Fecha de entrega

La fecha límite de entrega es el **jueves 21 de mayo de 2024**.

Presentación

Esta práctica tiene como objetivo el diseño e implementación de una aplicación paralela basada en MPI. Para su realización se pondrán en práctica muchos de los conceptos presentados en los temas 4 y 5 de la asignatura.

Hay que presentar los archivos con el código fuente realizado. Toda la codificación se hará exclusivamente en lenguaje C y MPI.

Enunciado de la práctica

Erase una vez, en un lejano país, un rey que poseía en su reino una pequeña y valiosísima colección de árboles, que había heredado de los viajes de sus antepasados a otros reinos lejanos. Para proteger los árboles de los ladrones, el rey ordenó que se construyese una valla a su alrededor. La tarea recayó sobre el mago del reino, hombre sabio y prudente.

Desafortunadamente, el mago se dio cuenta con gran presteza de que el único material disponible para construir la valla era la madera de los propios árboles. Dicho de otro modo, era necesario cortar algunos árboles para construir la valla alrededor de los restantes. Lógicamente, para no provocar la ira del rey y perder su empleo y su cabeza en el empeño, el mago quiso minimizar el valor de los árboles que tuviesen que ser cortados. Se retiró a su torre y estuvo allí hasta que encontró la mejor solución posible al problema planteado por el rey. La valla fue construida y todos vivieron felices.

En esta práctica, se desarrollará una aplicación paralela basada en las comunicaciones por paso de mensajes de MPI, con el objetivo de resolver el problema al que se enfrentó el mago de nuestra historia.

Resolución mediante la búsqueda exhaustiva.

La resolución de este problema se puede abordar siguiendo una estrategia simple en la que se generan y evalúan todas las combinaciones de n elementos cogidos de r en r (siendo r el número de árboles a cortar, $1 \leq r \leq n-1$), se calcula la longitud de la valla necesaria para cerrar los n-r árboles restantes y se comprueba que esta sea una solución factible si la madera de los r árboles cortados es suficiente para construir la valla. Para cada solución factible se evalúa si ésta minimiza o no el valor de los árboles cortados de la mejor solución encontrada hasta el momento con objeto de guardarla o descartarla, según sea el caso.

Las diferentes combinaciones de árboles talados se van a codificar mediante la representación binaria de un entero largo. Por lo tanto, en función de la arquitectura de procesador (32 bits o 64 bits) podremos codificar soluciones hasta 32 o 64 árboles. En esta codificación cualquier número en el rango representa una posible solución, mediante la conversión del número a binario y haciendo que cada digito se corresponda al estado de uno de los árboles de entrada, si el digito está a 1 significa que ese árbol se ha talado y el digito está a 0 que no. Los árboles se empiezan a numera por la derecha, siendo el bit menos significativo el primer árbol. En el del segundo ejemplo que os proporcionamos, la combinación óptima 22 se representa de forma binaria como 010110 y significa que se han talado los árboles 2,3 y 5 para vallar los arboles no talados (1,4 y 6).

Para calcular la longitud de la valla es necesario calcular el convex hull (ver Figura 1) de un conjunto de puntos Q (que se define como el menor polígono convexo P para el que cada punto de Q está en un lado del polígono P o en su interior). En [1][2] se describen diferentes algoritmos para calcular convex hulls.

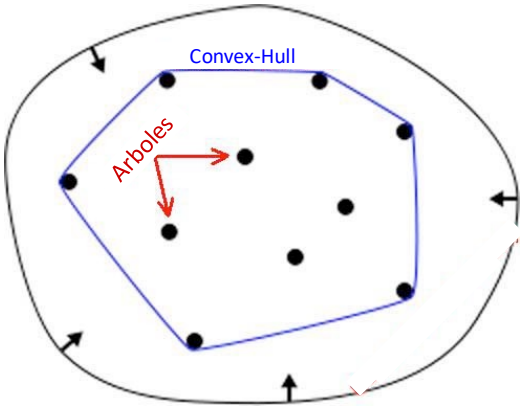

Figura 1. Convex-Hull para calcular la cerca de árboles.

Si los árboles talados generan suficiente madera para toda la longitud de la cerca que se necesita (convex hull árboles no talados) entonces se trata de una solución válida y el coste de esta es el coste de los árboles talados. El objetivo del problema es encontrar la combinación con un menor coste. Tened es cuenta que puede existir varias combinaciones con el menor coste, pero el algoritmo solo devuelve la primera encontrada.

Resolución mediante árboles de búsqueda (Branch & Bound) con poda.

Otro método alternativo para resolver el problema es la utilización un árbol de búsqueda para buscar la solución óptima.

En este caso, el árbol de búsqueda consistiría en un árbol binario, en donde el nivel del árbol representa el árbol que se está procesando. Cada nodo del árbol tiene dos hijos definidos en función de si el árbol de ese nivel se incorpora (se tala) o se descarta de solución previa.

