Prép'ISIMA

Programmation avancée en C

Approximation

Avec les méthodes des droites de regression ainsi que les ajustements

Clément Payard
Mathieu Laurençot

Encadrant : M. Suzanne ÉLODIE

2021

Table des matières

1 Rappel rapide du jeu 2048

1.1 Résumé du jeu

Le but du jeu est de rassembler des tuiles sur une grille, pour combiner les tuiles de mêmes valeurs et créer ainsi une tuile portant le fameux nombre 2048. Cependant, le jeu s'arrête quand le joueur ne peut plus faire aucun mouvement/fusionner aucune tuile.

1.2 Problèmes potentiels

Ce jeu à en effet quelques spécificités :

1.2.1 Gestion des fusions

Tout d'abord, le jeu peut fusionner plusieurs cases en même temps, sur différentes lignes et colonnes. De plus, sur ces mêmes lignes et colonnes, comme par exemple avec cette exemple :

2	2	4	4
4	8	2	2
0	0	0	0
0	0	0	0

Si le joueur fusionne à gauche, nous devrons fusionner les deux ET les 4 présents sur la première ligne.

De plus, lorsque l'on doit fusionner plusieurs cases, nous devons mettre les résultats de la fusion "collés" dans la direction voulue. Nous devons donc gérer ce cas en plusieurs étapes.

1.2.2 Différentes taille

Nous souhaitons faire plusieurs tailles de 2048, nous voulons donc avoir la possibilité de changer la taille de la grille.

1.2.3 Gestion des sauvegardes

Nous voulons que notre joueur puisse faire une pause. Nous nous imposons donc de faire un système de sauvegarde rapide et efficace.

2 Présentation du programme console

Nous ne mettrons pas l'intégralité du code des différentes résolutions, mais il est consultable dans les fichiers joints.

2.1 Introduction à la structure Terrain

Nous avons choisis d'utiliser une structure pour notre Terrain pour pouvoir manipuler notre grille aisément.

```
  typedef struct Terrain{
  int **tab;
  int max;
  int tailleX;
  int tailleY;
  int vide;
  int score;
}ter;
```

Elle est composé de divers éléments, mais seulement d'entiers :

1. Un tableau en 2 dimensions, qui serra la grille en elle même.
2. max, qui sera l'entier présent dans la grille maximal. Permet de gérer le cas de "victoire" si l'on fait 2048.
3. tailleX, qui sera le nombre de colonne
4. tailleY, qui sera le nombre de ligne
5. vide, qui sera le nombre de case vide du tableau
6. score, qui sera le score du Terrain en cours

Toutes ces éléments sont présents pour simplifier le nombre de paramètre pris par les divers fonctions suivantes

2.2 Présentation des différentes fonctions

2.2.1 Initivei

2.2.2 Affiche ter

2.2.3 SetRandomCase

2.2.4 writesaveFile

2.2.5 ReadEnCoursSave

$\text{Partie}_{\text{encours}}$ paramètre : time pour temps entre séquence de jeu lors du replay

N = type de la partie (), puis taille X et Y du ter n = rien puis emplacement de la case, en ne prenant pas en compte les case vides

Replay : même logique

Pour rejouer : re fait le replay

usleep((int)(1000000*time)); // on effectue le deplacement deja effectue par le joueur CaseMouve(&T, depNum);

stocker le nombre de case vide, et

un fichier moddifiable $\text{Param}_{\text{Name}}$

High score Nb de victoire Nb de défaite les nouvelles touches*4 nb de replay (pour le nom du fichier des replays au fur et à mesure)

2.3 Présentation de la méthode des ajustements

Le code pour la méthode des ajustements est disponible ici.

