Université de Paris

IUT de Paris – Rives de Seine

Département Informatique

SAé S1.02 – Comparaison d'approches algorithmiques

1 Présentation du jeu

Le Démineur (Minesweeper) est un jeu vidéo de réflexion dont le but est de localiser des mines cachées dans une grille représentant un champ de mines virtuel, avec pour seule indication le nombre de mines dans les zones adjacentes (cf. WikipédiA).

Le champ de mines du Démineur est représenté par une grille, qui peut avoir différentes formes : deux ou trois dimensions, pavage rectangulaire ou non, etc.

Chaque case de la grille peut soit cacher une mine, soit être vide. Le but du jeu est de découvrir toutes les cases libres sans faire exploser les mines, c'est-à-dire sans démasquer les cases qui les dissimulent.

Lorsque le joueur démasque une case vide comportant au moins une mine dans l'une de ses cases avoisinantes, un chiffre apparaît, indiquant ce nombre de mines. Si en revanche toutes les cases adjacentes sont vides, une case vide est affichée et la même opération est répétée sur ces cases adjacentes, et cela jusqu'à ce que la zone vide soit entièrement délimitée par des chiffres. En comparant les différentes informations récoltées, le joueur peut ainsi progresser dans le déminage du terrain. S'il se trompe et démasque une mine, il a perdu.

Le joueur peut signaler les cases contenant des mines présumées par un drapeau en marquant la case mais ce n'est aucunement obligatoire. Il faut faire attention à ne pas signaler une case saine par un drapeau, car cela peut induire en erreur.

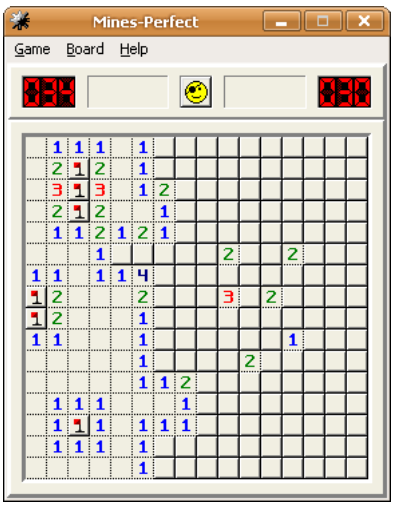

Figure 1: Une situation de jeu

La figure 1 présente une situation de jeu sur une grille 16×16. C'est une image écran d'une version classique du jeu. Les cases entourées d'un trait gras ne sont pas encore démasquées, les cases portant un drapeau ont été marquées par le joueur. Les autres cases ont toutes été démasquées. Ces dernières indiquent le nombre de mines dans les cases adjacentes si celui-ci est non nul et sont vides dans le cas contraire.

Dans notre version, nous nous limiterons à des grilles rectangulaires pavées par des carrés comme dans la figure 1. D'autre part, marquer une case ne contenant pas de mine sera considéré comme une erreur (i.e le joueur perd la partie).

2 Travail à faire

Votre programme doit être capable de réaliser 5 opérations élémentaires. Chacune de ces opérations est désignée par un numéro entier (de 1 à 5). Votre programme doit lire un entier pour savoir quelle opération il doit faire, lire les données à traiter, produire le résultat attendu et se terminer.

Toutes les données en entrée des programmes doivent être lues sur l'entrée standard et toutes les données produites doivent l'être sur la sortie standard.

Les opérations demandées sont les suivantes :

1. Produire un *problème* (voir ci-dessous) à partir du nombre de lignes, de colonnes et de mines.
2. Produire une *grille* (voir ci-dessous) à partir d'un problème et d'un *historique de coups* (voir ci-dessous).
3. Déterminer si la partie est gagnée à partir d'un problème et d'un historique de coups (i.e. toutes les cases ne contenant pas de mine sont démasquées).
4. Déterminer si la partie est perdue à partir d'un problème et d'un historique de coups (i.e. le dernier coup démasque une case contenant une mine ou marque une case n'en contenant pas).
5. Produire un *nouveau coup* à partir d'une grille.

Notez que l'ensemble de ces opérations permet de jouer une partie de DÉMINEUR de bout en bout.

Vous n'avez pas le droit d'utiliser les conteneurs de la bibliothèque standard du C++ (tels que `vector`, `stack`, etc). Par contre, vous pouvez employer les conteneurs vus en TD/TP (tels que `ConteneurTDE`, `Pile`, etc) et/ou les adapter pour développer vos propres composants.

3 Format de saisie et d'affichage

Les cases sont numérotées en commençant à 0 et dans le sens de la lecture – i.e. de gauche à droite et de haut en bas.

Un *problème* est décrit par les nombres de lignes, de colonnes et de mines suivis par la position des mines.

Si les données en entrée sont `1 4 6 5` (1 est le code d'opération pour produire un problème, sa taille doit être de 4 lignes sur 6 colonnes et il doit comporter 5 mines), une sortie valide de votre programme pourra être :

```
4 6 5 1 5 12 7 19
```

Votre programme aura placé aléatoirement les 5 mines sur la grille aux positions 1, 5, 12, 7 et 19.

