UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍAS INFORMÁTICA y DE TELECOMUNICACIÓN

Departamento de Ciencias de la Computación e Inteligencia Artificial

Doble Grado en Ingeniería Informática y Matemáticas

Curso 2022-2023

Algorítmica

Guión de Prácticas

Práctica 2: Divide y vencerás

Índice

1. Umbrales en la técnica "Divide y vencerás" 1

2. Enunciado de los problemas 2
2.1. Cálculo de posiciones alcanzables 3
2.2. Superficie de reparto con dragones 3
2.3. Calendarios de campeonatos 3
2.4. Amistad en la granja 4

3. Trabajo a realizar 4
3.1. Tareas 4
3.2. Memoria 5

4. Evaluación de la práctica 6

Objetivo

El objetivo de esta práctica es que el estudiante comprenda y asimile el funcionamiento de la técnica de diseño de algoritmos "Divide y Vencerás". Esta práctica será realizada por grupos de alumnos. A cada grupo se le asignará un problema para el que tendrán que diseñar varios algoritmos según las indicaciones incluidas en este guión.

1. Umbrales en la técnica "Divide y vencerás"

La idea fundamental de técnica "Divide y vencerás" es la descomposición de un problema dado en problemas, habitualmente del mismo tipo, pero de complejidad (*tamaño*) menor. Esto se suele traducir en un proceso recursivo que requiere de alguna condición de parada, por debajo de la cual ya no se aplica recursividad sino que se resuelven directamente los casos de problema que han surgido. Eso requiere que, además, del método general, se disponga de un algoritmo específico para resolver los casos más pequeños. A este algoritmo se le denomina algoritmo base o específico.

Una cuestión clave es determinar cuál es el criterio para dejar de usar la recursividad y aplicar el método específico directamente. Este criterio se

implementa definiendo un valor para el tamaño del problema a partir del cual se resuelve con el método específico. A este tamaño se le denomina **umbral** y su determinación es un aspecto crítico en la implementación del algoritmo "Divide y vencerás". Sabemos que su valor óptimo no es fijo, y que depende de diversos factores: algoritmo base, implementación, casos estudiados, etc. En esta práctica vamos a abordar su cálculo según los tres enfoques que se describen a continuación.

1. Cálculo del valor del umbral teórico. A partir de la expresión del tiempo de ejecución del algoritmo base (teórica) y la expresión recurrente del tiempo de ejecución para el método recursivo, hay que determinar cuál es el tamaño del problema para el que ambos tiempos son iguales (suponiendo un sólo nivel de recursión). Es decir, se plantea una ecuación igualando el tiempo del algoritmo específico al del desarrollo en un sólo paso del tiempo del algoritmo recursivo. Se resuelve esta ecuación y este valor es el *umbral teórico*.

   Este resultado tiene un valor normalmente indicativo dado que algoritmo específico se aplica para casos de tamaño pequeño, y en esta región las constantes ocultas tienen un papel importante.

2. Cálculo del valor del *umbral óptimo*. Está definido para cada implementación concreta del algoritmo. Se necesita conocer el tiempo exacto según un enfoque híbrido del tiempo de ejecución del algoritmo específico. Se plantea una ecuación igualando esta fórmula de tiempo con la expresión recurrente del método recursivo. Y se resuelve la ecuación resultante.

3. Umbrales de tanteo. Se evalúan los tiempos de ejecución resultantes para distintos valores (inferiores y superiores al umbral óptimo).

4. Cálculo gráfico del umbral. Representar en una gráfica comparativa la evolución del tiempo de ejecución del algoritmo para los distintos valores del umbral estudiados. Es decir, para cada selección distinta del umbral (teórico, óptimo y de tanteo) deberá dibujar una curva con los tiempos de los casos medidos empíricamente. Comente los resultados justificando cuál o cuáles de las variantes darían mejores resultados.

2. Enunciado de los problemas

En esta sección se incluyen los enunciados de los problemas que tienen que resolver cada grupo de alumnos. Los problemas considerados son los que se describen en las subsecciones de esta sección.