Pour la résolution par la méthode des ajustement, et après avoir résolu l'équation du type $Y = AX + B$, nous commençons tout d'abord, pour les deux méthodes, de transformer notre liste en tableau, ainsi que de créer le polynôme de degrés 1. Puis, grâce à la fonction *lnListe*, nous créons un second tableau qui contiendra les points avec application de la fonction logarithme (car on a posé $Y = ln(y)$, $X = ln(x)$, $A = b$ et $B = ln(a)$ pour la question 3 et $Y = ln(y)$, $X = x$, $A = d * ln(e)$ et $B = ln(c)$ pour la question 4).

```
polynome *P = creePolynome(1);
int n = ListLenght(listedepoint);
float **Tnormal = ListeToTabsPoints(listedepoint);
float **Tln = lnListe(Tnormal, n);
```

2.3.1 Question 3 (Exponentiel)

Nous initialisons les variables *a* et *b*, qui seront des variables temporaires.

Puis, nous calculons les moyennes de X et de Yln, pour faciliter la lecture du code et rendre le code plus facilement compréhensible.

```
float moyenneX = moyX(Tnormal, n);
float moyenneYln = moyY(Tln, n);
```

Subséquemment, nous pouvons effectuer le calcul des ajustement de la puissance appliquant la formule du cours, ainsi qu'en utilisant la formule posé précédemment.

Le premier calcul sera pour calculer l'exposant présent devant x, donc d.

```
a = (moyXY2tab(Tnormal, Tln, n) - moyX(Tnormal, n) * moyY(Tln, n)) /
      (moyXcar(Tnormal, n) - carmoyX(Tnormal, n));
```

Et le second sera pour la coefficient présent devant x.

```
float temp = moyenneYln - (a * moyenneX);
  b = exp(1) * exp(temp);
```

Enfin, nous pouvons retourner le polynôme P.

2.3.2 Question 4 (Puissance)

Nous initialisons les variables *a* et *b*, qui seront des variables temporaires.

Subséquemment, nous pouvons effectuer le calcul des ajustement de la puissance appliquant la formule du cours, ainsi qu'en utilisant la formule posé précédemment.

Le premier calcul sera pour calculer l'exposant, donc b.

```
b = (moyXY(Tln, n) - moyX(Tln, n) * moyY(Tln, n)) /
    (moyXcar(Tln, n) - carmoyX(Tln, n));
```

Et le second sera pour la coefficient présent devant x.

```
float temp = moyenneYnl - (b * moyenneXnl);
  a = exp(temp);
```

Enfin, nous pouvons retourner le polynôme P.

2.4 Gestion des sauvegardes

2.4.1 Présentation du fichier de sauvegarde

Un fichier de sauvegarde est présenter de cette manière : Unelettre Unchiffre Unautrechiffre Unelettre Unchiffre Unautrechiffre Unelettre Unchiffre Unautrechiffre Unelettre Unchiffre Unautrechiffre [...]

Un exemple concret :

```
N 4 2
n 2 3 n 2 4 h 2 3 b 2 2 g 2 4 g
```

Explication :

2.4.2 La sauvegarde en elle-même

La gestion des sauvegardes est particulière. En effet, lorsque le joueur joue un coup, nous enregistr

2.4.3 Récupération de la sauvegarde

3 SDL

Le code qui nous permet de gérer la fenêtre SDL est disponible ici.

3.1 Début

Pour démarrer SDL, nous devons initialiser de nombreuses variables, comme par exemple :

— La variable *Stape*, qui nous permet de fermer SDL si elle est égale à 0,
— *size*, qui va nous permettre de gérer la taille de l'écran,
— des variables permettant de garder un nombre d'image par seconde (fps) constant et agréable
— des variables permettant de détecter où le curseur de la souris se trouve sur l'écran
— les varibles permettant de dessiner le graphique
— etc.

3.2 Affichage et fonctionnalité

3.2.1 Affichage

Pour effectuer l'affichage d'une fenêtre SDL, nous devons passer par une boucle *while*.

Puis, nous distinguerons trois cas grâce à un *if* (et *else if*).

1. Dans le premier cas, SDL dessinera l'écran, s'il n'a pas été dessiné depuis un certain temps
2. Sinon, nous vérifierons également si les courbes sont en adéquation avec les polynômes. Si ce n'est pas le cas, nous entrons alors dans le *else if* qui va nous permettre d'écraser l'image précédente. Enfin,
3. si nous passons les deux conditions précédentes, nous devons **absolument** endormir le Central Processing Unit (CPU). Cela nous permet de ne pas utiliser tout le processeur de l'ordinateur.

