Estructuras de Datos y Algoritmos – IIC2133

AY – Parte I: Dinosaurios

Outline

- Problema de Dinosaurios
- Modelación
- Lógica algoritmo
- Pseudo-código

- Problema de Dinosaurios
- Modelación
- Lógica algoritmo
- Pseudo-código

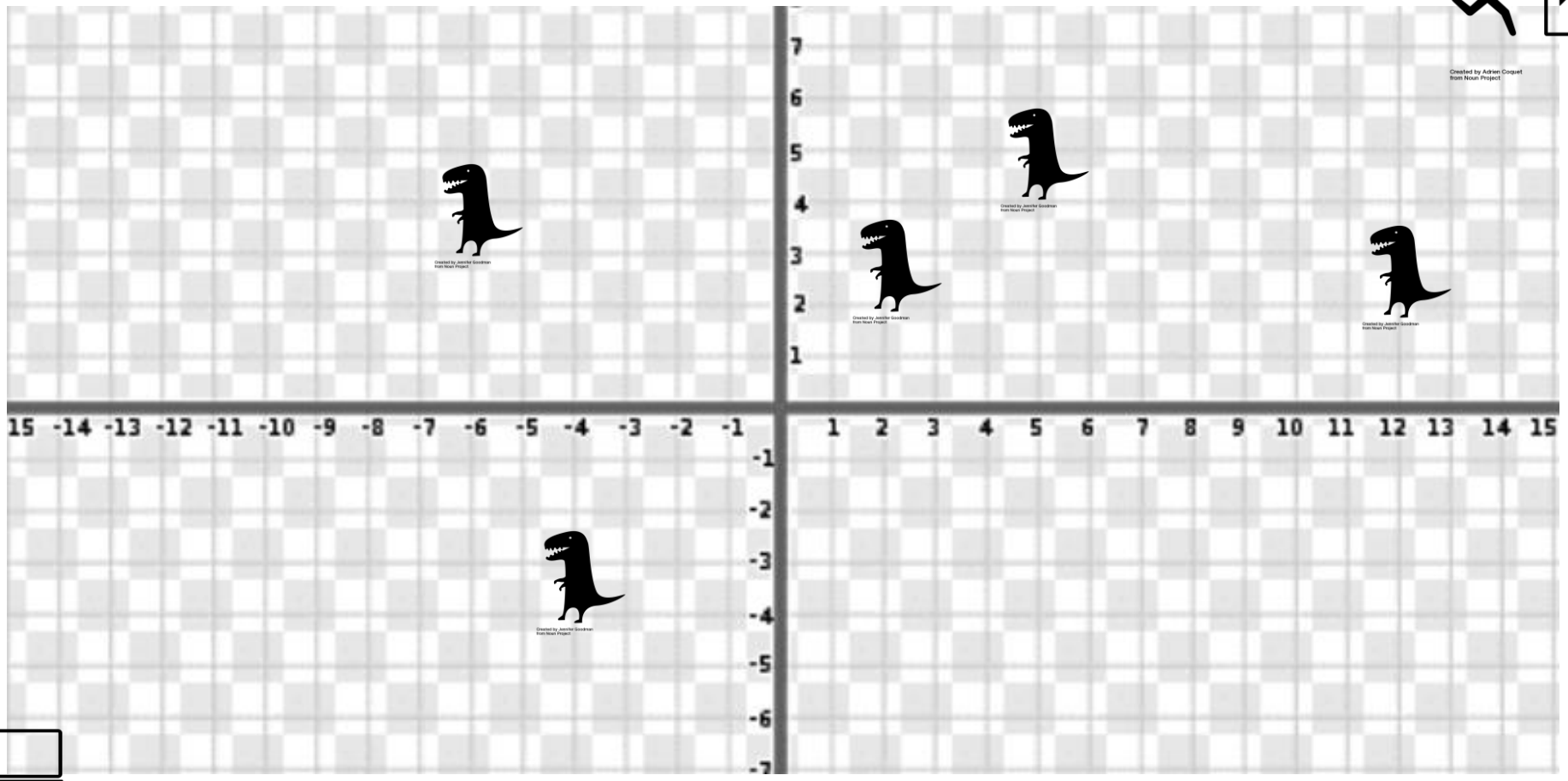
Problema de Dinosaurios

Usted es una científica/o en un laboratorio en una isla paradisiaca. Usted ha estado trabajando en una investigación de cómo traer a la vida a los ya extintos dinosaurios. Todo iba bien hasta que una noche, mientras usted trabajaba intensamente, los dinosaurios se escaparon de sus jaulas y entraron a su enorme laboratorio. Como investigador, y ser humano, usted busca sobrevivir el accidente de los dinosaurios.

