Ciencias de la Computación
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

Tarea 1: Pilas de Arena Abelianas

Alumno: José Miguel Pacheco

Fecha: 6 de Abril de 2018

Curso: CC3001-Algoritmos y Estructuras de Datos

Profesor: Patricio Poblete

Auxiliares: Gabriel Flores
Sven Reisenegger

Ayudantes: Fabián Mosso
Gabriel Chandia
Matías Risco
Matías Ramírez

Introducción

En el presente informe se muestra un algoritmo que permite estudiar derrumbes de arena de manera muy simplificada, es decir, dada una cantidad de granos de arena puestos unos sobre otros y al haber más de cierta cantidad comienzan a ocurrir derrumbes de arena hasta que se estabiliza el sistema. La solución aplicada para este problema consiste en analizar si la primera pila de arena cumple con la condición para un derrumbe y si es así simularlo, luego analiza cada nueva pila de arena creada y repite lo mismo hasta que ninguna cumpla con la condición de derrumbe lo que significa que todo el sistema está en equilibrio.

Análisis y soluciones del problema

El problema consiste en simular un derrumbe de arena, para lo cual se dice que los granos de arena se colocan en una superficie plana dividida en celdas cuadradas, y además si existe una pila de arena de 4 o más granos en una celda se reparte un grano hacia las cuatro celdas vecinas, y así el proceso se repite hasta que todas las celdas tengan 3 o menos granos de arena lo que significa que todo el sistema está en equilibrio y no ocurrirán más derrumbes.

La primera estrategia que se implementó para este problema fue la siguiente, se creó la clase PilaArena, la cual al ser llamada pregunta al usuario la cantidad de granos de arena ("n") con la cual quiere simular el derrumbe, al recibir el número se crea una matriz de $(\sqrt{n}+1)\times(\sqrt{n}+1)$, tiene esos valores ya que se coloca en el peor de los casos que seria que cada grano de arena quede en una celda y se le suma uno ya que para los números menores que 4 la matriz sería muy pequeña. Una vez creada la matriz, se colocan todos los granos de arena lo más al centro de la matriz que se pueda, es decir en la posición $\left[(\sqrt{n}+1)/2\right]\left[(\sqrt{n}+1)/2\right]$, luego lo que hace la clase, es que gracias a un "while" y dos "for" empieza a recorrer cada celda de la matriz reiteradas veces y mientras recorre la matriz analiza si en alguna celda por la que pasa hay 4 o más granos de arena, si es que los hay, gracias a un "while" comienza a repartir equitativamente 4 granos de arena, 1 a cada celda contigua hasta que en la celda donde se encuentra queden menos de 4 granos, además, por cada 4 granos de arena repartidos se aumenta un "contador" que permitirá saber cuántos derrumbes se hicieron en total, luego continua con la siguiente celda repitiendo el mismo proceso, si es que en una celda no hay más de 4 granos de arena pasa a la siguiente celda pero antes aumenta un contador "k" previamente definido el cual se reinicia a 0 cada vez que comienza a recorrer desde

el principio la matriz y que además permite terminar de recorrer la matriz cuando alcanza el valor de $(\sqrt{n}+1)\times(\sqrt{n}+1)$, ya que eso significaría que paso por todas las celdas de la matriz y que todas tenían menos de 4 granos de arena por lo que el sistema estaría en equilibrio. Una vez que el sistema está en equilibrio se crea una ventana que permitirá mostrar la matriz resultante luego de todo el proceso, además se imprimirá en pantalla el "contador" que permitirá saber la cantidad de derrumbes que se realizaron en todo el proceso.

