2011

Tecnológico de Costa Rica Sede San Carlos

Algoritmos y estructuras de Datos 2

IC: 3001

Profesora: Bach. Ana Lorena Valerio Solís

Luis Alonso Vega Brenes 201042592
Luis Diego Alfaro Alpízar 200941331

27/04/2011

Descripción del problema

El programa que se debe realizar debe ser capaz de generar un grafo para la administración de proyectos, utilizando el cálculo de tiempos con los algoritmos de PERT y CPM, determinar la ruta crítica y mostrar visualmente un listado de actividades realizando un recorrido por amplitud y por profundidad. La estructura que tomará el grafo depende de los datos de entrada, los cuales se ingresan mediante archivos de texto plano con un formato específico que se debe diseñar.

Es necesario mostrar medidas empíricas de los algoritmos utilizados, entre ellas las asignaciones, comparaciones, líneas de código totales y ejecutadas, tiempo de ejecución.

Una vez que el grafo está construido, deberá ser posible modificar el valor del tiempo de cualquier actividad en él.

Solución del problema

Para diseñar e implementar la aplicación, se tomaron en cuenta algunas características básicas que debían realizarse inicialmente. En esta sección explicaremos cronológicamente las fases que se fueron implementando y la manera en que se hicieron, para dar a entender más claramente cómo se llegó al resultado esperado.

Inicialmente partimos del hecho de la necesidad de leer un archivo[3], para lo cual era necesario plantear primeramente la forma en que estaría estructurado. Para que el archivo fuera de texto plano y más entendible para las personas que diseñaran o vieran el formato, fue necesario seleccionar varios caracteres especiales, los cuales funcionarían como separadores de secciones. Finalmente determinamos que una buena opción sería utilizar las comas (,) y los punto y comas (;) para separar atributos y actividades, respectivamente. Cada actividad debía almacenar información sobre su nombre, tiempo de duración y actividades predecesoras. Considerando que la cantidad de actividades que podían haber antes de cualquiera era indefinida, utilizamos el siguiente formato:

```
nombreActividad,tiempo,nomrePredecesor1,nombrePredecesor2,...,nombrePredecesorN;
```

Para señalar cuales son los predecesores de cada actividad se utiliza el nombre de cada una de ellas. El símbolo de punto y coma indica que ahí termina la actividad. Tomando esta estructura en consideración, diseñamos un algoritmo que leyera el contenido de texto del archivo dado y que lo subdividiera por caracteres separadores y que se creara una estructura de actividad para ser insertada en la lista.

Ahora bien, necesitamos implementar varias estructuras[1] dentro de un grafo, las cuales fueron la de vértice, arco y subarco. El vértice marca los eventos, es decir, el inicio o fin de una o más actividades. El arco describe una actividad con sus propiedades y los subarcos son simples contenedores utilizados para crear sublistas de arcos.

Estructura	Propiedades o atributos
Grafo	Vértice inicial, Primer actividad, Total de vértices, Ruta crítica
Vértice	ID, ET, Arcos Predecesores, Arcos Antecesores.
Arco	Nombre, Tiempo, Destino
Subarco	Arco Destino

Estructuras de datos utilizados

Una vez que fue posible reconocer la información representada por los archivos y estructurarla como actividades (arcos) procedimos a realizar un algoritmo que permitiera insertar la actividad según la posición correspondiente, la cual dependía de los predecesores que tuviera la actividad. Esta función fue un tanto extensa ya que era necesario considerar varias opciones cuando se insertaba en el grafo. Algunos ejemplos: Era la primera actividad, no tenía predecesores, los tenía pero eran compartidos, etc.