Un *historique de coups* est décrit par le nombre de coups suivi de la liste des coups (chacun séparé du suivant par au moins un espace). Un coup est désigné par une lettre (`D` pour démasquer ou `M` pour marquer) suivi d'une position désignant une case (non démasquée ou marquée par les coups précédents de l'historique). Dans l'exemple qui suit, l'historique est composé de 3 coups, les cases de position 15 et 0 ont été démasquées alors que la case 5 a été marquée :

```
3 D15 M5 D0
```

La *grille* présentée ci-dessous correspond à celle devant être produite par votre programme lorsque les données suivantes lui sont soumises : `2 4 6 5 1 5 12 7 19 3 D15 M5 D0` (2 est le code opération pour produire une grille, `4 6 5 1 5 12 7 19` est le problème et `3 D15 M5 D0` l'historique de coups). La grille obtenue est la suivante :

```
4 6
 --- --- --- --- --- ---
| 2 | . | 2 |   | 1 | x |
 --- --- --- --- --- ---
| . | . | 2 |   | 1 | 1 |
 --- --- --- --- --- ---
| . | . | 2 |   |   |   |
 --- --- --- --- --- ---
| . | . | 1 |   |   |   |
 --- --- --- --- --- ---
```

Les deux nombres qui précédent la grille proprement dite rappellent le nombre de lignes et de colonnes du problème. Les cases masquées sont repérées un point (`.`) si elles n'ont pas été marquées, et par un `x` dans le cas contraire. Les cases démasquées contiennent le nombre de mines contenues dans les cases adjacentes si celui-ci est non nul, et un espace dans le cas contraire.

Lorsque le dernier coup joué provoque la fin de la partie (que celle-ci soit gagnée ou perdue), toutes les cases contenant une mine sont représentées par un `m`. Par exemple, avec l'entrée `2 4 6 5 1 5 12 7 19 4 D15 M5 D0 D19` (ou `2 4 6 5 1 5 12 7 19 4 D15 M5 D0 M13`), votre programme doit afficher :

```
4 6
 --- --- --- --- --- ---
| 2 | m | 2 |   | 1 | m |
 --- --- --- --- --- ---
| . | m | 2 |   | 1 | 1 |
 --- --- --- --- --- ---
| m | . | 2 |   |   |   |
 --- --- --- --- --- ---
| . | m | 1 |   |   |   |
 --- --- --- --- --- ---
```

Pour l'opération 3 (resp. 4), votre programme doit afficher sur une unique ligne l'un des messages suivant en fonction de l'état du jeu : `game won` ou `game not won` (resp. `game lost` ou `game not lost`),

Enfin, un nouveau coup produit par votre programme (code opération 5) le sera sous la forme classique d'un coup (i.e. `D15` ou `M5`). Ce coup doit être légal dans le sens où la position désigne effectivement une case de la grille n'ayant pas été déjà jouée (démasquée ou marquée). Vous ferez en sorte de démasquer (ou marquer) en priorité les cases que votre programme aura su déduire comme étant libres (ou contenant une mine). Si votre programme n'arrive pas à trouver une telle case, il pourra toujours démasquer une case non encore jouée qu'il choisira aléatoirement.

Vous pouvez faire l'hypothèse que les données fournies en entrée de votre programme ne contiennent aucune erreur.

4 Qui, quoi et quand?

Votre projet doit être fait en binôme. Les groupes de 3 ne seront pas acceptés. Évitez de faire votre projet tout seul (soit vous êtes très fort et des personnes ont besoin de votre soutien, soit vous avez des difficultés et il faut vous faire aider).

Vous devez porter une attention particulière à la rédaction de votre dossier. Sa qualité est déterminante pour l'évaluation de votre travail. Votre dossier doit être un unique document `pdf` dont la composition est la suivante :

- Une page de garde indiquant le nom et **le groupe** des membres du binôme, l'objet du dossier.
- Une table des matières de l'ensemble du dossier.
- Une brève introduction du projet résumant les objectifs (inutile de recopier le sujet).
- Le graphe de dépendance des fichiers sources de vos applications. Tous les composants (qu'ils soient réutilisés ou développés) de vos applications devront figurer sur le graphe (*cf.* Cours 4).
- Les jeux d'essai (fichiers "in" et "out") que vous aurez mis au point.
- Un bilan du projet (les difficultés rencontrées, ce qui est réussi, ce qui peut être amélioré).
- En annexe, le code C++ complet de vos sources (pensez à présenter les fichiers selon un ordre logique).

Nous vous rappelons que le critère principal de notation est la structuration de votre code. Votre rapport doit mettre en avant la qualité de celle-ci.

Vous devez déposer une archive **zip** contenant votre rapport ainsi que l'ensemble des fichiers sources de votre application et vos propres jeux d'essai, le **lundi 3 janvier 2022** sur MOODLE. Votre rapport doit être rendu imprimé au secrétariat à la même date.