Proyecto de curso

Calculando el plan de riego óptimo de una finca

Análisis de Algoritmos II

Escuela de Ingeniería de Sistemas y Computación

Profesor Juan Francisco Díaz Frias Profesor Jesús Alexander Aranda
Monitor Mauricio Muñoz

Febrero de 2024

1. Introducción

El presente proyecto tiene por objeto verificar que los estudiantes han adquirido el siguiente resultado de aprendizaje: Aplica técnicas de algoritmos voraces y dinámicos, en la construcción de algoritmos, para solucionar problemas de naturaleza combinatoria. Para ello, los estudiantes deben demostrar que logran:

- Aplicar cada estrategia de diseño (fuerza bruta, voraz y programación dinámica) a un ejemplo práctico.
- Usar un enfoque de fuerza bruta para resolver un problema y determinar si la estrategia conlleva a una solución óptima.
- Usar un enfoque voraz para resolver un problema y determinar si la estrategia conlleva a una solución óptima.
- Usar programación dinámica para resolver un problema comprendiendo que dicha estrategia conlleva a una solución óptima.
- Reconocer las ventajas y desventajas de utilizar diferentes estrategias de diseño (fuerza bruta, voraz y programación dinámica).
- Comparar distintas estrategias (fuerza bruta, voraz y programación dinámica) para un problema dado a partir del análisis del respectivo análisis de complejidad y optimalidad.

Para ello el estudiante:

- Desarrolla un programa utilizando un lenguaje de programación adecuado para resolver en grupo un proyecto de programación planteado por el profesor.
- Escribe un informe de proyecto, presentando los aspectos más relevantes del desarrollo realizado, para que un lector pueda evaluar el proyecto.
- Desarrolla una presentación digital, con los aspectos más relevantes del desarrollo realizado, para sustentar el trabajo ante los compañeros y el profesor.

2. El problema del riego óptimo

2.1. El problema

Como todos lo saben, el Valle del Cauca es una región agrícola en general, y cañera en particular. Los cultivos de caña son inmensamente grandes; a veces se pueden recorrer kilómetros y kilómetros de carretera sin dejar de ver a lado y lado cultivos de caña de azúcar.

Los cultivos se organizan normalmente en *Fincas* y cada una de éstas en *Tablones*. Un tablón es entonces la unidad básica de estructuración de los cultivos de caña.

Cada tablón tiene asociadas unas características como su tamaño, los días de empezado el cultivo, la etapa actual del cultivo y muchas otras. Una muy importante es su capacidad hídrica, medida en términos de la cantidad de agua almacenada en el tablón, la cual se traduce finalmente en un *número máximo de días que el tablón puede permanecer sin ser regado* de manera que el cultivo sembrado en él no sufra (a este número lo llamaremos **tiempo de supervivencia**). De acuerdo a su tamaño, cada tablón tiene asociado también el *número de días que se demora regarlo* para restablecerle su capacidad hídrica, en caso de estar en su capacidad mínima (a este tiempo lo llamaremos **tiempo de regado**). Una última característica de cada tablón es su *prioridad*. Dependiendo de las características del cultivo, de su estado y de otras variables, cada tablón es etiquetado con un número entre 1 y 4, siendo 4 la prioridad más alta.

Un proceso esencial en el mantenimiento de los cultivos es entonces **la programación del riego de cada tablón** de una finca determinada. Por los altos costos y por la naturaleza del trabajo, es imposible tener un sistema fijo de riego asociado a cada tablón. En cambio se usan sistemas móviles de riego que permiten con ese solo recurso, regar varios tablones, simplemente movilizando el sistema móvil de un tablón a otro, obviamente con un costo de movilidad, que en todo caso es mucho menos costoso que tener un sistema fijo de riego en cada tablón.

Con el presente proyecto se pretende abordar a manera de ejercicio el problema de calcular las fechas de inicio de riego de cada tablón de una finca de manera que se minimice el tiempo de sufrimiento de los cultivos por falta de agua (y, por simplicidad, sin tener en cuenta los costos por la movilidad del sistema móvil de riego). Se supone que se tiene un solo recurso de riego para toda la finca y que el consumo de agua es proporcional al tiempo que lleva el tablón sin ser regado y a la capacidad hídrica inicial del mismo.