Eventualmente fue posible crear la estructura correctamente, por lo que restaba hacer uso de los algoritmos de PERT y CPM, y de los recorridos por Anchura (Amplitud) y Profundidad. Estos primeros debían calcular el Early Time de cada evento, variando solamente en el cálculo del tiempo de cada actividad. CPM es directo y utiliza el tiempo dado para cada actividad[5], por lo que no fue necesario agregar más que un método que recorriera el grafo por cada ruta posible y marcara los eventos con los tiempos menores con los que se llegaba a cada uno. Por otro lado, PERT necesita utilizar una fórmula que considera el tiempo optimista y pesimista[4], así como el tiempo estimado. Como referencia, mostramos la fórmula:

$$t = \frac{a + 4m + b}{6}$$

Donde a y b son los tiempos optimista y pesimista y m el tiempo más probable. Para calcular los tiempos optimistas y pesimistas sin tener que agregar espacio en el archivo, decidimos implementar un sistema de probabilidades, en el que el tiempo optimista era el tiempo normal con una reducción del 30%, mientras que el pesimista lo calculamos con un aumento de 40% en el tiempo normal. Tomando estos cálculos, tenemos la siguiente fórmula generada que utilizamos para calcular PERT:

$$t = \frac{0.7m + 4m + 1.4m}{6}$$

Junto a estos métodos, se encuentra el de calcular la ruta crítica de cada grafo. Para implementarla decidimos recorrer por profundidad cada vértice pero solo por los arcos que, sumando su tiempo y el tiempo acumulado, obtuvieran el ET del vértice asociado a su destino. Al final, solo podrá llegarse al final de una forma y se almacena la información de los arcos visitados.

Luego debíamos implementar los métodos de recorrido e impresión del grafo. Iniciamos con el método por amplitud, pues era más sencillo. Simplemente recorremos la lista general de eventos o vértices y en cada uno revisamos sus arcos sucesores. Imprimimos tanto la información de cada evento como la de cada actividad.

Después del primer método de recorrido, implementamos el método por profundidad. Un algoritmo recursivo que se ramifica por cada ruta posible del grafo. Por cada arco visitado se muestra la información pertinente al mismo, tomando en cuenta el evento que lo origina, el vértice destino, el tiempo de duración y, por supuesto, el nombre de la actividad.

Para lograr modificar el tiempo de las actividades una vez que estuvieran en el grafo (almacenadas en memoria), utilizamos un algoritmo de búsqueda por nombre, ya implementado y lo utilizamos para obtener la actividad seleccionada de una lista y modificar su propiedad de tiempo según lo que ingresa el usuario en un campo de texto.

Análisis de Resultados.

Tarea a realizar	Realizado con éxito	No realizado	Defectuosas
Abrir y leer archivos de texto desde cualquier unidad	X		
Insertar actividades leídas desde el archivo	X		
Modificar el peso de los arcos	X		
Imprimir la estructura grafo en anchura con los datos generados con el algoritmo de PERT.	X		
Imprimir la estructura grafo en anchura con los datos generados con el algoritmo de CPM.	X		
Imprimir la estructura grafo en profundidad con los datos generados con el algoritmo de PERT.	X		
Imprimir la estructura grafo en profundidad con los datos generados con el algoritmo de CPM.	X		
Imprima todas las variables de medición	X		

Análisis empírico de costes de algoritmos

Anchura y profundidad

Tabla 1.1: Método de Anchura.

Operaciones	Cantidad de actividades		
	10	20	40
Asignaciones	179	365	893
Comparaciones	84	168	396
Cantidad de líneas de código ejecutadas	311	635	1553
Tiempo de ejecución	2 ms	3 ms	10 ms

Tabla 1.2: Factor de incremento del número de operaciones al crecer el tamaño

Talla	***Factor talla***	***Factor Asig***	***Factor Comp***	***Tiempo de ejecución***	***Cantidad de líneas ejecutadas***
De 10 a 20	2	2	2	2	2
De 20 a 40	2	2	2	3	2

El comportamiento esperado del algoritmo será que, en general:

Operaciones	***O(1)***	***O(n)***	***O(n^2)***	***O(n*log n)***	
Las asignaciones crecen		**X**			con la talla
Las comparaciones crecen		**X**			con la talla
Tiempo de ejecución crece				**X**	con la talla
Cantidad de líneas ejecutadas crece		**X**			con la talla

Tabla 2.1: Método de Profundidad.

