Solution:
« students of ENSTA work hard »

Question 2

On vous propose d'utiliser un arbre binaire de recherche équilibré pour décoder automatiquement le message caché.

[1] https ://fr.wikipedia.org/wiki/Acrostiche

On suppose pour le moment que l'on peut manipuler des chaînes de caractères dans les AVL. Décrivez brièvement l'algorithme de vous mettriez en œuvre pour décoder automatiquement le message caché.

Solution:
Pour chaque ligne, j'ajoute les mots dans un AVL. Puis je parcours l'AVL dans l'ordre infixe pour avoir une liste de mots triés. Enfin, en fonction du numéro de ligne, je choisi le premier ou le dernier mot à afficher.

Question 3

Afin d'utiliser, des AVL manipulant des chaînes de caractères, une fonction de comparaison et de création ont été définies dans le fichier `sort-strings.c` du répertoire `exo5`. En conséquence, il faut utiliser dans cet exercice la version générique des AVL dont l'API est

```
typedef struct generic_avlnode_t {
  void               *data;
  int                hidden;
  int                factor;

} generic_avlnode_t;

typedef generic_bitree_t generic_bistree_t;

void generic_bistree_init(generic_bistree_t *tree,
                          int (*compare)(const void *key1, const void *key2),
                          void* (*build) (const void *data),
                          void (*destroy)(void *data));

void generic_bistree_destroy(generic_bistree_t *tree);

int generic_bistree_insert(generic_bistree_t *tree, void *data);

int generic_bistree_remove(generic_bistree_t *tree, void *data);

int generic_bistree_lookup(generic_bistree_t *tree, void **data);

int generic_bistree_size(generic_bistree_t* tree);
```

Écrivez un programme qui permet d'afficher le message caché en vous appuyant sur cette API pour la partie manipulation d'AVL et l'algorithme que vous avez défini à la précédente question.

Solution:

```
int main () {

  char* strings[5][5] = { {"wize", "students", "work", "very", "tremendously"},
                         {"beautiful", "blue", "flowers", "of", "Earth"},
                         {"great", "place", "is", "ENSTA", "Paris"},
                         {"work", "happily", "to", "be", "here"},
                         {"hard", "time", "to", "spend", "here"}
  };

  for (int i = 0; i < 5; i++) {
    generic_bistree_t tree;
    generic_bistree_init (&tree, compare, build, free);

    for (int j = 0; j < 5; j++) {
      generic_bistree_insert (&tree, strings[i][j]);
    }
    generic_list_t t_inorder;
    generic_list_init (&t_inorder, NULL, NULL, NULL);
    generic_bitreenode_t* root = generic_bitree_root (&tree);
    generic_inorder (root, &t_inorder);

    if (i % 2 == 0) {
      generic_list_elmt_t *elt = generic_list_head(&t_inorder);
      printf ("%s\n", (char*)((generic_avlnode_t *) generic_list_data(elt))->data);
    }
    else {
      generic_list_elmt_t *elt = generic_list_tail(&t_inorder);
      printf ("%s\n", (char*)((generic_avlnode_t *) generic_list_data(elt))->data);
    }

    generic_list_destroy (&t_inorder);
    generic_bistree_destroy (&tree);
  }

  return EXIT_SUCCESS;
}
```

Exercice 6 – Réseau informatique (6 pts)

Le matériel pour cet exercice est donné dans le répertoire `exo6`.

Un réseau informatique connecte plusieurs ordinateurs qui doivent communiquer entre eux. La latence représente le délai de communication, plus elle est faible moins de temps un message met pour arriver à destination. On considère un étage de bureaux donné à la figure 1. Le câblage réseau s'est fait au fur et à mesure des années si bien que différentes technologies sont présentes dans l'infrastructure avec des latences différentes et il y a une disparité dans les connexions (à l'aide de routeurs) entre les différentes pièces de l'étage.

La salle serveur a une connexion directe avec tous les bureaux. Le bureau de Danil est connecté au bureau de Clément qui est connecté au bureau de Caroline, qui est lui même connecté au bureau de Thibault qui est connecté au bureau de Gwendal. La salle de réunion, l'open space et la salle du conseil forme un graphe complet au niveau du réseau. La salle du conseil est reliée au bureau de Gwendal comme la salle de réunion. Et la salle de réunion est connecté au bureau de Danil.

