PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMOS AVANZADOS

2da. práctica (tipo B)
(Segundo Semestre 2024)

Duración: 1h 50 min.

- **No puede utilizar apuntes, solo hojas sueltas en blanco.**
- En cada función el alumno deberá incluir, a modo de comentario, la forma de solución que utiliza para resolver el problema. De no incluirse dicho comentario, el alumno perderá el derecho a reclamo en esa pregunta.
- No puede emplear plantillas o funciones no vistas en los cursos de programación de la especialidad.
- Los programas deben ser desarrollados en el lenguaje C++. Si la implementación es diferente a la estrategia indicada o no la incluye, la pregunta no será corregida.
- Un programa que no muestre resultados coherentes y/o útiles será corregido sobre el 50% del puntaje asignado a dicha pregunta.
- Debe utilizar comentarios para explicar la lógica seguida en el programa elaborado. El orden será parte de la evaluación.
- Se utilizarán herramientas para la detección de plagios, por tal motivo si se encuentran soluciones similares, se anulará la evaluación a todos los implicados y se procederá con las medidas disciplinarias dispuestas por la FCI.
- **Solo está permitido acceder a la plataforma de PAIDEIA, cualquier tipo de navegación, búsqueda o uso de herramientas de comunicación se considera plagio por tal motivo se anulará la evaluación y se procederá con las medidas disciplinarias dispuestas por la FCI.**
- Para esta evaluación solo se permite el uso de las librerías **iostream, iomanip, climits cmath, fstream, vector, string o cstring**
- Su trabajo deberá ser subido a PAIDEIA.
- **Es obligatorio usar como compilador NetBeans.**
- Los archivos deben llevar como nombre su código de la siguiente forma codigo_LAB2_P# (donde # representa el número de la pregunta a resolver)

Pregunta 1 (10 puntos)

Una aeronave sale desea llegar de la ciudad 1 a la cuidad n, con la menor cantidad de aterrizajes, por tal motivo durante su trayecto debe bajar en ciudades que tengan el combustible suficiente para avanzar la mayor cantidad posible sin aterrizar y así minimizar la cantidad de descensos. A continuación, algunos ejemplos:

De acuerdo con los datos de entrada: {5, 2, 1, 4, 1} n=5

La respuesta será **1** aterrizaje, ya que el avión partió con el combustible para volar hacia 5 ciudades, y la ciudad 5 de destino era la cuarta desde su despegue, en otras palabras, le quedaba combustible para avanzar hacia una ciudad más.

De acuerdo con los datos de entrada: {1, 2, 1, 4, 2, 3} n=6

La respuesta será **3** aterrizajes, ya que el avión partió con el combustible para volar solo hasta 1 ciudad, aterrizando en la ciudad 2, la cual tenía el combustible para llevarla hasta 2 ciudades más adelante, por tal motivo eligió entre la ciudad 3 y 4, seleccionando la ciudad 4, ya que tenía combustible suficiente para terminar la ruta en la ciudad 6.

De acuerdo con los datos de entrada: {1, 2, 1, 4, 2, 3, 5, 2, 4, 2, 6} n=11

La respuesta será **4** aterrizajes.

Desarrolle un programa **empleando la técnica de programación dinámica**, que le permita calcular la cantidad mínima de aterrizajes que realizará el avión en su ruta de la ciudad 1 a n.

Pregunta 2 (10 puntos)

El **problema del corte de barra** (Rod Cutting Problem) es un problema clásico de optimización en programación dinámica. El objetivo es cortar una barra de longitud n en piezas de diferentes longitudes para maximizar las ganancias. Cada longitud tiene un valor asociado, y se debe decidir cómo cortar la barra para obtener el mayor beneficio posible.

Descripción

Supongamos que tienes una barra de longitud n, y una lista de precios donde el precio de una barra de longitud i es p[i]. Debes determinar cómo cortar la barra en segmentos de varias longitudes para maximizar la ganancia total.

Ejemplo

Considera que tienes una barra de longitud n = 4 y la siguiente tabla de precios para las distintas longitudes:

Longitud de barra	1	2	3	4
Precio	2	5	7	8

El objetivo es encontrar la mejor forma de cortar la barra para maximizar las ganancias.

Posibles cortes:

1. No cortar la barra: Ganancia = 8 (barra de longitud 4).
2. Cortar la barra en dos piezas de longitud 2: Ganancia = 5 + 5 = 10.
3. Cortar la barra en una pieza de longitud 1 y otra de longitud 3: Ganancia = 2 + 7 = 9.
4. Cortar la barra en cuatro piezas de longitud 1: Ganancia = 2 + 2 + 2 + 2 = 8.

Mejor opción:

El mejor corte sería dividir la barra en dos piezas de longitud 2, lo que da una ganancia total de 10.

La idea es resolver subproblemas más pequeños y construir una solución óptima usando los resultados de esos subproblemas. Se define una tabla dp[i] que almacena la ganancia máxima para una barra de longitud i. El valor óptimo para una barra de longitud n se calcula usando la relación:

$$dp[n] = \max(p[j] + dp[n-j]) \quad \text{para} \quad 1 \leq j \leq n$$

Se le pide construir un programa en C++ **empleando la técnica de programación dinámica**, usando los datos anteriores y que permita calcular la ganancia máxima obtenida al cortar la barra.

Para que las soluciones sean válidas en ambas preguntas debe mostrar el arreglo o matriz de soluciones parciales, recuerde que solo debe emplear iteraciones, si alguna respuesta emplea recursión la pregunta queda anulada.

Al finalizar el laboratorio, comprima la carpeta de su proyecto empleando el programa Zip que viene por defecto en el Windows, **no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares**. Luego súbalo a la tarea programa en Paideia para este laboratorio.

Profesores del curso:

Manuel Tupia
Rony Cueva

San Miguel, 21 de septiembre del 2024