Operaciones	Cantidad de actividades		
	10	20	40
Asignaciones	235	1715	17777
Comparaciones	130	900	9200
Cantidad de líneas de código ejecutadas	387	2697	27597
Tiempo de ejecución	2 ms	11 ms	84 ms

Tabla 2.2: Factor de incremento del número de operaciones al crecer el tamaño

Talla	***Factor talla***	***Factor Asig***	***Factor Comp***	***Tiempo de ejecución***	***Cantidad de líneas ejecutadas***
De 10 a 20	2	7	7	6	7
De 20 a 40	2	10	10	8	10

El comportamiento esperado del algoritmo será que, en general:

Operaciones	***O(1)***	***O(n)***	***O(n^2)***	***O(n * log n)***	
Las asignaciones crecen				**X**	con la talla
Las comparaciones crecen				**X**	con la talla
Tiempo de ejecución crece				**X**	con la talla
Cantidad de líneas ejecutadas crece				**X**	con la talla

¿Los datos de la Tabla 2.1 coinciden con los de la tabla 1.1? Marcar con X la respuesta que coincida con tu caso.

(a)	Coinciden todos los resultados	
(b)	Sólo coinciden las asignaciones	
(c)	Sólo coinciden las comparaciones	
(d)	Sólo coinciden en el tiempo de ejecución.	
(e)	No coincide ninguno	**X**

Justifica el resultado obtenido en la cuestión anterior

Lo más probable es que la diferencia entre ambos algoritmos se base en la forma de implementación en que ambos se realizaron. Siendo anchura un método iterativo y profundidad uno recursivo.

Adicionalmente tenemos la siguiente tabla con la comparación de implementación.

Tabla 3. Comparación de implementación de algoritmos de recorrido del grafo

Algoritmo	**Clasificación**	**Líneas de código totales**
Anchura	Iterativo	21
Profundidad	Recursivo	12

Los datos de prueba se encuentran en la sección de Anexos, anexo 1.

PERT y CPM

Tabla 3.1: Método de PERT.

Operaciones	Cantidad de actividades		
	10	20	40
Asignaciones	81	235	2145
Comparaciones	68	248	1862
Cantidad de líneas de código ejecutadas	171	565	4627
Tiempo de ejecución	0 ms.	0 ms.	0 ms.

Tabla 3.2: Factor de incremento del número de operaciones al crecer el tamaño

Talla	***Factor talla***	***Factor Asig***	***Factor Comp***	***Tiempo de ejecución***	***Cantidad de líneas ejecutadas***
De 10 a 20	2	3	4	0	3
De 20 a 40	2	9	7	0	8

El comportamiento esperado del algoritmo será que, en general:

Operaciones	***O(1)***	***O(n)***	***O(n^2)***	***O(n * log n)***	
Las asignaciones crecen			**X**		con la talla
Las comparaciones crecen				**X**	con la talla
Tiempo de ejecución crece	**X**				con la talla
Cantidad de líneas ejecutadas crece				**X**	con la talla

Tabla 4.1: Método de CPM.

Operaciones	Cantidad de actividades		
	10	20	40
Asignaciones	59	189	1352
Comparaciones	68	248	1862
Cantidad de líneas de código ejecutadas	149	519	3834
Tiempo de ejecución	0 ms.	0 ms.	0 ms.

Tabla 4.2: Factor de incremento del número de operaciones al crecer el tamaño

Talla	***Factor talla***	***Factor Asig***	***Factor Comp***	***Tiempo de ejecución***	***Cantidad de líneas ejecutadas***
De 10 a 20	2	3	4	0	4
De 20 a 40	2	7	8	0	7

El comportamiento esperado del algoritmo será que, en general:

Operaciones	***O(1)***	***O(n)***	***O(n^2)***	***O(n * log n)***	
Las asignaciones crecen				**X**	con la talla
Las comparaciones crecen				**X**	con la talla
Tiempo de ejecución crece	**X**				con la talla
Cantidad de líneas ejecutadas crece				**X**	con la talla

¿Los datos de la Tabla 3.1 coinciden con los de la tabla 4.1? Marcar con X la respuesta que coincida con tu caso.