F: Finca/Sec de tablones
Tiene 0 o más tablones

T: Tablon
ts: supervivencia (resiste sin regarse)
tr: regado (se demora regando)
pi: prioridad (De 1 a 4, 4 es más prioridad)

2.2. Formalización

Una finca $\mathcal{F}$ es una secuencia de tablones $\mathcal{F} =< T_0, \ldots, T_{n-1} >$. Cada T_i es una tupla $< ts_i^{\mathcal{F}}, tr_i^{\mathcal{F}}, p_i^{\mathcal{F}} >$, donde $ts_i^{\mathcal{F}}$ representa el tiempo de supervivencia, $tr_i^{\mathcal{F}}$ el tiempo de regado y $p_i^{\mathcal{F}}$ la prioridad del tablón i de la finca $\mathcal{F}$, para $0 \leq i < n$.

pi es programacion de riego

Una programación de riego de la finca $\mathcal{F}$ es una permutación $\Pi =< \pi_0, \pi_2, \ldots, \pi_{n-1} >$ de $< 0, 1, 2, \ldots, n-1 >$ que indica el orden en que se regarán los tablones. Primero el tablón T_{π_0}, luego el tablón T_{π_1} y así sucesivamente hasta regar el tablón $T_{\pi_{n-1}}$.

Dada una programación de riego Π para la finca $\mathcal{F}$, se puede calcular, para cada tablón T_i, el tiempo en que se iniciará su riego t_i^{Π} según esa programación, así:

1. El tablón T_{π_0} se comienza a regar en tiempo $t_{\pi_0}^{\Pi} = 0$.
2. El tablón T_{π_j} se comienza a regar en tiempo $t_{\pi_j}^{\Pi} = t_{\pi_{j-1}}^{\Pi} + tr_{\pi_{j-1}}^{\mathcal{F}}, j = 1..n-1$.

t_inicio anterior
t_regado anterior

El costo de riego del tablón T_i de la finca $\mathcal{F}$, dada una programación de riego Π se define así:

t_sup - (t_ini + r_reg)
t_sup - t_reg >= t_ini

$$CR_{\mathcal{F}}^{\Pi}[i] = \begin{cases} ts_i^{\mathcal{F}} - (t_i^{\Pi} + tr_i^{\mathcal{F}}) & \text{si } ts_i^{\mathcal{F}} - tr_i^{\mathcal{F}} \geq t_i^{\Pi} \\ p_i^{\mathcal{F}} * ((t_i^{\Pi} + tr_i^{\mathcal{F}}) - ts_i^{\mathcal{F}}) & \text{sino} \end{cases}$$

Costo de riego de un tablon (i)

prio * ((t_ini + t_reg) - t_sup))

Por último, el costo total de riego de la finca $\mathcal{F}$, dada una programación de riego Π se define como:

Costo de regar Finca = Suma de costos de cada tablon

$$CR_{\mathcal{F}}^{\Pi} = \sum_{i=0}^{n-1} C_{\mathcal{F}}^{\Pi}[i].$$

El problema del riego óptimo se define entonces así:
Entrada: Una finca $\mathcal{F}$
Salida: Una programación de riego Π para la finca $\mathcal{F}$, tal que $CR_{\mathcal{F}}^{\Pi}$ sea mínimo

Tablon: <ts, tr, pi>

2.3. ¿Entendimos el problema?

2.3.1. Ejemplo 1

t0 t1 t2 t3 t4

- Entrada: $\mathcal{F}_1 =<< 10, 3, 4 >, < 5, 3, 3 >, < 2, 2, 1 >< 8, 1, 1 >, < 6, 4, 2 >>$
- Programación 1: $\Pi_1 =< 0, 1, 4, 2, 3 >$, luego el tiempo en que se inicia el riego de cada tablón sería:

  Orden de riego: t0, t1, t4, t2, t3

  Empieza t0 en 0 termina en 3
  Empieza t1 en 3 termina en 6
  Empieza t4 en 6 termina en 10
  Empieza t2 en 10 termina en 12
  Empieza t3 en 12 termina en 13

  $$t^{\Pi_1} =< 0, 3, 10, 12, 6 > .$$

  Por tanto

  $$CR_{\mathcal{F}_1}^{\Pi_1} = \underset{t0}{7} + \underset{t1}{1 * 3} + \underset{t2}{10 * 1} + \underset{t3}{5 * 1} + \underset{t4}{4 * 2} = 33$$

- Programación 2: $\Pi_2 =< 2, 1, 4, 3, 0 >$, luego el tiempo en que se inicia el riego de cada tablón sería:

  $$t^{\Pi_2} =< 10, 2, 0, 9, 5 > .$$

  Por tanto

  $$CR_{\mathcal{F}_1}^{\Pi_2} = 3 * 4 + 0 + 0 + 2 * 1 + 3 * 2 = 20$$

- En este caso es mejor solución Π_2 que Π_1. ¿Habrá mejores?