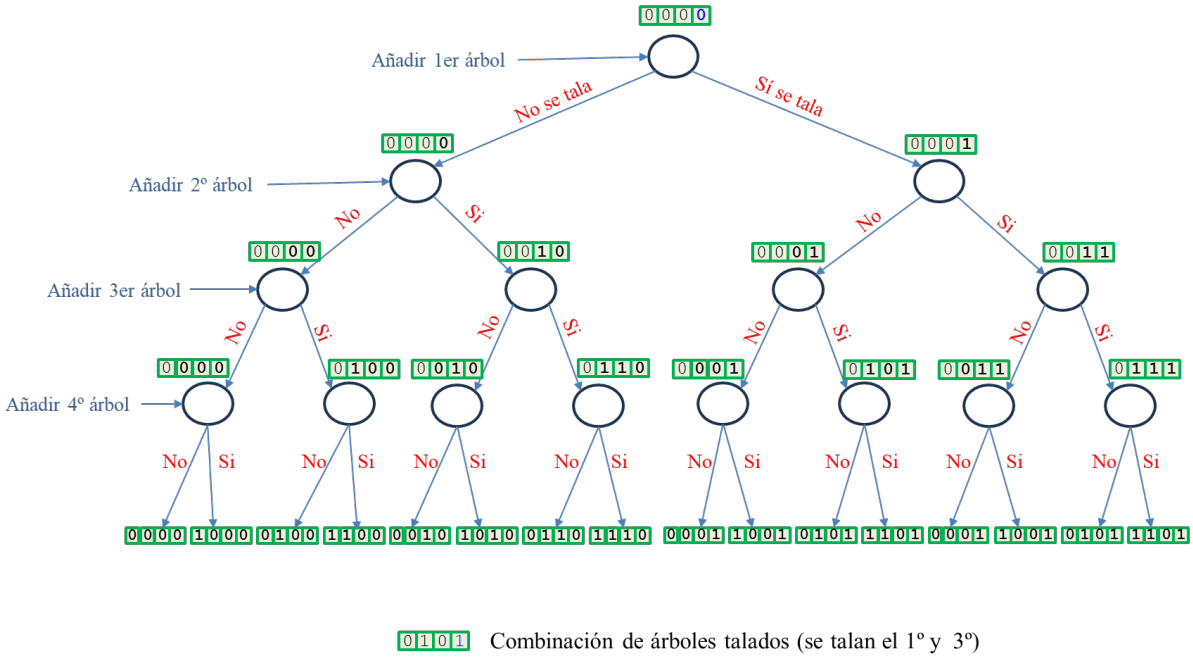

Figura 2. Árbol de búsqueda espacio de soluciones para calcular la cerca de árboles.

De esta forma, tal y como se muestra en la figura 2, si queremos resolver el problema para un bosque con 4 árboles, tendríamos un árbol binario de 4 niveles, para un total de 16 posibles combinaciones/soluciones.

Además, en este método se puede aplicar la técnica de la poda. Está técnica consiste en que mientras se va procesando las diferentes soluciones (nodos) de árbol de búsqueda, una vez se alcanza un nodo que ya no puede modificar la mejor solución encontrada hasta ese momento (óptimo parcial) ya no es necesario seguir procesando sus nodos hijos y por lo tanto, se produce una poda de esa rama del árbol.

Se pide implementar la versión paralela de los dos métodos para calcular la solución óptima del problema de la cerca de los árboles, teniendo en cuenta su eficiencia, tanto desde el punto de vista del balanceo de carga entre los procesadores y como del solapamiento de las comunicaciones.

Ficheros de prueba

Para probar y depurar el programa podéis utilizar los ficheros que se adjunta con la versión secuencial de la práctica. Tenéis diferentes tamaños de fichero para poder utilizar durante la depuración (con preparar algunos ficheros de ejemplo sencillos con pocos árboles (3 y 6 árboles8). Junto con estos ficheros de pequeño tamaño, también disponéis de ficheros de pruebas medianos y grades que requieren un mayor tiempo de procesamiento secuencial para evaluar la escalabilidad de solución propuesta.

- **Datos de entrada.**

Los datos de entrada del problema están almacenados en un archivo donde se describirá un bosque hipotético. La primera línea contiene un número entero n, $2 \leq n \leq 64$, que corresponde al número de árboles en el bosque. Los árboles se identifican mediante un valor entero correlativo de 1 a n. Cada una de las siguientes n líneas del fichero contiene 4 enteros x_i, y_i, v_i, l_i que describen un árbol en particular. La posición del árbol en el plano está definida por (x_i, y_i), su valor es v_i y la longitud de la valla que se puede construir con la madera de ese árbol es l_i. Los valores de v_i y l_i están entre 0 y 10000.

- **Datos de salida.**

El programa debe calcular el subconjunto de árboles que, usando su madera, permite encerrar a los restantes árboles dentro de una única valla. El objetivo es lograr encontrar el subconjunto que requiere un menor coste. Si existe más de un subconjunto con el coste mínimo, hay que escoger el que tenga el menor número de árboles. Por simplicidad, se considerará que los árboles tienen diámetro cero.
La salida del programa debe identificar cada uno de los árboles a cortar, la longitud de la madera sobrante, el valor del conjunto de árboles cortados y el valor de los árboles restantes.