(a)	Coinciden todos los resultados	
(b)	Sólo coinciden las asignaciones	
(c)	Coinciden las comparaciones	**X**
(d)	Coinciden en el tiempo de ejecución.	**X**

Justifica el resultado obtenido en la cuestión anterior

Los algoritmos son muy similares y realizan una cantidad idéntica de comparaciones, y varían ligeramente en la cantidad de asignaciones. El tiempo de ejecución no fue medible realmente, pues no se encontró en el ambiente de desarrollo y las librerías usadas, ninguna forma de obtener tiempos más exactos y los algoritmos tomaban muy poco tiempo como para ser reconocido en intervalos muy grandes.

Adicionalmente tenemos la siguiente tabla con la comparación de implementación.

Tabla 3. Comparación de implementación de algoritmos de recorrido del grafo

Algoritmo	**Líneas de código totales**
PERT	9
CPM	9

Los datos de prueba se encuentran en la sección de Anexos, anexo 1.

Conclusiones

Para describir el porqué de las medidas de los 4 algoritmos es más fácil cuando se conoce la forma en la que fueron diseñados e implementados. A continuación discutimos un poco los resultados obtenidos entre cada par de algoritmos.

- Anchura vs. Profundidad

En el caso de los recorridos del grafo, cabe destacar que es más fácil visualizar las rutas observando los datos generados por el método por profundidad, pues permite observar jerárquicamente la forma en la que se pueden desglosar los caminos y opciones en cada actividad. Sin embargo, encontrar cada ruta tiene un costo, y en este caso el algoritmo recursivo produce que se necesiten de más operaciones para ejecutarse. Por su parte, el recorrido por anchura permite encontrar rápidamente cada actividad y mostrar su información, pero es más difícil tener una idea visual de cómo está construido todo el grafo. Para concluir con esta comparación, podemos decir que tener una mejor visualización de los datos le produce un mayor costo al computador, mientras que una ilustración básica de estos induce a un menor costo y mayor velocidad.

- PERT vs. CPM

Los métodos de cálculo de tiempo son muy similares, y varían en sus comparaciones, lo que produce que varíen también en el total de líneas de código que son ejecutadas. También se puede destacar que al ser algoritmos que se ejecutan muy rápidamente, los tiempos no pudieron ser capturados, ya que el intervalo de la clase que se utilizó no fue lo suficientemente específica. Una vez más, recalcamos que sólo varía en las asignaciones, debido a que se necesita ejecutar la fórmula de tres tiempos en uno, y en otro no. Por esta pequeña diferencia podemos decir que el método CPM es más rápido y requiere menos operaciones.

Recomendaciones

- Con respecto al proyecto creemos que es bastante razonable, pero poco útil, no tiene mayor utilidad que realizarlo sin uso continuo con datos no muy reales, se pueden hacer mediciones empíricas para determinar una serie de comportamientos según el algoritmo y recorrido utilizado, para una tarea programada siguiente, sería mejor mejorar alguno existente de alguna empresa, para aplicación de negocios o para una manejador de proyectos real, quizás alguna especie de Project Manager.
- Con respecto al lenguaje podemos decir que bastante interesante y amigable para la programación orientada a objetos y estructuras de almacenamiento como arreglos, estructuras, variables entre otros, se recomienda la explicación más de clases propias del programa y su funcionamiento interno además de la diferencia entre este y sus sucesores C, Java y C# entre otros.
- Con respecto al tiempo, es lo suficiente para la realización de proyecto de esta índole con el tiempo que hay. Recomendaríamos que pudieran haber adelantos ya sea por semana, o a la mitad de la entrega, para ver el avance y si es lo más solicitado por el profesor a cargo de impartir el curso ya que esto podría corregir posibles problemas, además de los más frecuentes.
- Con respecto a la explicación en clase, podemos decir que si se explica, pero al parecer no queda muy claro; porque cuando se llega a realizar la práctica no se pueden realizar análisis muy acertadas, se recomienda que se aclare un poco más el tema desarrollado en la clase o que se realice más práctica.
- Con respecto horas de consulta recomendaríamos que se puedan realizar en un tipo foro en tiempo real a horas que la profesora pueda realizar las horas de consulta puesto que no siempre se puede ir a la oficina y no siempre se tiene mucho tiempo. Además a veces se llega a la oficina y esta con otros estudiantes, y muchas, ahí se pierde el tiempo de la consulta. Tiempo del cual no se aprovechó de la mejor manera.