2.1. Cálculo de posiciones alcanzables

Supongamos que tenemos un conjunto n posiciones $\mathcal{P}$ ubicadas sobre un mapa. Para cada posición conocemos sus coordenadas x e y en el mapa ($p_i = (x_i, y_i)$). Por simplicidad, asumiremos que no existen dos posiciones distintas con las mismas coordenadas, esto es dadas $p_i = (x_i, y_i)$ y $p_j = (x_j, y_j)$ se verifica que no pueden darse simultáneamente $x_i = x_j$ e $y_i = y_j$. Desde una posición p_i se puede alcanzar a otra posición p_j si $x_i > x_j$ y $y_i > y_j$.

El puesto de mando tiene como objetivo colocar un conjunto de fortificaciones en la zona delimitada por el mapa y para ello necesita conocer para cada ubicación $p_i \in \mathcal{P}$ cuántas posiciones se pueden alcanzar así como el listado de las mismas.

Por ejemplo, supongamos que tenemos el conjunto de posiciones siguientes:

p0	p1	p2	p3	p4	p5	p6	p7	p8	p9
(2,78)	(3,25)	(4,18)	(0,95)	(1,8)	(8,47)	(9,7)	(5,77)	(6,75)	(7,71)

Entonces, para cada punto, el número de posiciones alcanzables es:

p0	p1	p2	p3	p4	p5	p6	p7	p8	p9
1	1	1	0	0	3	0	3	3	3

donde, por ejemplo, para p5 se alcanzan las posiciones p1, p2 y p4.

2.2. Superficie de reparto con dragones

Los diseñadores de un videojuego ubicado en la "Tierra Media" de J.R.R. Tolkien quieren implementar el reparto de vituallas a hobbits y enanos dispersos en territorio hostil, controlado por los orcos. Los enanos han conseguido domar varios dragones y los utilizan para sobrevolar el territorio enemigo y lanzar paquetes con provisiones desde ellos. Tras cada incursión pueden conocer la ubicación exacta en que aterriza cada uno de sus paquetes. Pero quieren caracterizar la superficie total cubierta en cada envío. Es decir, considerando como entrada el conjunto $\mathcal{P}$ de puntos $p_i = (x_i, y_i)$ donde han caído cada uno de los n paquetes lanzados, determinar el polígono convexo de menor superficie que incluye todos los puntos.

2.3. Calendarios de campeonatos

Para el próximo curso la E.T.S.I.I.T. va a organizar un campeonato de ajedrez. Se han inscrito n jugadores. El campeonato se va a organizar en forma de torneo de modo que cada jugador ha de jugar exactamente una vez

contra cada adversario. Además, cada jugador ha de jugar exactamente una única partida en cada jornada, con la excepción posible de una sola jornada en la que no jugará. Se necesita un algoritmo que genere el calendario de jornadas para el torneo de forma que:

1. Si n es par, el campeonato se realizará en $n - 1$ jornadas.
2. Si n es impar (y, trivialmente, mayor que 1), el campeonato se realizará en n jornadas.

2.4. Amistad en la granja

Un granjero está muy preocupado por el bienestar de su cabaña vacuna. Ha adquirido nuevos terrenos para ampliar el área en que sus vacas pueden pastar y estar relajadas para así producir leche de más calidad. Sabe que las vacas también tienen sus propias preferencias sociales y que, cuando están libres, cada vaca se acerca a aquella compañera con la que tiene mayor afinidad. Para tener un buen seguimiento de la actividad de cada vaca, cada una tiene un collar equipado con un GPS. De este modo, tiene perfectamente ubicados a todos los ejemplares. Una de las cuestiones de mayor interés es conocer las afinidades entre las vacas.

Para cada día que salen del establo al prado se quiere saber cuál es la pareja de vacas que tienen mayor afinidad, es decir, cuál es el par de vacas que están más cerca. Diseñad un algoritmo para identificar cuáles son esas dos vacas para cada distribución de las mismas.

