INFO0947: Compléments de Programmation
Récursivité et Listes Circulaires

B. Donnet, S. Liénardy
Université de Liège

1 Problème

Flavius Josephus fut un célèbre soldat et historien Juif ayant vécu de l'an 37 à l'an 100 de notre ère. Lors de la guerre contre les Romains, à la bataille de Jotapata (aux environs de l'an 70 de notre ère), il fut assiégé dans une caverne avec 40 de ses compatriotes par les soldats Vespasiens.

La majorité des 41 Juifs enfermés dans la caverne décidèrent que le suicide était un meilleur sort qu'une vie d'esclavage aux mains des Romains. Josephus, un homme qui aimait la vie par dessus tout, n'approuva pas cette décision. Il fit remarquer à ses compatriotes que le suicide était un acte déshonorant pour un soldat: un soldat qui se respecte doit périr de la main d'un autre soldat.

Pour sauver l'honneur de la majorité des soldats, il suggéra que lui-même et ses 40 compatriotes se disposent en cercle et que chaque soldat tue celui qui se trouve après lui dans le cercle (en tournant dans le sens des aiguilles d'une montre), le commandant de la troupe initiant le processus en exécutant le premier soldat. L'opération devait ensuite se répéter jusqu'à ce qu'il ne reste plus qu'un seul survivant. Ce survivant serait ainsi l'unique soldat à ce suicider, permettant aux autres de sauver l'honneur.

Jospehus choisit soigneusement sa place dans le cercle (si on considère que le commandant occupait la première position dans le cercle, Josephus se placa à la 19ème place). Il faut ainsi le dernier soldat survivant et, s'étant dissimulé, réussit à échapper aux Romains...

2 Récursivité

On peut, bien entendu, obtenir une version généralisée du problème que Josephus a dû résoudre. Ainsi, étant donnés n soldats, placés en cercle aux positions $[0; n-1]$ avec 0 comme position de départ, il faut retirer chaque m-ième soldat jusqu'à ce que tous les soldats (même Josephus pour simplifier les choses) soient retirés.

Dans l'exemple donné par la Fig. 1, nous commençons avec 8 soldats, et nous tuons à chaque tour le troisième soldat sur la gauche (remarquez que lorsqu'il reste au plus trois personnes vivantes, le soldat tuant se compte lui-même dans cette distance de trois soldats).

En réfléchissant et en formalisant les données, on peut obtenir une formulation récursive du problème qui permet d'obtenir, assez élégamment, la position à laquelle doit se tenir Josephus, étant donné n et m, afin de survivre.

3 Liste Chaînée Circulaire

Une façon tout aussi naturelle de résoudre le problème est déjà illustrée par la Fig. 1. Il suffit d'implémenter le problème comme une liste chaînée circulaire, de construire la liste pour une valeur de n donnée et, ensuite, itérer de façon à supprimer les différentes cellules. La cellule restante sera celle où doit se trouver Jospehus afin de survivre.

De façon à simplifier les choses, les différents soldats seront identifiés par une valeur unique $\in \mathbb{N}_0$.

4 Travail à Réaliser

On demande, dans ce projet, de résoudre le problème en suivant le cheminement suivant:

