Quarantine

Christian Kauth

Hard Task

Hallucinations now reach their apex and pictures start changing over time! Write a function
`void load_picture(int N)` that receives the size N of the initial all-0 picture. Heidi will query you about the status of specific pixels via calls to `bool query(int r, int c)`, but she will interlace those queries with occasional updates in her picture by calling
`void toggle(int r1, int c1, int r2, int c2)`, which indicates that all pixels in the rectangle $[r_1..r_2] \times [c_1..c_2]$ toggled, i.e. every 0 switches to 1 and every 1 switches to 0. It is guaranteed that $0 \leq r_1 \leq r_2 \leq N-1$ and $0 \leq c_1 \leq c_2 \leq N-1$. There will be 10 pictures (calls to `load_picture`) and a cumulative total of 10^6 toggles or queries to be answered within seconds.

The sequence `load_picture(100)`, `query(16,3)`, `toggle(0,0,100,100)`, `query(20,13)`, shall return `false` and `true`.

Tâche Difficile

Les hallucinations sont à un état critique et les images commencent à changer dans le temps ! Ecrivez une fonction
`void load_picture(int N)` qui reçoit la taille N des images initiales (dont tous les pixels sont à zéro). Heidi vous demandera l'état de pixels particuliers à travers la fonction `bool query(int r, int c)`, mais elle entrelacera ces questions avec des mises à jour occasionnelles dans son image en appelant
`void toggle(int r1, int c1, int r2, int c2)`, qui indique que tous les pixels dans le rectangle $[r_1..r_2] \times [c_1..c_2]$ ont changé de valeur, c'est à dire que tous les 0 passent à 1 et tous les 1 passent à 0. Il est garanti que $0 \leq r_1 \leq r_2 \leq N-1$ et $0 \leq c_1 \leq c_2 \leq N-1$. Il y aura 10 images (appels à `load_picture`) et le total (cumulé) sera de 10^6 changements de valeurs ou questions auxquelles il faudra répondre en quelques secondes.

La séquence `load_picture(100)`, `query(16,3)`, `toggle(0,0,100,100)`, `query(20,13)`, doit retourner `false` et `true`.

Schwierige Aufgabe

Heidis Halluzinationen erreichen ihren Höhepunkt. Die Bilder fangen an, sich zu verändern! Schreibe eine Funktion `void load_picture(int N)`. Diese erhält die Grösse N eines Bildes, das zu Beginn ganz mit 0 gefüllt ist. Heidi wird danach den Status einzelner Pixel mittels `bool query(int r, int c)` abfragen. Sie wird zwischendurch das Bild anpassen, indem sie
`void toggle(int r1, int c1, int r2, int c2)` aufruft. Diese Funktion schaltet alle Pixel im Rechteck $[r_1..r_2] \times [c_1..c_2]$ um, d.h. jede 0 wird zu 1, und umgekehrt. Heidi garantiert, dass $0 \leq r_1 \leq r_2 \leq N-1$ und $0 \leq c_1 \leq c_2 \leq N-1$. Die Tests bestehen aus 10 Bildern (Aufrufen von `load_picture`) und insgesamt 10^6 Anfragen und Aktualisierungen, welche euer Programm innert Sekunden bewältigen muss.

Für die Befehle `load_picture(100)`, `query(16,3)`, `toggle(0,0,100,100)`, `query(20,13)` soll euer Programm `false` und `true` zurückgeben.