CI-1221 Estructuras de Datos y Análisis de Algoritmos

I ciclo de 2018

III Tarea Programada

Fechas límite de entrega:
Parte I: martes 12 de junio
Parte II: martes 26 de junio

Descripción del problema

La tarea consiste en resolver el problema del Viaje del profesor Casasola al Parque de Diversiones. El profesor Casasola entra al parque a la hora T_e, y aunque le fascina la diversión, debe salir a más tardar a la hora T_s, porque su casa...

está sola. 

Su tarea es escribir algoritmos que usen las técnicas de búsqueda exhaustiva, programación dinámica y, si es posible, algoritmos ávidos, que, para una horas de entrada T_e y de salida T_s cualesquiera, indiquen en qué orden debe el profesor visitar las atracciones para lograr disfrutarlas la mayor cantidad de veces en el tiempo disponible.

Para ello asuma que:

1. Las atracciones están enumeradas de 1 a n.
2. Para disfrutar de una atracción se debe hacer fila y el tiempo de espera varía según la atracción y la hora.
3. El tiempo está discretizado en múltiplos de 5 min y va de las 9:00 a. m. ($T = 0$) a las 11:55 a. m. ($T = 35$).
4. El tiempo que se demora de la entrada del parque a la atracción i es e_i ($i = 1, 2, \ldots, n$).
5. El tiempo de espera en una fila está dado por $f_{i,j}$, donde i corresponde a la atracción ($i = 1, 2, \ldots, n$) y j al tiempo en el que se llegó a la fila ($j = 0, 1, \ldots, 35$).

6. El tiempo de disfrute de la atracción i es d_i $(i = 1, 2, \ldots, n)$.

7. El tiempo que toma trasladarse de una atracción i a una atracción j es $t_{i,j}$ $(i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n)$.

8. El tiempo que se tarda en llegar de la atracción i a la salida del parque es s_i $(i = 1, 2, \ldots, n)$.

9. Al profesor no le importa repetir una atracción una y otra vez, siempre y cuando no sea de forma consecutiva.

El profesor planea llegar al parque a las 9:00 a. m., 9:30 a. m. o 10:00 a. m. y permanecer en él 1 o 2 horas. Indíquele al profesor a qué hora debe llegar al parque para disfrutar las atracciones la mayor cantidad de veces si decide permanecer en él cada uno de estos lapsos, es decir, cuál de los tres momentos de llegada anteriores es el mejor si va a permanecer en el parque una hora y si va a permanecer dos horas). Para ello básese en la información suministrada por los estudiantes que visitaron el parque. (Los tiempos de espera no especificados corresponden a lapsos en los que la atracción no estuvo disponible).

Conformación de grupos

Esta tarea se puede realizar en grupos de dos personas o de forma individual.

Entregables de la tarea

La tarea tiene dos entregas. La primera es un reporte preliminar que incluye la formulación y resultados obtenidos con el método de búsqueda exhaustiva (parte I). La segunda es un reporte final que incluye tanto este método como los de programación dinámica y algoritmos ávidos (parte II). A cada uno de los reportes debe adjuntarse el código de los algoritmos en el archivo de cabeceras «`Parque.h`» mostrado en el listado 1. A cada método le entran como parámetros los tiempos de espera en el lapso que el profesor puede estar en el parque (es decir, de T_e a $T_s - 1$), los tiempos de disfrute (que no varían con la hora) y los tiempos de traslado de una atracción a otra (que tampoco varían con la hora).

Los tiempos de espera están dados por la matriz `espera` en la que cada fila corresponde a una hora distinta de llegada a la fila de una atracción (la primer fila corresponde a T_e y la última a $T_s - 1$) y cada una de las n columnas corresponde a una distinta atracción. (Si una atracción no está disponible a una hora específica, el tiempo de espera incluye tanto el tiempo que falta hasta que la atracción esté disponible como el tiempo que se demora haciendo fila. Por ejemplo, si la entrada $e_{i,j}$ tiene un valor de 1 y la atracción no está disponible en los dos tiempos anteriores $i-i$ e $i-2$, las entradas $e_{i-1,j}$ y $e_{i-2,j}$ deben tener valores de 2 y 3, respectivamente, puesto que si se llega a la atracción a la hora i y se tiene que hacer fila 5 minutos, es lógico suponer que si hubiera llegado 5 minutos antes se hubiera tenido que esperar 10 minutos y que si hubiera llegado 10 minutos antes se hubiera tenido que esperar 15 minutos).