Les latences de segments de réseaux sont les suivantes :

- La latence entre l'open space, la salle de conseil et la salle de réunion est de 6 sur chaque segments.
- On a une latence de 5 entre la salle serveur et le bureau de Thibault ainsi qu'entre les bureaux de Clément et Caroline.
- On a une latence de 1 entre la salle du conseil et le bureau de Gwendal et entre la salle serveur et le bureau de Gwendal.

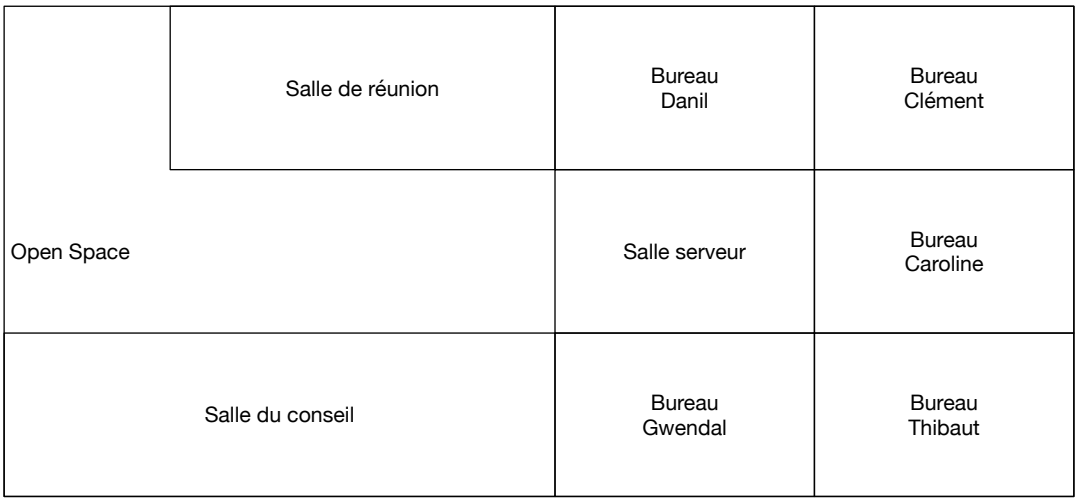


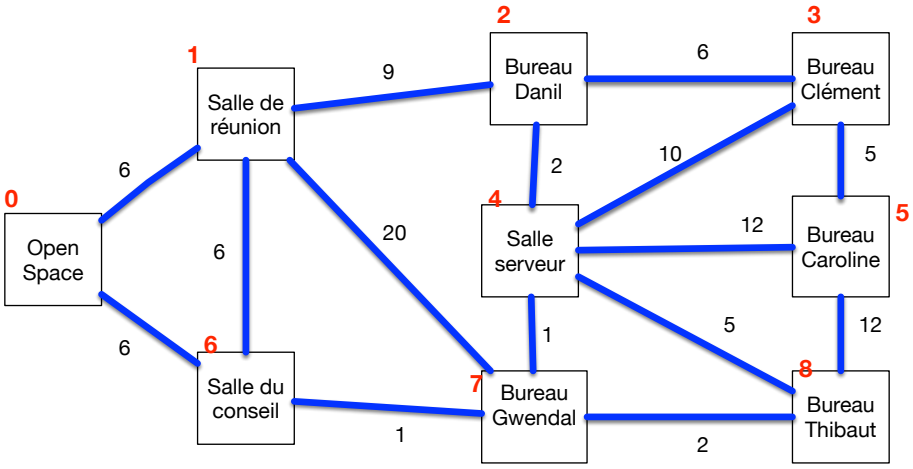
FIGURE 1 : Plan étage de bureaux

- On a une latence de 20 entre le bureau de Gwendal et la salle de réunion qui a aussi une latence de 9 avec le bureau de Danil.
- On a une latence de 2 entre la salle serveur et le bureau de Danil et entre le bureau de Gwendal et le bureau de Thibault.
- On a une latence de 12 entre la salle serveur et le bureau de Caroline et entre ce dernier et le bureau de Thibault.
- On a une latence de 10 entre la salle serveur et le bureau de Clément.
- Pour finir une latence de 6 est présente entre les bureaux de Danil et Clément.

Question 1

Modélisez le réseau informatique de cet étage par un graphe pondéré (on considère que les connexions réseaux sont bidirectionnelles).

Solution:



Question 2

Nous considérons une communication entre Clément et une personne dans l'open space. Nous cherchons à calculer le chemin qui sera emprunté par les messages qui minimise la latence. Quel algorithme allez-vous utiliser pour cela ? Donnez le chemin emprunté par les messages en déroulant à la main l'algorithme choisi (on présentera l'évolution du tableau des parents et du tableau des distances à la source).