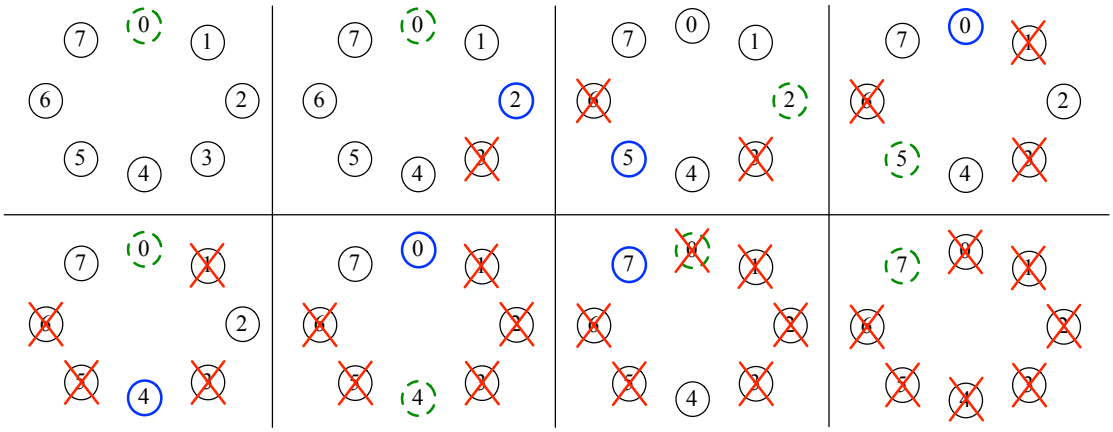

Figure 1: Exemple avec $n = 8$ et $m = 3$. Le cercle en pointillé vert indique le point de départ, le cercle bleu indique le "tueur". La figure doit être lue de gauche à droite (la situation de départ étant illustrée en haut à gauche) et de haut en bas (la situation finale est indiquée en bas à droite)

1. fournir une fomulation récursive du problème, comme demandé dans la Sec. 2;[1]
2. spécifier et construire une fonction récursive, dont le prototype sera `int josephus_rec(int n, int m);`, permettant de résoudre le problème. En outre, vous devrez donner une trace de l'exécution de cette fonction avec les valeurs $n = 8$ et $m = 3$.
3. fournir une structure de données pour une liste chaînée circulaire, comme décrite à la Sec. 3;
4. donner, pour la version avec liste chaînée circulaire une découpe en sous-problèmes respectant le problème de Josephus;
5. spécifier complètement (et le plus formellement possible) chaque sous-problème
6. construire des morceaux de programme correspondant à chaque sous-problème (par l'*approche constructive*) vue au cours;
7. rédiger la fonction finale dont le prototype sera `int josephus_liste(int n, int m);` qui commencera par créer la structure de données circulaire et, ensuite, déterminera quelle position devrait occuper Josephus;
8. une comparaison, en terme de complexité théorique, entre les fonctions `josephus_rec()` et `josephus_liste()`.

5 Aspects Pratiques

Le travail à rendre se compose d'une archive `tar.gz`[2]. Votre archive portera la nom suivant: `josephus-groupeXX.tar.gz`, où `XX` fait référence à l'identifiant de votre groupe.[3]

Une fois décompressée, votre archive devra donner naissance à deux répertoires: `rapport/` (cfr. Sec. 5.1) et `code/` (Sec. 5.2).

Tout non-respect des consignes se verra sanctionné par 2 points en moins dans la note finale de votre projet.

[1] Ne pas hésiter à introduire de nouvelles notations.

[2] Une rapide recherche via Google vous apprendra comment compresser des fichiers dans une archive `tar.gz`.

[3] Pour rappel, il est interdit de changer de groupe entre les différents projets. Idéalement, votre identifiant doit être un nombre composé de deux chiffres (compris entre 01 et 24).

5.1 Rapport

Votre rapport devra être rédigé via l'outil LaTeX en utilisant l'en-tête suivant[4]:

```
\documentclass[a4paper, 11pt]{book}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[francais]{babel}
\usepackage{array}
\usepackage{shortvrb}
\usepackage{listings}
\usepackage[fleqn]{amsmath}
\usepackage{amsfonts}
\usepackage{fullpage}
\usepackage{enumerate}
\usepackage{graphicx}              % import, scale, and rotate graphics
\usepackage{subfigure}             % group figures
\usepackage{alltt}
\usepackage{url}
\usepackage{indentfirst}
\usepackage{eurosym}
\usepackage[french,onelanguage,ruled,vlined]{algorithm2e}

\setcounter{tocdepth}{3}     % Dans la table des matieres
\setcounter{secnumdepth}{3}  % Avec un numero.

\title{INFO0947: Josephus}
\author{Groupe XX: Prénom_Nom, Prénom_Nom}
\date{}

\begin{document}

\end{document}
```

Dans la liste des auteurs, l'élément `XX` correspond à votre numéro de groupe. Vous ferez suivre votre numéro de groupe par les prénoms et noms des différents membres du groupe.