Puis, nous avons aussi un cas de débogage. En effet, si l'on n'est pas entré dans le while depuis une seconde ou plus, il peut y avoir un problème. On recommence alors une seconde "propre", en mettant certaines variables à 0.

3.2.2 Fonctionnalités :

Divers fonctionnalités sont présentes : Vous pouvez afficher la fenêtre grâce à la touche "g". Vous pouvez désormais voir la liste des points, les courbes représentant les différentes méthodes d'approximation, ainsi que la liste de points à droite.

De plus, si jamais vous voulez rajouter des points à la liste, cette fonctionnalité est disponible grâce au bouton gauche de la souris. Le bouton droit aura pour effet de supprimer le point sélectionné.

Le curseur aura alors une position (x et y) qui sera automatiquement ajouté dans la liste des points. Les courbes ainsi que les polynômes vont s'adapter automatiquement !

D'autre part, vous pouvez zoomer et dézoomer grâce à la molette de la souris.

Enfin, vous pouvez activer ou désactiver les différentes courbes des fonctions en appuyant sur leur nom.

3.3 Fin de SDL

La fonction *end-sdl* nous permet de fermet la fenêtre SDL proprement, ainsi que faire les opérations nécessaires pour vider la mémoire qui a besoin d'être libéré.

4 TODO Présentation des nouvelles fonctionnalités

Les différents résultats sont disponibles en faisant les test grâces aux fonctionnalités implémentées dans le programme.

4.1 Série S

Liste de points sans difficulté particulière, mais pas dans le bon ordre.

Résultats : On peut voir que les résultats sont les bons : l'approximation à l'air (graphiquement) efficace.

4.2 Les trois séries :

Suites de points normaux. Difficultés potentielles : aucune, sauf pour le dernier qui ne pourra sûrement pas être calculé (car ce n'est pas une suite de points avec différents x)

Résultats :

1. Les approximations sont les bonnes, aucune difficulté.
2. Les approximations sont bonnes, aucune difficulté également.
3. Ne peut pas se calculer : en effet, les points sont sur le même X. Les approximations sont donc incalculables (et infaisable).

4.3 Dépenses mensuelles et revenus :

Pour ces données, il y a un grand nombre de données, ainsi qu'un axe des x qui commence avec de "grandes valeurs".

Difficultés potentielles : précisions dû aux résultats importants obtenus.

Résultats :Les polynômes ont soit des coefficients très importants, soit des coefficients presque négligeable. En revanche, on peut constater qu'il y a des approximations de calcul dans les deux méthodes. En effet, les deux polynômes finaux ne sont pas exactement les mêmes, même si ils sont tous les deux du même ordre de grandeur.

4.4 Série chronologique avec accroissement exponentiel

L'exponentiel de la forme $y = ce^{dx}$ est parfaitement sur les données, ce qui prouve que cette approximation est adaptée.

4.5 Vérification de la loi d Pareto

L'exponentiel de la forme $y = ax^b$ est parfaitement sur les données, ce qui prouve la vérification de cette loi.

4.6 Commentaire global

Pour ses différents jeux d'essais, on peut constater plusieurs choses :

1. La méthode des droites de régression et deux droites de régression ne donnent pas la même chose
   Et c'est normal. En effet, avec l'un, nous appliquons la formule sur l'ensemble des points. Tandis que sur la deuxième, nous séparons le tableau en deux, puis effectuons une moyenne des deux droites obtenues. Leur efficacité dépend donc des points donnés, et est donc "aléatoire".
2. Les méthodes d'ajustements sont plus ou moins précises en fonction des cas (comme on peut le constater dans le programme)
3. Évaluation des coûts : Pour les droites d'approximation, la complexité est de l'ordre (o)n^2, elle est également de (o)n^2 pour les méthodes des questions 3 et 4.

5 Conclusion

Maintenant, on vous souhaite bonne chance.