Solution:
Algorithme de Dijkstra, plus court chemin d'une source à tous les autres sommets avec des pondération uniquement positive.
Il faut définir une numérotation des pièces pour représenter le graphes avec des sommets à valeur entière. Le chemin est

- $3 -> 2 -> 4 -> 7 -> 6 -> 0$
- Évolution algorithme à faire

cf le cours pour développer l'exécution de l'algorithme.

Question 3

Mettez en œuvre informatiquement l'algorithme que vous avez choisi à la question précédente.

Solution:

```
int main (int argc, char** argv) {
  integer_graph_t graph;
  integer_graph_init (&graph);

  for (int i = 0; i < 9 ; i++) {
    integer_graph_ins_vertex(&graph, i);
  }
  /* The graph is not directed so we enter the edges in one direction
     and in the other direction */
  integer_graph_ins_edge(&graph, 0, 1, 6.0); integer_graph_ins_edge(&graph, 1, 0, 6.0);
  integer_graph_ins_edge(&graph, 0, 6, 6.0); integer_graph_ins_edge(&graph, 6, 0, 6.0);

  integer_graph_ins_edge(&graph, 1, 2, 9.0); integer_graph_ins_edge(&graph, 2, 1, 9.0);
  integer_graph_ins_edge(&graph, 1, 6, 6.0); integer_graph_ins_edge(&graph, 6, 1, 6.0);
  integer_graph_ins_edge(&graph, 1, 7, 20.0); integer_graph_ins_edge(&graph, 7, 1, 20.0);

  integer_graph_ins_edge(&graph, 2, 4, 2.0); integer_graph_ins_edge(&graph, 4, 2, 2.0);
  integer_graph_ins_edge(&graph, 2, 3, 6.0); integer_graph_ins_edge(&graph, 3, 2, 6.0);

  integer_graph_ins_edge(&graph, 3, 4, 10.0); integer_graph_ins_edge(&graph, 4, 3, 10.0);
  integer_graph_ins_edge(&graph, 3, 5, 5.0); integer_graph_ins_edge(&graph, 4, 5, 5.0);

  integer_graph_ins_edge(&graph, 4, 5, 12.0); integer_graph_ins_edge(&graph, 5, 4, 12.0);
  integer_graph_ins_edge(&graph, 4, 7, 1.0); integer_graph_ins_edge(&graph, 7, 4, 1.0);
  integer_graph_ins_edge(&graph, 4, 8, 5.0); integer_graph_ins_edge(&graph, 8, 4, 5.0);

  integer_graph_ins_edge(&graph, 5, 8, 12.0); integer_graph_ins_edge(&graph, 8, 5, 12.0);

  integer_graph_ins_edge(&graph, 6, 7, 1.0); integer_graph_ins_edge(&graph, 7, 6, 1.0);

  integer_graph_ins_edge(&graph, 7, 8, 2.0); integer_graph_ins_edge(&graph, 8, 7, 2.0);

  print_graph (&graph);
  printf ("\n\n");

  generic_list_t *paths;
  integer_shortest (&graph, 0, &paths);

  /* On recherche le chemin en transformant le resultat en tableau de parents */
  int* parents = malloc (9 * sizeof(int));
  if (parents == NULL) {
    generic_list_destroy (paths);
    integer_graph_destroy (&graph);
    return EXIT_FAILURE;
  }

  generic_list_elmt_t *elem = generic_list_head (paths);
  for (; elem != NULL; elem = generic_list_next (elem)) {
    ed_t* v = generic_list_data (elem);
    parents[v->vertex] = v->parent;
  }

  int u = 3;
  while (u != -1) {
    printf ("%d␣<-␣", u);
    u = parents[u];
  }
  printf ("\n");

  generic_list_destroy (paths);
  integer_graph_destroy (&graph);
  return EXIT_SUCCESS;
}
```

Exercice 7 – Correcteur orthographique (8 pts)

Le matériel pour cet exercice est donné dans le répertoire `exo7`.

On considère dans cet exercice la distance de Levenshtein qui permet de connaître la distance entre deux mots en terme de modification d'un mot vers un autre. Plus précisément, cette distance calcule le nombre de suppression de caractère, d'ajout de caractère et de substitutions entre deux mots. C'est une distance qui est très utilisée pour proposer des suggestions de correction de mots à partir d'un dictionnaire.