Une fois compilé, votre document LaTeX devra produire un fichier PDF dont le nom sera `josephus-groupeXX.pdf`, où `XX` représente toujours votre numéro de groupe.

Le dossier `rapport/` sera composé des éléments suivants :

- votre rapport au format `.tex` ainsi que toutes les sources utiles à la compilation;
- votre rapport déjà compilé au format `.pdf`

Votre rapport devra contenir tous les éléments nécessaires à la compréhension de votre code source. Soit:

- l'expression récursive du problème de Josephus;
- la spécification formelle des différentes fonctions/procédures;
- votre découpe en sous-problèmes, la description de chaque sous-problème et comment ils s'emboîtent;
- la spécification formelle de chaque sous-problème;

[4]Un template LaTeX vous est fourni sur la page Web du cours

- si un problème nécessite l'introduction d'une boucle, il faudra fournir un dessin de la situation générale et en dériver un invariant formel;
- les différentes étapes ({Pré} *INIT* {Inv}, ...) de la construction de chaque sous-problème et la fonction de terminaison de chaque sous-problème (quand cela s'avère pertinent);
- la trace d'exécution pour l'appel `josephus_rec(8, 3)`;
- une expression mathématique (et toutes les justifications nécessaires) de la complexité théorique des fonctions `josephus_rec()` et `josephus_liste()`.

Soyez clairs et précis dans vos explications. N'hésitez pas à ajouter un schéma si vous le jugez nécessaire. Dans ce cas, expliquez-le : un schéma seul ne suffit pas!

Soignez votre orthographe : au delà de trois fautes, chaque faute vous fera perdre 0,25 points.

5.2 Code Source C

Votre code source devra être rédigé en langage C. Votre code source devra être accompagné d'un Makefile[5] rédigé par vos soins, la commande `make` permettant la génération d'un fichier binaire exécutable appelé `josephus`. Ce fichier exécutable sera situé dans le même répertoire que celui où se trouve le code source. Son exécution doit nous permettre de tester la validité de votre code. La fonction `main` de cet exécutable devra être implémentée dans un fichier appelé `main.c`.

Votre code source sera adéquatement découpé en headers et modules. Les différents sous-problèmes identifiés dans votre rapport devront apparaitre clairement dans votre code. La fonction `josephus_rec()` doit être déclarée dans un header nommé `recursivite.h` et la fonction `josephus_liste()` doit, elle, être déclarée dans un header nommé `liste_circulaire.h`. En outre, ces fonctions doivent pouvoir être compilées et utilisées sans la présence du fichier `main.c`.

Il est **interdit**[6] d'utiliser des fonctions ou procédures de librairies tierces (y compris la librairie standard du C), à l'exception de `<assert.h>`.

6 Agenda

Les archives `tar.gz` doivent être soumises sur la plateforme `http://cicada.run.montefiore.ulg.ac.be` avant le **1 avril** à **11h** (l'heure du serveur faisant foi). Toute soumission postérieure à cette date ne sera pas prise en compte. Comme l'heure de l'horloge du serveur de soumission et celle de votre ordinateur peuvent être légèrement différentes, il est **fortement déconseillé** de soumettre votre projet à 10h59 et 59 secondes : ne prenez pas de risque inutile.

[5] cfr le cours INFO0030, "Chapitre 1: Compilation Multi-Fichiers".

[6] Cette restriction ne s'applique pas au fichier `main.c` où est implémenté l'exécutable de test `josephus`