2.3.2. Ejemplo 2

- Entrada: $\mathcal{F}_2 =<< 9, 3, 4 >, < 5, 3, 3 >, < 2, 2, 1 >, < 8, 1, 1 >, < 6, 4, 2 >>$
- Programación 1: $\Pi_1 =< 2, 1, 4, 3, 0 >$, luego el tiempo en que se inicia el riego de cada tablón sería

  $$t^{\Pi_1} =< 10, 2, 0, 9, 5 > .$$

  Por tanto

  $$CR_{\mathcal{F}_2}^{\Pi_1} = 4 * 4 + 0 + 0 + 2 * 1 + 3 * 2 = 24$$

- Programación 2: $\Pi_2 =< 2, 1, 4, 0, 3 >$, luego el tiempo en que se inicia el riego de cada tablón sería

  $$t^{\Pi_2} =< 9, 2, 0, 12, 5 > .$$

  Por tanto

  $$CR_{\mathcal{F}_2}^{\Pi_2} = 3 * 4 + 0 + 0 + 5 * 1 + 3 * 2 = 23$$

- La mejor solución entre estas dos, es Π_2. ¿Habrá mejores?

3. El proyecto

Su proyecto consiste en explorar tres alternativas para solucionar el problema, una de fuerza bruta, una voraz y una usando programación dinámica, programarlas, analizarlas y hacer un informe al respecto.

3.1. Usando fuerza bruta

Considere el algoritmo que genera todas las soluciones posibles y entre ellas escoge la mejor.

3.1.1. Complejidad

¿Cuál es el orden de complejidad del algoritmo? Argumente su respuesta.

3.1.2. Corrección

¿El algoritmo siempre da la respuesta correcta al problema? Argumente su respuesta.

3.2. Usando un algoritmo voraz

3.2.1. Describiendo el algoritmo

Describa un algoritmo voraz para abordar el problema (ustedes como grupo deciden su estrategia voraz; la idea es que sea la más sencilla posible que ojalá les permita encontrar el óptimo siempre o la mayoría de las veces). **Se dará el puntaje máximo por este criterio a los mejores algoritmos voraces en términos de relación calidad de la solución / tiempo usado para encontrarla**.

3.2.2. Entendimos el algoritmo

Calcule la salida de su algoritmo para las entradas presentadas en 2.3.1 y 2.3.2 y calcule el costo de la solución. ¿Es la solución óptima?

Describa al menos 4 entradas más, calcule la solución entregada por el algoritmo y verifique si es o no la óptima. Guarde los resultados en una tabla.

3.2.3. Complejidad

¿Cuál es el orden de complejidad del algoritmo? Argumente su respuesta.

3.2.4. Corrección

¿El algoritmo siempre da la respuesta correcta al problema? Argumente su respuesta. Si la respuesta es negativa, adicionalmente analice cuándo si y cuándo no da la respuesta correcta.

3.3. Usando programación dinámica

Una alternativa a las dos anteriores sería utilizar programación dinámica. Para hacerlo es necesario comprobar que el problema tiene la propiedad de subestructura óptima y dejarse guiar por ella.

3.3.1. Caracterizando la estructura de una solución óptima

Caracterice la estructura de una solución óptima y a partir de ella describa el conjunto de subproblemas para los cuáles será necesario calcular las soluciones óptimas. ¿Cuántos subproblemas hay?

3.3.2. Definiendo recursivamente el valor de una solución óptima

Defina recursivamente el costo de una solución óptima para cada subproblema definido en el punto anterior.

3.3.3. Describiendo el algoritmo para calcular el costo de una solución óptima

Describa el algoritmo que calcula el costo de una solución óptima al problema original, basado en el punto anterior.

3.3.4. Describiendo el algoritmo para calcular una solución óptima

Tome el algoritmo descrito en el punto anterior y adicione:

- Una estructura que permita almacenar datos para construir una solución óptima cuyo valor sea el calculado por el algoritmo.
- Un algoritmo que recorra esa estructura construyendo una solución de costo óptimo.

3.3.5. Complejidad

Complejidad en tiempo. Calcule la complejidad en tiempo del algoritmo en términos de n.

Complejidad en espacio. Calcule la complejidad en espacio del algoritmo en términos de n.

¿Es útil en la práctica? Suponga que tiene un equipo que procesa 3×10^8 operaciones por minuto y que utiliza 4 bytes de almacenamiento por celda.
Si $n = 2^k$:

- Argumente para qué valores de k el tiempo que se tomará el equipo en resolver el problema es aceptable o inaceptable en la práctica.
- Argumente para qué valores de k el espacio que se tomará el equipo en resolver el problema es aceptable o inaceptable en la práctica.