On se propose d'écrire un petit programme qui permet de trouver le mot le plus proche dans un dictionnaire d'animaux. Ce dictionnaire est décrit par le tableau suivant :

```
char* dictionnaire[] = { "belette", "blaireau", "cerf",
  "couleuvre", "chat", "campagnol", "daim", "ecureuil",
  "founine", "geai", "genette", "grenouille", "herisson",
  "hermine", "lapin", "loup", "lievre", "lynx", "loir",
  "martre", "merle", "mesange", "ours", "pinson", "pivert",
  "processionnaire", "pie", "putois", "renard", "salamandre",
  "sanglier", "sitelle", "taupe", "troglodyte", "vipere" };
```

.

Nous considérerons que le programme prend en paramètre de la ligne de commande un mot et renvoie la liste des mots du dictionnaire qui sont à distance inférieure ou égale à 2. Ci-dessous un exemple d'utilisation du programme

```
alex@MacAlex exo7 % ./levenshtein.x
Usage: ./levenshtein.x <chaine>
alex@MacAlex exo7 % ./levenshtein.x toto
Mot les plus proche:
alex@MacAlex exo7 % ./levenshtein.x sangler
Mot les plus proche: sanglier,
alex@MacAlex exo7 % ./levenshtein.x lop
Mot les plus proche: loup, loir,
```

Nous cherchons à concevoir un algorithme qui fait la suggestion de mots les plus proches qui soit dans la classe de complexité au maximum $O(n\log(n))$.

Question 1

Quelle structure de données vous semble adaptée pour mettre en œuvre cet algorithme ?

Solution:
Une structure de tas min ou rien (c'est-à-dire qu'un simple parcours de tableau suffit).

Question 2

Pour cet exercice excepté la fonction qui calcule la distance de Levenshtein et une proposition de structure de données pour représenter un couple (mot, distance), vous devez programmer complètement la solution.

Solution:

.

Exercice 8 – Amitiés (2 pts)

Exercice de réflexion sans matériel associé.

Étant donné un réseau social avec n membres et un fichier journal contenant m horodatages auxquels des paires de membres ont formé leur amitié, on cherche à concevoir un algorithme pour déterminer le moment le plus précoce auquel tous les membres sont connectés (c'est-à-dire que chaque membre est l'ami d'un ami d'un ami, ..., d'un ami). On supposera que le fichier journal est trié par horodatage croissant et que l'amitié est une relation d'équivalence.

Voici un exemple de fichier journal :

```
0 1 2023-03-14 16:00:00
1 9 2023-03-14 16:01:00
0 2 2023-03-14 16:02:00
0 3 2023-03-14 16:04:00
0 4 2023-03-14 16:06:00
0 5 2023-03-14 16:03:00
0 6 2023-03-14 16:10:00
0 7 2023-03-14 16:12:00
0 8 2023-03-14 16:14:00
1 2 2023-03-14 16:16:00
1 3 2023-03-14 16:18:00
1 4 2023-03-14 16:20:00
1 5 2023-03-14 16:22:00
2 1 2023-03-14 16:24:00
2 3 2023-03-14 16:26:00
2 4 2023-03-14 16:28:00
5 5 2023-03-14 16:30:00
```

La première colonne est le numéro du membre *i*, la seconde colonne est le numéro du membre *j* suivi d'une date et d'une heure.

Question 1

Quelle structure de données vue en cours sera utilisée pour concevoir cet algorithme ?

Solution:
une structure d'`union-find`

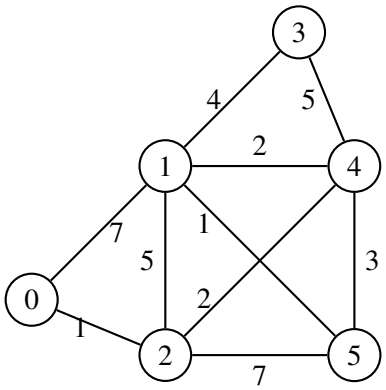
Question 2

Quel sera le critère d'arrêt de votre algorithme pour trouver la date la plus précoce ?

Solution:
L'algorithme s'arrêtera quand il n'y aura plus qu'une composante dans la structure d'`union-find`.

Exercice 9 – Optimisation (4 pts)

On considère le graphe non orienté pondéré suivant



Question 1

Décrivez l'exécution de l'algorithme de Kruskal sur ce graphe en explicitant l'évolution des structures de données `union-find` et tas min.

Solution:
cf le cours pour développer la solution

Question 2

Est-ce que ce que graphe sans pondération admet un chemin eulérien ? Pourquoi ?

Solution:
Car tous les sommets ont un degré pair sauf 2.