Teoría de Algoritmos I

Trabajo práctico n.º 2

El trabajo práctico consiste en las dos partes que se listan a continuación.

Lineamientos básicos:

- el trabajo se realizará en grupos de tres o cuatro personas.
- la fecha de entrega es el **lunes 14 de noviembre de 2016**. Se debe entregar en el horario de clase en papel (informe + `código en monoespacio`), más una entrega en digital de código (.zip) e informe (.pdf) al correo de entregas del curso: `tps.7529rw@gmail.com`.

Contenidos

Programación dinámica

En esta primera mitad del trabajo práctico se examinan los algoritmos de programación dinámica que resuelven los siguientes problemas:

- el problema de la mochila
- el problema del viajante de comercio

ambos algoritmos no polinómicos.

En un futuro trabajo práctico se comparará la complejidad de estos algoritmos con los algoritmos de aproximación correspondientes, esto es: que resuelven los mismos problemas de manera no exacta.

Por tanto, lo que se pide en esta parte es:

1. dos implementaciones por programación dinámica, una para cada problema,

siguiendo un formato de entrada específico

2. un breve informe con los tiempos de ejecución sobre distintas instancias del problema.

A continuación se especifica formalmente cada problema, formato de entrada, y lineamientos para el estudio de complejidad.

El problema de la mochila

Se pide implementar una solución por programación dinámica a la **versión 0-1 del problema de la mochila**. El capítulo 6.4 de Kleinberg y Tardos (2006) describe el problema y el algoritmo correspondiente. A continuación se incluye síntesis del problema.[1]

Dado un conjunto de elementos $S = \{1, 2, \ldots, n\}$ el problema consiste encontrar la sequencia $X = (x_1, x_2, \ldots, x_n)$ que maximice:

$$\sum_{i \in S} v_i x_i$$

con la restricción:

$$\sum_{i \in S} w_i x_i \leq W$$

donde $v_i, w_i, W \geq 0$ y $x_i \in \{0, 1\}$.

A diferencia del caso $v_i = w_i$, en esta versión la relación de recurrencia propuesta toma dos variables, el número de ítems a considerar y el máximo peso permitido w:

$$P(i, w) = \begin{cases} \text{si } w < w_i : & P(i-1, w) \\ \text{si no} : & \max \begin{bmatrix} P(i-1, w) \\ P(i-1, w - w_i) + v_i \end{bmatrix} \end{cases}$$

La solución X se puede derivar de la llamada inicial $P(n, W)$.

Archivos de prueba

Para verificar el algoritmo se pueden usar los archivos de prueba de David Pisinger. Los archivos están clasificados por tamaño y dificultad; cada archivo incluye 1000 instancias con su solución correspondiente.

Recomendaciones durante el desarrollo:

- usar inicialmente el archivo de coeficientes pequeños (smallcoeff.tgz, 142 MiB).
- usar los archivos con $n = 50$ y $R = 1000$: *knapPI_{0-9}_50_1000.csv*.

- una vez el verificada la corrección de la implementación, verificar que el tiempo de ejecución no se dispara para valores de n mayores.[2]
- usar para el informe el archivo de casos “difíciles” (hardinstances.tgz, 45 MiB).

Cada archivo CSV tiene 100 instancias de prueba del mismo tamaño y en el mismo formato, separadas por `-----`. El formato de cada instancia es, línea a línea:

- el nombre de la instancia, por ejemplo *knapPI_1_50_1000_1*
- el número de ítems en el problema, por ejemplo `n 500`
- la capacidad de la mochila W, por ejemplo `c 995`
- el valor óptimo conseguido por Pisinger (formato: `z 8373`) y, en la línea siguiente, el tiempo de resolución que obtuvo (formato: `time 0.01`)
- *n* líneas siguiendo el formato CSV: `num_item,valor,peso,x`, donde *x* es 1 o 0 según el ítem se incluyó o no en la solución, respectivamente.[3]
- la línea de fin de instancia, `-----`.

El problema del viajante de comercio

Se pide implementar el algoritmo de Bellman–Held–Karp para resolver la **versión asimétrica del problema del viajante de comercio**, esto es: dado un grafo dirigido, *completo* y pesado, encontrar un tour o ciclo hamiltoniano de costo mínimo que comience y termine en un vértice v_0 dado. (El costo se define como la suma de pesos de las aristas recorridas.)