[**Sugerencia:**] Utilice las siguientes aproximaciones:
$n + 1 \sim n,\ A + 1 \sim A$
1 minuto $= \frac{1}{3^4 2^8 5^2}$ años
1 año $= 3^4 2^8 5^2$ minutos $= 518400$ minutos
2^{10} Bytes $= 1$ KB
2^{10} KB $= 1$ MB
2^{10} MB $= 1$ GB
10^3 MB ~ 1 GB

3.4. Implementación

Cada grupo debe entregar el código correspondiente a:

- En el ambiente de programación de su preferencia, un programa que contenga una función por cada una de las soluciones diseñadas y analizadas (fuerza bruta, voraz y programación dinámica). Cada función recibe una finca f y devuelve una pareja (Π, CR_f^{Π}) tal que Π es la respuesta al problema. Esas funciones en su código deben llamarse `roFB, roV` y `roPD`.

- Usando la tecnología de su preferencia, una interfaz que permita leer entradas al problema desde un archivo de texto en un formato especificado, y escribir las salidas al problema en un archivo de texto en un formato especificado, así como visualizar las entradas, después de leerlas, y las salidas después de ser calculadas por cualquiera de las funciones solicitadas. Las salidas deben poder ser revisadas por medio de la interfaz en cuanto a su estructura y calidad (optimalidad). En particular debe ser fácil comprobar la corrección del costo de la solución dada.

Todos los programas deben correr bajo un mismo ambiente; se debe entregar el programa **ejecutable** y el código fuente.

3.4.1. Formato de las entradas

Las entradas vendrán en un archivo de texto con $n + 1$ líneas así:

```
n
ts0,tr0,p0
ts1,tr1,p1
.
.
.
ts(n-1),tr(n-1,p(n-1)
```

Es decir, la primera línea trae el número de tablones de la finca (un entero n); las siguientes n líneas traen los tiempos de supervivencia, tiempos de regado y prioridades de cada uno de los n tablones.

3.4.2. Formato de las salidas

Las salidas se deben producir en un archivo de texto con $n + 1$ líneas así:

```
Costo
pi0
pi1
.
.
.
pi(n-1)
```

Es decir, la primera línea trae el costo de la solución; luego vienen n líneas, con los índices de los tablones en el orden en que deben ser regados.

3.5. Entrega

La entrega se debe realizar vía el campus virtual en las fechas previstas para ello, por uno sólo de los integrantes del grupo.

La fecha de entrega límite es el 4 de abril de 2024 a las 23:59. La sustentación será en las sesiones del 11 y 16 de abril de 2024.

Debe entregar un informe en formato pdf, los archivos fuente, un archivo Readme.txt que describa todos los archivos entregados y las instrucciones para ejecutar la aplicación. Todo lo anterior en un solo archivo empaquetado cuyo nombre contiene los apellidos de los autores y cuya extensión corresponde al modo de compresión. Por ejemplo ArandaDiazMuñoz.zip o ArandaDiazMuñoz.rar, o ArandaDiaz-Muñoz.tgz o ...

4. Sustentación y calificación

El trabajo debe ser sustentado por los autores en el día y hora especificado para su grupo. **La calificación del proyecto para el grupo** se hará teniendo en cuenta los siguientes criterios:

1. Informe (1/2) Debe responder a cada una de las solicitudes formuladas en el enunciado y debe contener al menos una sección dedicada a cada tipo de algoritmo implementado (fuerza bruta, voraz, programación dinámica), además de una sección dedicada a comparar los resultados de las tres soluciones implementadas (para esto use tablas y diseñe casos de prueba y documente cómo los diseñó) y una sección dedicada a concluir sobre las ventajas y desventajas de usar en la práctica los diferentes enfoques.

2. Implementación (1/2). Esto quiere decir que el código entregado funciona por lo menos para las diferentes pruebas que tendremos para evaluarlo y corresponde con todo lo solicitado (incluyendo la interfaz de lectura y visualización solicitada, y la generación de casos de prueba)

En todos los casos la sustentación será pilar fundamental de la nota individual asignada a cada integrante del grupo. En la sustentación se demuestra la capacidad del grupo de navegar en el código y realizar cambios rápidamente en él, así como la capacidad de responder con solvencia a las preguntas que se le realicen.

Cada persona de cada grupo, después de la sustentación, tendrá asignado un número real (el factor de multiplicación) entre 0 y 1, correspondiente al grado de calidad de su sustentación. **Su nota definitiva será la calificación del proyecto para el grupo, multiplicada por ese valor.** Si su asignación es 1, su nota será la del proyecto. Pero si su asignación es 0.9, su nota será 0.9 por la nota del proyecto. La no asistencia a la sustentación tendrá como resultado una asignación de un factor de 0.

La idea es que lo que no sea debidamente sustentado no vale así funcione muy bien!!! Y que, del trabajo en grupo, es importante que todos aprendan, no sólo algunos.

Éxitos!!!