Para esto, usted necesita escapar a salvo del laboratorio. Gracias a chips identificadores en cada dinosaurio, usted puede saber desde su estación de trabajo en la esquina del laboratorio, cuál es la posición de cada dinosaurio al iniciar el escape. Lamentablemente, no sabe con seguridad el radio D (desde sus posiciones inciales) con que los dinosaurios podrán detectarlo en su escape.

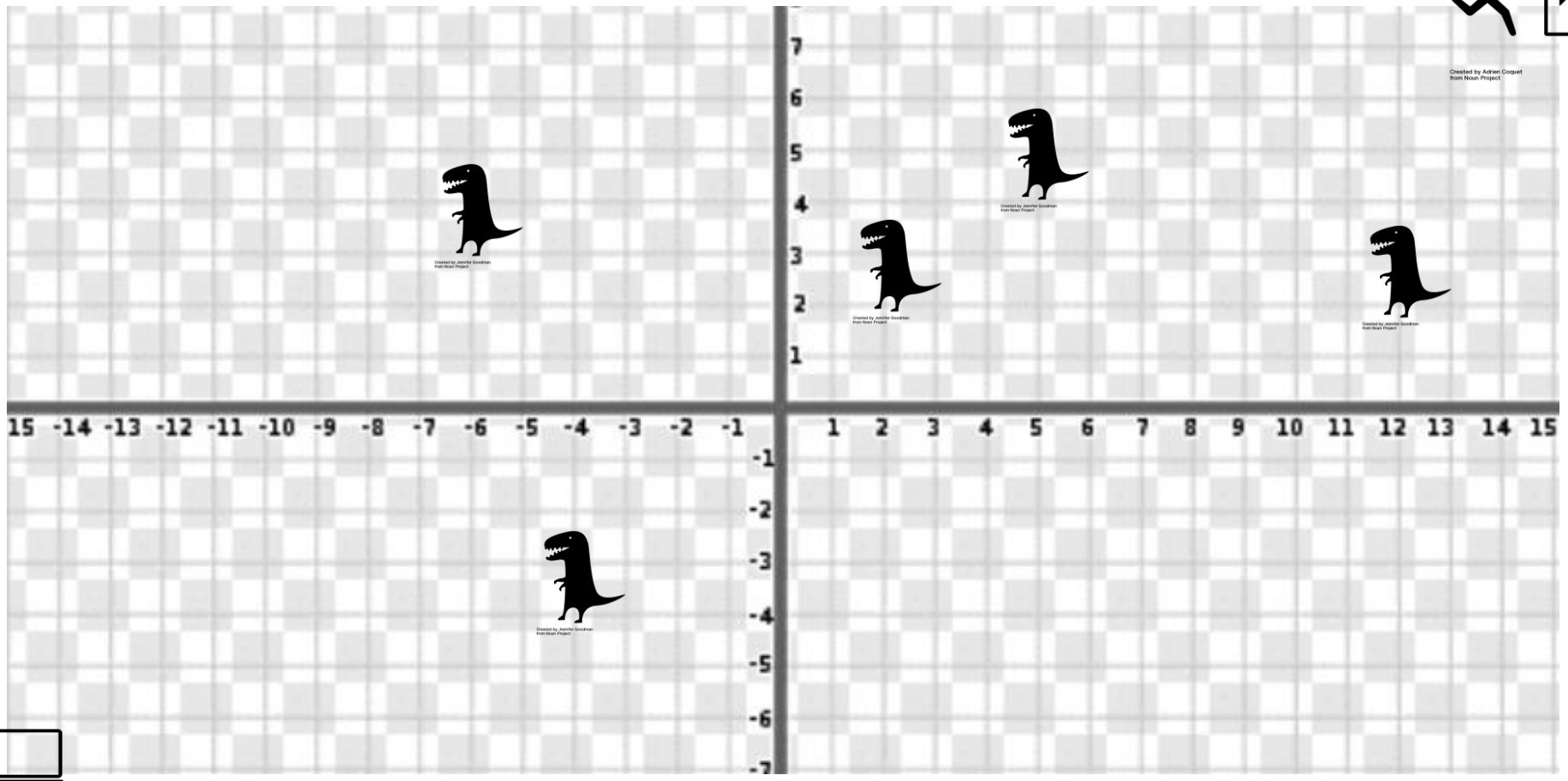
Usted debe diseñar un algoritmo que, dado un estado del mapa de dinosaurios, le indique el rango D máximo que podrían tener los dinosaurios, tal que exista alguna ruta de escape segura. De esta forma, cuando se sienta listo para escapar podrá revisar si es una buena idea (dado el rango D), o si espera a otro momento.