La segunda estrategia implementada es en gran parte igual a la primera estrategia, con la diferencia que al momento de que en una celda hayan más de 4 granos de arena, reparte a las celdas contiguas el máximo múltiplo de 4 posible, es decir, (la cantidad de granos de arena en la celda) /4 y en la celda que se encontraba queda el resto de la división por 4, es decir, (la cantidad de granos de arena en la celda)%4 y todo esto se hace en una iteración y es solo un derrumbe, todo lo demás es prácticamente igual a lo explicado anteriormente.

```
while (k<(largo)*(largo)) {
k=0;
for (int i=0;i<largo;i++){
    for (int j=0;j<largo;j++){
        if (matriz[i][j]<4) {
            k++;
        }
        while (matriz[i][j]>=4){
            matriz[i][j+1]=(matriz[i][j+1])+1;
            matriz[i][j-1]=(matriz[i][j-1])+1;
            matriz[i+1][j]=(matriz[i+1][j])+1;
            matriz[i-1][j]=(matriz[i-1][j])+1;
            matriz[i][j]=(matriz[i][j])-4;
```

Figura 1: Parte central del código de la parte 1

```
while (k<(largo)*(largo)) {
k=0;
for (int i=0;i<largo;i++){
    for (int j=0;j<largo;j++){
        if (matriz[i][j]<4) {
            k++;
        }
        if (matriz[i][j]>=4){
            matriz[i][j+1]=(matriz[i][j+1])+(matriz[i][j])/4;
            matriz[i][j-1]=(matriz[i][j-1])+(matriz[i][j])/4;
            matriz[i+1][j]=(matriz[i+1][j])+(matriz[i][j])/4;
            matriz[i-1][j]=(matriz[i-1][j])+(matriz[i][j])/4;
            matriz[i][j]=(matriz[i][j])%4;
            contador++;
```

Figura 2: Parte central del código de la parte 2

Resultados

A continuación, se muestra el resultado de aplicar el programa para dos valores diferentes de granos de arena, la primera imagen con 128 granos de arena y la segunda imagen con 500000 granos de arena.