Para simplificar la programación del algoritmo, consideraremos que una instancia del problema de tamaño N queda definida por el vértice inicial $0 < v_0 \leq N$ y una matriz de costos no negativos $C = N \times N$, por ejemplo:

$$C = \begin{pmatrix} 0 & 2 & 9 & 10 \\ 1 & 0 & 6 & 4 \\ 15 & 7 & 0 & 8 \\ 6 & 3 & 12 & 0 \end{pmatrix}$$

El ciclo hamiltoniano H se puede derivar de la llamada inicial $D(v_0, V - \{v_0\})$ con la recurrencia:

$$D(v, S) = \begin{cases} \text{si } S = \emptyset: & c_{vv_0} \\ \text{si no}: & \min\limits_{u \in S} \left[c_{vu} + D(u, S - \{u\}) \right] \end{cases}$$

Archivos de prueba

Para realizar pruebas del algoritmo se pueden usar, entre otros, las instancias recopiladas por John Burkardt. Se recomienda comenzar por los dos archivos más pequeños:

- p01.tsp (y su solución, p01_s.txt)
- fri26.tsp (incluye la solución en el apartado *TOUR_SECTION*)

En los archivos TSP se incluye la matriz de distancias en dos formatos posibles:

- *FULL_MATRIX*: la matriz de distancias al completo.
- *LOWER_DIAG_ROW*: la mitad inferior de la matriz de distancias, esto es, los valores que quedan por debajo de la diagonal, incluyendo esta. (En este caso, es claro que las distancias son simétricas entre cada par de ciudades..)

Para el informe se puede incluir una instancia de tamaño mayor: att48_d.txt (y su solución, att48_s.txt).

Flujo de redes

Problema[4]

Para complementar sus ingresos, el Ing. F. B. planea abrir una consultoría y actuar de intermediario entre diversas empresas y profesionales para llevar a cabo sus proyectos.

A través de sus contactos, F. B. obtuvo una lista de proyectos posibles $P = \{P_1, P_2, \ldots, P_m\}$ y se dispone ahora a contratar egresados de la Facultad para implementarlos.

El Ing. obviamente desea maximizar sus ganancias dadas las siguientes restricciones:

- cada proyecto requiere conocimientos específicos sobre una o más áreas de computación. En particular, cada proyecto P_i declara un conjunto de requisitos $R_i \subseteq A$; donde A representa el total de las n áreas que se enseñan durante la carrera ($|A| = n$).
- la contratación de un experto en un área determinada A_k $(0 < k \leq n)$ tiene un coste monetario de c_k.

  Los expertos, no obstante, son multi-tarea, y durante el periodo de contratación pueden contribuir a múltiples proyectos simultáneamente. En otras palabras, una vez se dispone de un experto en un área A_k no se necesita contratar a un segundo.
- cada proyecto realizado P_i proporciona una ganancia monetaria de g_i : si y solo si hay un experto contratado para cada una de las áreas en R_i. En caso contrario, no es posible comenzar el proyecto.

Así las cosas, el Ing. B. necesita saber los proyectos a aceptar y los expertos a contratar para maximizar sus ganancias netas (esto es, lo ganado con cada proyecto menos el monto de los salarios).

Consigna

Se pide:

1. Modelar el problema mediante un flujo de red.
2. Implementar un algoritmo que determine qué proyectos aceptar y qué expertos contratar.
3. Analizar en un breve informe la complejidad del algoritmo en términos del número de proyectos m, número de áreas n, y la cantidad total r de requisitos por parte de todos los proyectos $(r = \sum_{i=1}^{m} |R_i|)$.

Formato de entrada

El formato estándar para describir una instancia del problema será un archivo de $n + m + 2$ líneas, tal que:

```
N
M
c₁
c₂
...
g₁ r₁₁ r₁₂ …
g₂ r₂₁ r₂₂ …
...
```

Por ejemplo:

```
3
2
16
13
9
20 1 2
10 3
```

---

1. Kleinberg y Tardos primero describen una versión restringida en la que $v_i = w_i$, y en la segunda mitad del capítulo describen el algoritmo para $v_i \neq w_i$. ↩
2. Cada instancia de prueba en los archivos incluye el tiempo de ejecución que obtuvo

Pisinger durante la investigación. Todos las instancias de *smallcoeff.tgz* resolvieron en menos de 1 segundo. ↩

3. En el archivo README, se usa *p* para representar el valor (por 'profit') y *w* para el peso (por 'weight'). ↩

4. Como referencia bibliográfica para este ejercicio se recomienda el capítulo 7.11 de Kleinberg & Tardos. El texto que presentamos sigue la formulación original en Cormen et al., pp. 761-762, ej. 26-3. ↩