Problema de Dinosaurios



- Problema de Dinosaurios
- Modelación
- Lógica algoritmo
- Pseudo-código

Modelación


15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
7 6 5 4 3 2 1
-1 -2 -3 -4 -5 -6 -7

Modelación


Created by Jennifer Goodman
from Noun Project


Created by Jennifer Goodman
from Noun Project


Created by Jennifer Goodman
from Noun Project


Created by Jennifer Goodman
from Noun Project


Created by Jennifer Goodman
from Noun Project

Modelación

Grafo:

- Vértices: Dinosaurios (D) y 4 murallas (M)
- Aristas: Distancias D-D y D-M*

- Problema de Dinosaurios
- Modelación
- Lógica algoritmo
- Implementación

Lógica del algoritmo

Lógica del algoritmo

… Dado un D

Lógica del algoritmo

… Dado un D

- 4 condiciones:
  - M(arriba) - … - M(abajo)
  - M(arriba) - … - M(derecha)
  - M(izquierda) - … - M(derecha)
  - M(izquierda) - … - M(abajo)

Lógica del algoritmo

… Dado un D

- 4 condiciones:
  - M(arriba) - … - M(abajo)
  - M(arriba) - … - M(derecha)
  - M(izquierda) - … - M(derecha)
  - M(izquierda) - … - M(abajo)
- Necesitamos encontrar los conjuntos de dinosaurios (y murallas) conectados…

Lógica del algoritmo

… Dado un D

- 4 condiciones:
  - M(arriba) - … - M(abajo)
  - M(arriba) - … - M(derecha)
  - M(izquierda) - … - M(derecha)
  - M(izquierda) - … - M(abajo)
- Necesitamos encontrar los conjuntos de dinosaurios (y murallas) conectados… necesitamos saber si están en conjuntos disjuntos

Conjuntos disjuntos

Lógica del algoritmo

… Para encontrar D

Lógica del algoritmo

… Para encontrar D

- Equivalencia con MST

Lógica del algoritmo

… Para encontrar D

- Equivalencia con MST
- Pero no buscamos completar árbol, sólo conectar una pareja de murallas entre las 4 opciones (sub-árbol)

Conjuntos disjuntos

Lógica del algoritmo

… Para encontrar D

- Equivalencia con MST
- Pero no buscamos completar árbol, sólo conectar una pareja de murallas entre las 4 opciones
- Kruskal

Lógica del algoritmo

… Para encontrar D

- Equivalencia con MST
- Pero no buscamos completar árbol, sólo conectar una pareja de murallas entre las 4 opciones
- Kruskal… modificado

- Problema de Dinosaurios
- Modelación
- Lógica algoritmo
- Pseudo-código

Pseudo-código

$kruskal\big(G(V,E)\big)$:

    Ordenar E por costo, de menor a mayor
    $foreach\ v \in V$: $make\ set(v)$
    $T \leftarrow \emptyset$
    $foreach\ (u,v) \in E$:
        $if\ find\ set(u) \neq find\ set(v)$:
            $T \leftarrow T \cup \{(u,v)\}$
            $union(u,v)$
    $return\ T$

Pseudo-código

¿Complejidad?