3. Trabajo a realizar

Para alcanzar el objetivo previsto con esta práctica, los alumnos deberán realizar las tareas que se detallan en la sección siguiente, documentando el trabajo realizado en una memoria con el contenido y estructura que se indica en la subsección 3.2.

3.1. Tareas

1. Estudiar en profundidad el problema asignado, asegurándose de comprender bien las entradas y salidas involucradas, así como las relaciones entre éstas. Deberán identificar el tipo de información que define cada caso del problema y las magnitudes que definen su tamaño.

2. Diseñar un algoritmo específico para resolver el problema, que se aplicará en la resolución de aquellos casos cuyo tamaño sea inferior al *umbral*.

3. Diseñar un algoritmo para resolver el problema basado en la técnica "Divide y vencerás".

4. Implementar tanto el algoritmo específico como el basado en "Divide y vencerás".

5. Diseñar e implementar un generador de casos para el problema asignado.

6. Realizar el análisis de la eficiencia teórica, empírica e híbrida de ambos algoritmos.

7. Calcular los umbrales (teórico, óptimo y de tanteo) para las implementaciones de algoritmos realizadas.

8. Elaborar una memoria que documente el trabajo realizado de acuerdo a las indicaciones incluidas en la subsección 3.2.

3.2. Memoria

Todo el trabajo realizado debe redactarse en una memoria. La estructura de este documento será la siguiente:

1. Portada. Incluyendo las denominaciones de titulación, asignatura y práctica. También el nombre completo de los alumnos que forman el grupo y su dirección de correo electrónico.

2. Autores. Indicar el % del trabajo realizado por cada alumno, especificando qué tareas ha realizado cada uno.

3. Objetivos. Descripción del objetivo de la práctica.

4. Definición del problema. Descripción de los casos usados en la evaluación de la eficiencia. Descripción completa del entorno de análisis: hardware, sistema operativo, compilador, etc. empleados. Descripción del método de medición de tiempos.

5. Algoritmo específico. Diseño del algoritmo. Detalles de implementación. Análisis de la eficiencia teórica, empírica e híbrida.

6. Algoritmo divide y vencerás. Diseño del algoritmo. Detalles de implementación. Cálculo de umbrales teórico, óptimo y de tanteo. Análisis de la eficiencia teórica, empírica e híbrida.

7. Conclusiones.

4. Evaluación de la práctica

Esta práctica se realizará por grupos formados por 3 alumnos. Los grupos serán conformados por los propios alumnos. Cada grupo se registrará a través de la actividad "Configura tu grupo de trabajo" en la página de la asignatura en Prado.

Una vez constituidos los grupos, el profesor asignará a cada grupo el problema que deben resolver.

Los alumnos habrán de entregar dos archivos en la actividad correspondiente incluida en la página correspondiente a la asignatura en la plataforma Prado. El primero archivo será la memoria, en formato pdf.

El segundo será el código fuente de las implementaciones realizadas empaquetado en un fichero .zip. El código estará organizado en tres carpetas de nombres: `Generador`, `Epecifico` y `Dyv`. Cada una de esas carpetas incluirá todos los módulos de código fuente necesarios para generar el programa binario binario correspondiente las respectivas implementaciones de: generador de casos del problema, algoritmo específico y algoritmo basado en "Divide y vencerás". Además, en la carpeta raíz habrá un único fichero `Makefile` cuyo objetivo por defecto construirá los tres binarios de nombres `generador`, `especifico` y `dyv`, respectivamente. Los programas `especifico` y `dyv` aceptarán un único argumento en línea de órdenes, que será el nombre de un fichero que incluirá los datos de un caso y enviarán a la salida estándar la solución calculada.

La fecha límite para entregar la memoria es el día **9 de abril de 2023** a las 23:59 horas.

Además de hacer el trabajo y entregar la memoria, cada equipo tendrá que elaborar una breve presentación del trabajo realizado que expondrá públicamente en clase de prácticas, de acuerdo a la convocatoria establecida por el profesor. Es obligatoria la participación de todos los miembros del grupo en la exposición.