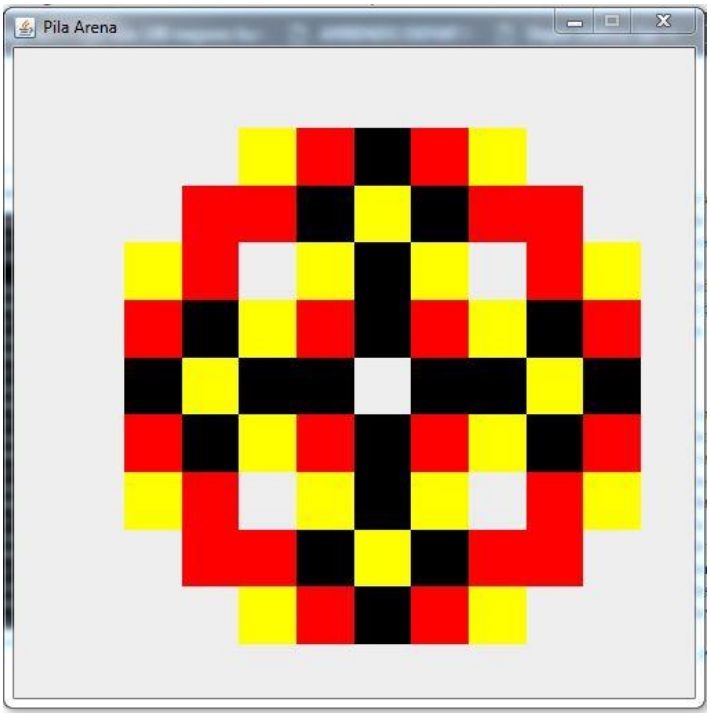

Figura 3: Simulación de derrumbe con 128 granos de arena

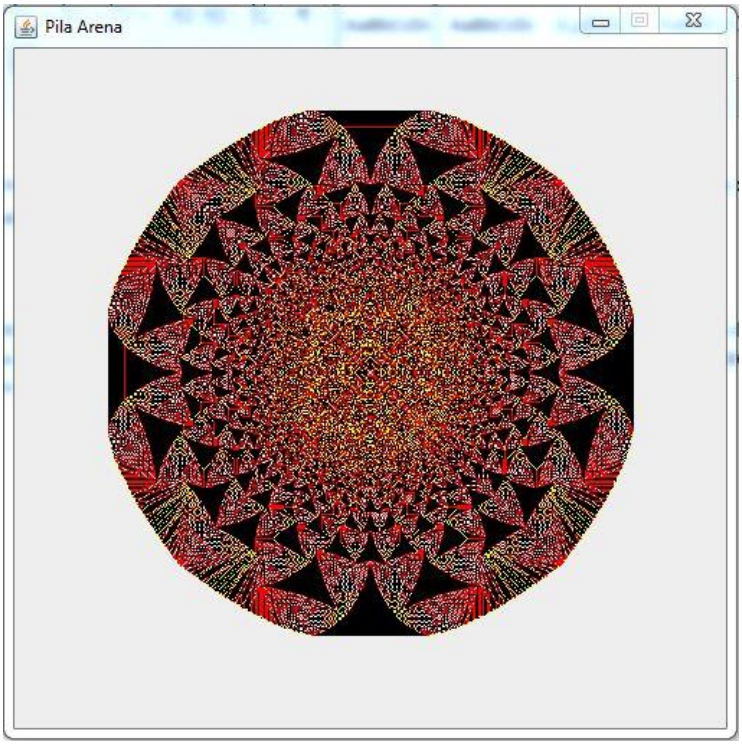

Figura 4: Simulación de derrumbe con 500000 granos de arena

En la siguiente tabla se compara el número de iteraciones realizadas por el código de la parte 1 de la tarea en comparación con el código de la parte 2.

Granos de Arena	Iteraciones Parte 1	Iteraciones Parte 2
128	342	227
1000	18226	11724
10000	1830917	1196817
50000	45202532	29393221
100000	178641503	115491346
500000	4386785739	2813716317

Tabla 1: Comapracion de iteraciones entre los codigos de la parte 1 y la parte 2 de la tarea

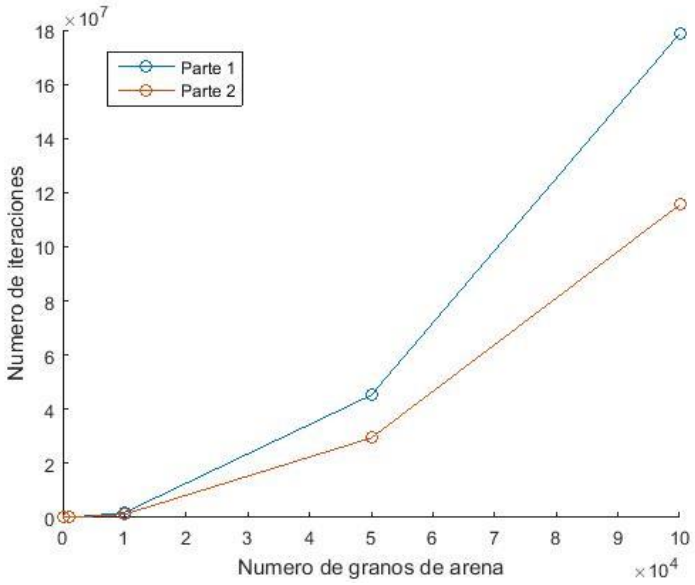

Figura 5: Grafico comparativo entre el algoritmo de la parte 1 y el de la parte 2

Conclusiones

Observando los resultados mostrados en la tabla 1 se puede concluir que la optimización realizada en la parte 2 disminuye considerablemente el número de derrumbes que ocurren hasta que el sistema esté en equilibrio, en otras palabras, el programa de la parte 2 es aproximadamente un 35% más eficiente en la cantidad de derrumbes que el programa de la parte 1, por lo que esa optimización si valió la pena aunque no es la mejor optimización que se puede hacer, ya que se podrían hacer cálculos para que la dimensión de la matriz sea justo lo necesario, lo que permitiría que el programa no recorra partes de la matriz que nunca va a ocupar, lo que optimizaría más el tiempo de ejecución