Listado 1: Plantilla para la solución «Parque.h».

```
#ifndef PARQUE
#define PARQUE
#include <vector>

class Parque{
    public:
    Parque(){}
    ~Parque(){}
    std::vector<int> busquedaExhaustiva(const int** espera, int
        m, int n, const int* disfrute, const int** traslado);
    std::vector<int> programacionDinamica(const int** espera,
       int m, int n, const int* disfrute, const int** traslado)
       ;
    std::vector<int> algoritmoAvido(const int** espera, int m,
       int n, const int* disfrute, const int** traslado);
};

#endif
```

Los tiempos de disfrute están dados por el vector `disfrute`. La interpretación es simple: la i-ésima entrada indica el tiempo de disfrute de la i-ésima atracción. Se asume que el tiempo de disfrute es el mismo independientemente de la hora.

Los tiempos de traslado están dados por la matriz `traslado`, en la que la primera de las $n + 1$ filas corresponde al tiempo de traslado de la entrada a cada una de las atracciones y la última de las $n + 1$ columnas corresponde al tiempo de traslado de cada una de las atracciones a la salida del parque. Las otras entradas de la matriz indican el tiempo de traslado de una atracción a otra. Específicamente, la entrada `traslado[i][j]` indica el tiempo que toma trasladarse de la atracción $i - 1$ a la atracción j. (Se le resta uno a i porque la primer fila de la matriz contiene los tiempos de traslado de la entrada a cada una de las atracciones).

Forma de entrega

La tarea debe entregarse antes de la fecha y hora especificadas, por medio de la plataforma Mediación Virtual. La tarea se puede dejar en estado «borrador» (no es necesario «enviarla a revisión»). Basta con enviar una versión por grupo (no es necesario que cada miembro del grupo la envíe; con solo uno que lo haga es suficiente). Es responsabilidad del estudiante verificar que la plataforma haya recibido la tarea y que esté intacta (bajando la tarea y verificando su integridad, especialmente si está comprimida, p. ej., en formato zip). En caso de que haya múltiples entregas, se considerará solamente

la *última*, y si esta se entregó después de la fecha y hora límites, se aplicará la penalización acordada en la CARTA AL ESTUDIANTE: *la calificación recibida por tareas tardías no será más alta que la calificación recibida por cualquiera de los estudiantes que haya entregado la tarea a tiempo.* Si tiene problemas para subir la tarea y el plazo está cerca de cumplirse, envíela a los asistentes, con copia al profesor. Favor poner en el «asunto» lo siguiente: *Tarea de Estructuras de Datos.* En caso de que se reciban múltiples versiones por correo electrónico, se revisará solamente la *primera* enviada. La política de penalización por entrega tardía mencionada en el párrafo anterior también aplica a entregas hechas por correo electrónico.

Parte I.

1. Búsqueda exhaustiva (50 pts.)

1. Encuentre una representación vectorial de la solución al problema, indicando claramente el significado de cada una de sus entradas. [2 pts.]

2. Determine el espacio E en el que habita la solución, definiendo claramente el (los) conjunto(s) a el (los) que pertenece(n) cada una de las entradas. [2 pts.]

3. Determine la cardinalidad del espacio. [2 pts.] (Válido solo si el espacio es correcto).

4. Si es posible, acote el espacio utilizando una restricción del tipo $E' = \{\sigma \in E : \text{restricción sobre } \sigma\}$. [5 pts.] (Válido solo si el espacio es correcto).

5. Escriba un algoritmo en C++ que explore todas las soluciones candidatas en E o E' y devuelva una solución óptima. Utilice el encabezado definido en el fichero «`Parque.h`». [30 pts.]

   Observación. El puntaje obtenido en esta sección es

   $$\frac{30}{k} \sum_{i=1}^{k} \frac{A'_i}{A_i},$$

   donde k es la cantidad de casos de prueba (se darán oportunamente), A'_i es la cantidad de atracciones que el algoritmo permitió visitar (incluyendo repeticiones) y A_i es la cantidad máxima de atracciones que realmente se pueden visitar (incluyendo repeticiones). (Esto último lo conocen el asistente y el profesor, pero lo guardarán en secreto).