Estructuras de Datos y Algoritmos – IIC2133

AY – Parte II: Snek

Problema del Snek

Matrix Snek

Dada una matriz A de m x n, escribe un algoritmo capaz de encontrar la `serpiente` de largo mayor. Se define una `serpiente` como una secuencia de números adyacentes con a lo más 1 de diferencia entre elementos consecutivos. La serpiente sólo se puede mover hacia abajo y a la derecha.

$$A = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{1} & 2 \\ 7 & 8 & \mathbf{2} & 5 \\ 6 & 6 & \mathbf{3} & \mathbf{2} \\ 9 & 3 & 2 & \mathbf{3} \end{bmatrix}$$

Outline

- Problema del Snek
- Decisiones estratégicas
- Lógica algoritmo
- Implementación

- Problema del Snek
- Decisiones estratégicas
- Lógica algoritmo
- Implementación

Problema del Snek

Matrix Snek

Dada una matriz A de m x n, escribe un algoritmo capaz de encontrar la `serpiente` de largo mayor. Se define una `serpiente` como una secuencia de números adyacentes con a lo más 1 de diferencia entre elementos consecutivos. La serpiente sólo se puede mover hacia abajo y a la derecha.

$$A = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{1} & 2 \\ 7 & 8 & \mathbf{2} & 5 \\ 6 & 6 & \mathbf{3} & \mathbf{2} \\ 9 & 3 & 2 & \mathbf{3} \end{bmatrix}$$

- Problema del Snek
- Decisiones estratégicas
- Lógica algoritmo
- Implementación

Decisiones estratégicas

¿Qué estrategia algorítmica?

Decisiones estratégicas

¿Qué estrategia algorítmica?

- Programación Dinámica

Decisiones estratégicas

¿Qué estrategia algorítmica?

- Programación Dinámica

¿Top-Down o Bottom-Up?

Decisiones estratégicas

¿Qué estrategia algorítmica?

- Programación Dinámica

¿Top-Down o Bottom-Up?

- Bottom-Up

Decisiones estratégicas

¿Qué estrategia algorítmica?

- Programación Dinámica

¿Top-Down o Bottom-Up?

- Bottom-Up

¿Recursos adicionales?

Decisiones estratégicas

¿Qué estrategia algorítmica?

- Programación Dinámica

¿Top-Down o Bottom-Up?

- Bottom-Up

¿Recursos adicionales?

- Array 2D (sub-soluciones), Stack para Snek final

- Problema del Snek
- Decisiones estratégicas
- Lógica algoritmo
- Implementación

Lógica del algoritmo

Recorrer cada celda de la matriz una sola vez desde fila superior a inferior y columna izquierda a derecha, para siempre haber visitado los predecesores de una celda c antes de visitarla, y evaluar el mayor largo posible hasta ella.

Lógica del algoritmo

1. Primero inicializamos las estructuras y variables.

Lógica del algoritmo

1. Primero inicializamos las estructuras y variables.
2. Como ya se dijo antes, recorremos cada elemento de A solamente una vez.

Lógica del algoritmo

1. Primero inicializamos las estructuras y variables.
2. Como ya se dijo antes, recorremos cada elemento de A solamente una vez.
3. Luego, se construye el Snek iterando sobre su largo final, partiendo en la celda final.

Lógica del algoritmo

1. Primero inicializamos las estructuras y variables.
2. Como ya se dijo antes, recorremos cada elemento de A solamente una vez.
3. Luego, se construye el Snek iterando sobre su largo final, partiendo en la celda final.
4. Finalmente, se puede retornar el Snek

Lógica del algoritmo

1. Primero inicializamos las estructuras y variables.
2. Como ya se dijo antes, recorremos cada elemento de A solamente una vez.
3. Luego, se construye el Snek iterando sobre su largo final, partiendo en la celda final.
4. Finalmente, se puede retornar el Snek

2. Recorrer cada elemento

a) Para cada elemento se revisan sus precursores. Estas son la celda arriba y la celda a la izquierda del elemento.

2. Recorrer cada elemento

a) Para cada elemento se revisan sus precursores. Estas son la celda arriba y la celda a la izquierda del elemento.

b) En estas se revisa que cumpla que las celdas estén conectadas (diferencia nivel>1).