Ejemplo 1

Entrada	**Salida**
3 3 0 10 2 5 5 20 25 7 -3 30 32	Cortar árboles: 2 Madera sobrante: 15.00 Valor árboles cortados: 20 Valor árboles restantes: 40

Ejemplo 2

Entrada	Salida
6 0 0 8 3 1 4 3 2 2 1 7 1 4 1 2 3 3 5 4 6 2 3 9 8	Cortar árboles: 2 4 5 Madera sobrante: 3.16 Valor árboles cortados: 9 Valor árboles restantes: 24

Evaluación

La evaluación de la práctica se realizará en función de la calidad de la aplicación paralela implementada y las prestaciones obtenidas.

Se utilizarán los siguientes criterios a la hora de evaluar la práctica:

- Aspectos que se evalúan para cada una de las versiones:
  - Versión no parelela (Susp)
  - No se implementa uno de los dos métodos (-3.0p)
  - Resultados incorrectos o no determinista (-2.0p)
  - Eficiencia aplicación / Speed-up (+1.0p,..,-3.0p)
  - No solapar cómputo y comunicaciones: sin comunicaciones asíncronas (-1.0p)
  - Trabajo procesos desbalanceado (-1.0p)
  - Extra: Utilizar tipos derivados (+0.5p)
  - Extra: Utilizar comunicaciones colectivas (+0.5p)
- Evaluación del informe de la práctica:

  + Informe completo, con diseño, validación realizada de la aplicación, descripción de los problemas encontrados y análisis de prestaciones. (-1,0p,...,+1.0p)
- Estilo del código.
  - Comentarios
  - Utilizar una correcta identación
  - Descomposición Funcional (Código modular y estructurado)
  - Control de errores.
  - Test unitarios para verificar la correcta ejecución de la lógica de la aplicación.

Se puede tener en cuenta criterios adicionales en función de la implementación entregada.

Análisis prestaciones

Calcular el tiempo de ejecución de la versión paralela y compararlo con el de la versión secuencial. Discutir los resultados obtenidos. El objetivo de la práctica es que **la versión paralela sea más rápida que la versión secuencial**, cuando se procese ficheros grandes.

Analizar también, el tiempo de ejecución de la versión paralela en función del número de procesos/procesadores. Entregar un pequeño informe en donde se muestren (preferiblemente de forma gráfica) y expliquen los resultados obtenidos.

Este análisis se tiene que realizar para los métodos de la aplicación implementados (exhaustivo y Branch&bound). Presentar de forma gráfica y comentar los resultados obtenidos en el informe de la práctica.

Versión Secuencial

Junto con el enunciado se os proporciona una versión secuencial en C para resolver el problema descrito. Esta versión consiste en un proyecto CLion que calcula la solución del problema de optimación utilizando ambos métodos.

- Ejecutar con ejemplo pequeño
  - ./CalcCerca ./ConjuntoPruebas/Ejemplo1_3Arboles.dat
- Ejecutar con ejemplo pequeño (generando resultados en Ejemplo2_6Arboles.res)
  - ./CalcCerca ./ConjuntoPruebas/Ejemplo2_6Arboles.dat ./Ejemplo2_6Arboles.res

Referencias

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. "**Introduction to Algorithms**", Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0262032937.

[2] "**Convex Hull**", http://softsurfer.com/Archive/algorithm_0109/algorithm_0109.htm

Formato de entrega

MUY IMPORTANTE: La entrega de código que no compile correctamente, implicará suspende TODA la práctica.

No se aceptarán prácticas entregadas fuera de plazo (salvo por razones muy justificadas).

La entrega presencial de estos ejercicios es obligatoria.

Comenzar vuestros programas con los comentarios:

```
/* ------------------------------------------------------------------
Práctica 1.
Código fuente: CalcArbolesParallel.c
Grau Informática
NIF i Nombre completo autor1.
NIF i Nombre completo autor2.
------------------------------------------------------------------ */
```

Para presentar la práctica dirigiros al apartado de Actividades del Campus Virtual de la asignatura de Computación Distribuida y Aplicaciones, ir a la actividad "Práctica 1" y seguid las instrucciones allí indicadas.

Se creará un fichero tar con todos los ficheros fuente de la práctica, con el siguiente comando:

```
$ tar -cvf prac2.tgz fichero1 fichero2 ...
```

se creará el fichero "prac2.tgz" en donde se habrán empaquetado y comprimido los ficheros "fichero1", fichero2, y ...

Para extraer la información de un fichero tar se puede utilizar el comando:

```
$ tar -xvf tot.tar
```

El nombre del fichero tar tendrá el siguiente formato: "Apellido1Apellido2PRA1.tar". Los apellidos se escribirán sin acentos. Si hay dos autores, indicar los dos apellidos de cada uno de los autores separados por "_". Por ejemplo, el estudiante "Perico Pirulo Palotes" utilizará el nombre de fichero: `PiruloPalotesPRA1.tar`

Fecha de entrega

Entrega a través de Sakai el 21 de mayo, entrega presencial la semana del 29 de mayo de 2024.