6. Determine una cota asintótica para el tiempo de ejecución del algoritmo. [2 pts.] (Válido solo si el algoritmo es correcto).

7. Ejecute el algoritmo con cada uno de los casos de prueba dados. Si el tiempo de ejecución excede 10 minutos para una cierta visita, puede abortarla y omitir visitas más largas. Reporte en un cuadro el tiempo de ejecución de cada una de las ejecuciones y grafique los tiempos como función de la duración de la visita. Si hay varios casos de prueba con la misma duración, en el gráfico use el tiempo promedio de ellos. [5 pts.] (Válido solo si el algoritmo es correcto).

8. Indique si el crecimiento de los tiempos fue el esperado y explique por qué. [2 pts.] (Válido solo si el algoritmo es correcto).

Parte II.

2. Programación dinámica (50 pts.)

1. Determine si el problema se puede resolver a partir de soluciones a subproblemas, y si es así, describa cómo construir tal solución. [2 pts.]

2. Determine un oráculo (describiendo claramente su significado y el de sus argumentos) [3 pts.], el valor objetivo [2 pts.], y los pasos base [2 pts.] y recursivo [8 pts.] que permiten construirlo.

3. Basado en su respuesta al punto anterior, escriba un algoritmo que resuelva el problema. Utilice el encabezado definido en «`Parque.h`». [20 pts.]
   Observación. El puntaje obtenido en esta sección será
   $$\frac{20}{K}\sum_{i=1}^{K}\frac{A'_i}{A_i},$$
   donde k es la cantidad de casos de prueba, A'_i es la cantidad de atracciones que el algoritmo permitió visitar (incluyendo repeticiones) y A_i es la cantidad máxima de atracciones que realmente se pueden visitar (incluyendo repeticiones).

4. Determine una cota asintótica para el tiempo de ejecución del algoritmo. [3 pts.] como función de

5. Ejecute el algoritmo usando los casos de prueba suministrados. Indique si las soluciones obtenidas producen el mismo número de atracciones visitadas (incluyendo repeticiones) que las soluciones obtenidas mediante búsqueda exhaustiva [3 pts.]. Grafique los tiempos de ejecución como función de la cantidad de grupos, representando los tiempos en escala logarítmica, y agregue al gráfico la curva correspondiente a los tiempos producidos por el algoritmo de búsqueda exhaustiva [4 pts.]. Compare las curvas y diga si la relación entre los tiempos fue la esperada [3 pts.] (Válido solo si el algoritmo es correcto).

3. Algoritmos ávidos (5–35 pts.)

1. Determine si es posible resolver el problema mediante un algoritmo ávido cuyo tiempo de ejecución sea asintóticamente mejor que el de programación dinámica. Explique su respuesta. [5 pts.]

2. Si su respuesta al punto anterior fue afirmativa:

   a) Escriba un algoritmo ávido que resuelva el problema. Utilice el encabezado definido en «`Parque.h`». [10 pts.]
   Observación. El puntaje obtenido en esta sección será

   $$\frac{10}{K} \sum_{i=1}^{k} \frac{A'_i}{A_i},$$

   donde k es la cantidad de casos de prueba, A'_i es la cantidad de atracciones que el algoritmo permitió visitar (incluyendo repeticiones) y A_i es la cantidad máxima de atracciones que realmente se pueden visitar (incluyendo repeticiones).

   b) Utilice el «método de la transformación» para demostrar que la estrategia es correcta. [7 pts.] (Válido solo si el algoritmo es correcto).

   c) Determine una cota asintótica para el tiempo de ejecución del algoritmo. [3 pts.] (Válido solo si el algoritmo es correcto).

   d) Ejecute el algoritmo usando los datos de prueba suministrados. Indique si las soluciones obtenidas producen el mismo número de atracciones visitadas (incluyendo repeticiones) que los algoritmos anteriores [3 pts.]. Grafique los tiempos de ejecución como función de la cantidad de grupos, representando los tiempos en escala logarítmica, y agregue al gráfico la curva correspondiente a los tiempos producidos por el algoritmo de búsqueda exhaustiva y programación dinámica [4 pts.]. Compare las curvas y diga si la relación entre los tiempos fue la esperada [3 pts.] (Válido solo si el algoritmo es correcto).