2. Recorrer cada elemento

a) Para cada elemento se revisan sus precursores. Estas son la celda arriba y la celda a la izquierda del elemento.

b) En estas se revisa que cumpla que las celdas estén conectadas (diferencia nivel>1).

c) Si eso se cumple se revisa si se actualiza el valor guardado en B. Este representa el largo mayor acumulado hasta el elemento revisado. Si el del antecesor es igual o mayor al de la casilla que se está revisando, el valor de la casilla aumenta y toma como nuevo valor el del antecesor + 1. En caso contrario se queda igual

2. Recorrer cada elemento

a) Para cada elemento se revisan sus precursores. Estas son la celda arriba y la celda a la izquierda del elemento.

b) En estas se revisa que cumpla que las celdas estén conectadas (diferencia nivel>1).

c) Si eso se cumple se revisa si se actualiza el valor guardado en B. Este representa el largo mayor acumulado hasta el elemento revisado. Si el del antecesor es igual o mayor al de la casilla que se está revisando, el valor de la casilla aumenta y toma como nuevo valor el del antecesor + 1. En caso contrario se queda igual

d) Después, en caso de ser mayor, se actualiza el valor del mejor largo con respecto del valor de B de la casilla revisada.

Lógica del algoritmo

1. Primero inicializamos las estructuras y variables.
2. Como ya se dijo antes, recorremos cada elemento de A solamente una vez.
3. Luego, se construye el Snek iterando sobre su largo final, partiendo en la celda final.
4. Finalmente, se puede retornar el Snek

3. Construir Snek

a) Se agrega la celda al stack.

3. Construir Snek

a) Se agrega la celda al stack.

b) Se busca si la celda adyacente hacia arriba tenía un largo menor en 1 y estaba conectada

3. Construir Snek

a) Se agrega la celda al stack.

b) Se busca si la celda adyacente hacia arriba tenía un largo menor en 1 y estaba conectada

c) Si no, se hace lo mismo para la de la izquierda*

Lógica del algoritmo

1. Primero inicializamos las estructuras y variables.
2. Como ya se dijo antes, recorremos cada elemento de A solamente una vez.
3. Luego, se construye el Snek iterando sobre su largo final, partiendo en la celda final.
4. Finalmente, se puede retornar el Snek

- Problema del Snek
- Decisiones estratégicas
- Lógica algoritmo
- Implementación

1. Inicializar estructuras y variables

```
1: function FindSnek(A,m,n)
2:     B ← Matriz de tamaño m x n llena de 1s
3:     Mejor_Largo ← 0
4:     Columna_Final ← 0
5:     Fila_Final ← 0
6:     Snek ← Stack
```

2. Recorrer cada elemento

```
8:  for i = 0...m do
9:      for j = 0...n do
10:         if i = j = 0 then
11:             Continuar                          ▷ No consideramos la primera celda
12:         end if
13:         if A_{i,j} = A_{i-1,j} ± 1 then
14:             B_{i,j} ← máx(B_{i-1,j} + 1, B_{i,j})
15:         end if
16:         if A_{i,j} = A_{i,j-1} ± 1 then
17:             B_{i,j} ← máx(B_{i,j-1} + 1, B_{i,j})
18:         end if
19:         if Mejor_Largo < B_{i,j} then          ▷ Tenemos una serpiente más larga?
20:             Columna_final ← j
21:             Fila_final ← i
22:             Mejor_Largo ← B_{i,j}
23:         end if
24:     end for
25: end for
```

3. Construir Snek

26: **for** i = 0 ... $Mejor\_Largo$ **do** ▷ Reconstrucción de atrás hacia adelante

27: $Snek.push(A_{Fila\_F,Col\_F})$

28: **if** $Fila\_F > 0 and A_{Fila\_F,Col\_F} = A_{Fila\_F-1,Col\_F} and B_{Fila\_F-1,Col\_F} = B_{Fila\_F,Col\_F} - 1 \pm 1$ **then**

29: $Fila\_F-$

30: **else if** $A_{Fila\_F,Col\_F} = A_{Fila\_F,Col\_F-1} and B_{Fila\_F,Col\_F-1} = B_{Fila\_F,Col\_F} - 1 \pm 1$ **then**

31: Col_F-

32: **end if**

33: **end for**

4. Retornar Snek

34: **RETORNAR** Snek

¿Dudas o Comentarios?