Literatura citada

1. Estructuras de datos, Salvador Pozo Coronado. Consultado el 10 de abril de 2011. http://c.conclase.net/edd/.
    - Explicación general y superficial de varias estructuras de datos y sus respectivos usos en la programación.
2. Operator New. Consultado el 11 de abril de 2011. http://www.cplusplus.com/reference/std/new/operator%20new/
    - Explica el almacenamiento dinámico de instancias, utilizando el operador "new" del lenguaje.
3. Input/Output with files. Consultado el 12 de abril de 2011. http://www.cplusplus.com/doc/tutorial/files/
    - Manejo de archivos con la librería <fstream>, operaciones de posicionamiento, lectura y escritura.
4. El método PERT. Consultado el 24 de abril de 2011. http://ocw.upm.es/proyectos-de-ingenieria/proyectos-de-desarrollo-rural-i/Materiales-de-cada-tema/Planificacion-de-proyectos.-Metodo-PERT.pdf
    - Manejo de grafos para la administración de proyectos utilizando el algoritmo PERT.
5. Project Management FAQ. Consultado el 24 de abril de 2011. http://www.codeproject.com/KB/aspnet/ProjectManagementFAQ.aspx
    - Varios métodos de recorrido y cálculo de tiempos usando los algoritmos PERT y CPM, entre otros.

Anexos

1. Datos de prueba

Grafo 1, 10 actividades

`Grafo1 (10act).txt`

ID_ACT	*Predecesora(s)*	***Dur***
A	--	**3**
B	--	**2**
C	--	**4**
D	--	**3**
E	A, B	**2**
F	E	**4**
G	F	**2**
H	D	**1**
I	G, H	**2**
J	C, I	**4**

Grafo 2, 20 actividades

`Grafo2 (20act).txt`

Id_act	*Predecesora(s)*	*Duración*
A	--	**1**
B	--	**2**
C	A	**1**
D	C	**2**
E	B, C	**6**
F	D	**10**
G	F	**3**
H	G	**1**
I	F	**1**
J	E, H	**5**
K	I	**2**
L	F, J	**1**
M	F	**2**
N	L, M	**4**
O	G, J	**2**
P	O	**2**
Q	I, P	**1**
R	P	**7**
S	I, N	**7**
T	S	**3**

Grafo 3, 40 actividades

`Grafo3 (40act).txt`

Id_act	*Predecesora(s)*	*Duración*
A	--	**1**
B	A	**5**
C	A	**1**
D	A	**5**
E	C, D	**3**
F	E	**4**
G	D	**1**
H	G	**1**
I	H	**3**
J	F, I	**4**
K	F, I	**1**
L	J, K	**2**
M	L	**2**
N	L	**2**
O	D	**2**
P	B, M, N, O	**1**
Q	P	**6**
R	Q	**1**
S	I	**2**
T	B	**2**
U	A,B	**1**
V	T	**2**
X	U,V	**3**
Y	R,T,S	**5**
Z	A,B,C,D	**6**
AA	G,H	**8**
BB	O,P,Q	**6**
CC	AA	**1**
DD	CC	**4**
EE	BB	**2**
FF	AA,EE	**1**
GG	AA,Z	**9**
HH	BB.GG	**4**
II	X,Y,Z	**5**
JJ	EE,FF	**4**
KK	Q,T,R	**6**
LL	II,JJ	**3**
MM	A,B,C	**4**
NN	E,F,G	**2**
OO	**NN**	**7**