7529. Teoría de Algoritmos I
Trabajo Práctico 2

Ferreyra, Oscar, *Padrón Nro. 89563*
fferreyra38@gmail.com

Martin, Débora, *Padrón Nro. 90934*
demartin@fi.uba.ar

Eisner, Ariel, *Padrón Nro. 90697*
aeeisnerg@gmail.com

2do. Cuatrimestre de 2016

Índice

1. Programación dinamica

1.1. El problema de la mochila

Dado un conjunto de elementos S=1,2,... el problema consiste encontrar la secuencia X=() que maximice:

$$\sum_{i=1}^{10} t_i \sum_{i=S} v_i x_i$$

con la restricción:

$$\sum_{i=S} w_i x_i < W$$

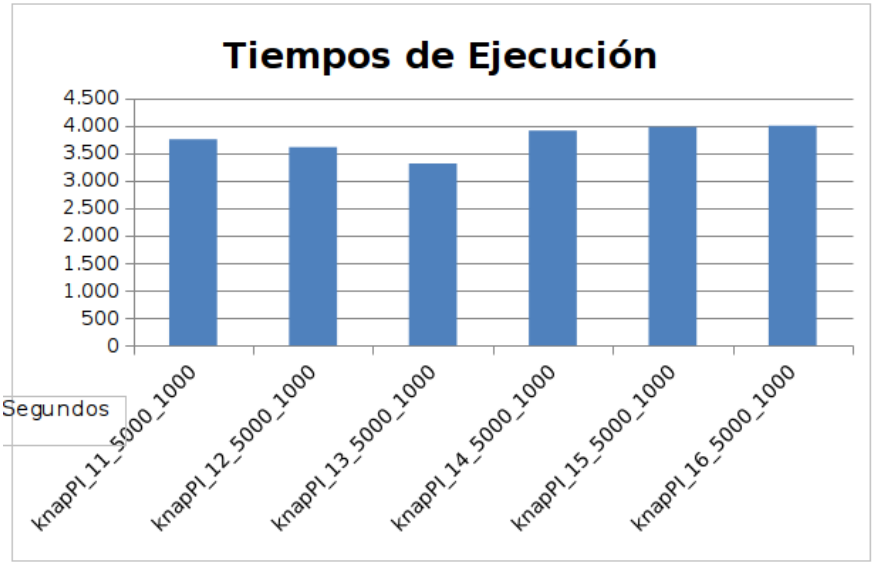
donde

$$v_i, w_i, W \geq 0$$

y

$$x_i \in \{0, 1\}.$$

Pruebas



Se seleccionó el lote de archivos con sufijo 5000_1000 del archivo **hardinstances.tgz** dado que los demás archivos generaban matrices demasiado grandes y el programa abortaba por falta de memoria.

1.2. El problema del viajante de comercio

Al implementar el algoritmo de Bellman-Held-Karp se notó que insume una gran cantidad de espacio en memoria, por lo que no fue posible realizar corridas para matrices de orden superior. El espacio en memoria que insume este algoritmo es del orden de $O(2^n n)$.

La complejidad del algoritmo se calcula en $O(2^n n^2)$ debido a las comparaciones realizadas y la adición de elementos a los distintos sets. A continuación se muestran los tiempos de ejecución dependiendo de la cantidad de vertices.

Cantidad de ciudades	Tiempo de ejecución (segundos)
15	16
17	299
19	7237
21	22157

Cuadro 1: Tiempo en función de la cantidad de ciudades para el algoritmo de Bellman-Held-Karp

2. Flujo de redes

Se tiene un conjunto de proyectos y cada uno, para llevarse a cabo depende de la contratación de determinados expertos, cada proyecto brinda un beneficio y cada contratación representa un costo. Dado una lista de proyectos con sus respectivos expertos, elegir aquellos proyectos que maximicen las ganancias.

2.1. Resolución

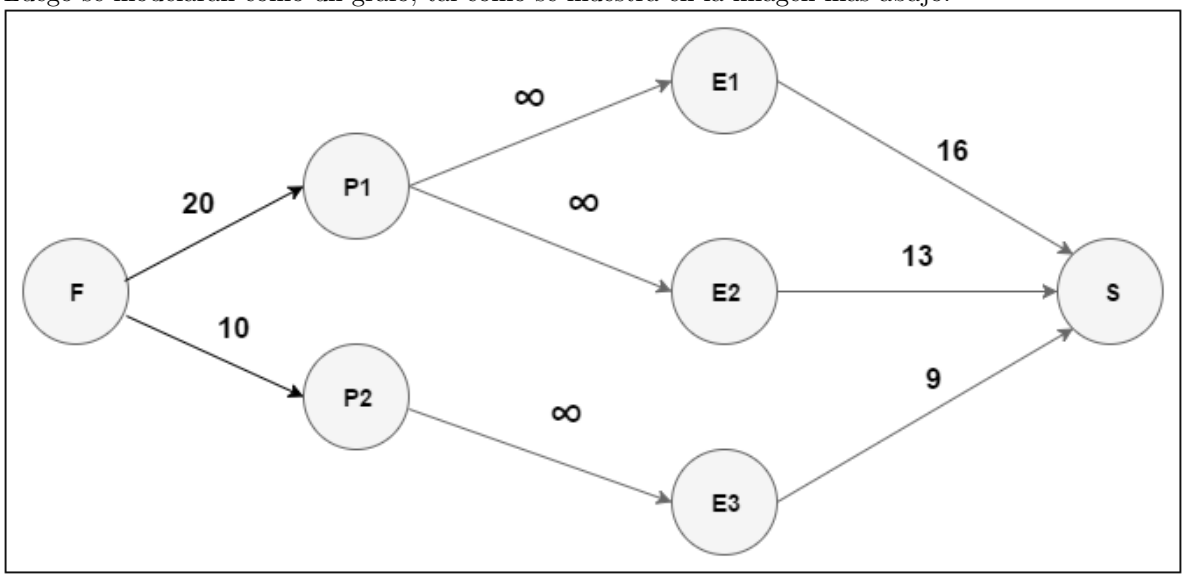
Si se considera cada proyecto y cada experto como un nodo en el flujo de red, es necesario también agregar un nodo fuente y un nodo sumidero para que pueda ser un flujo de red, de la fuente solamente salen los proyectos, y al sumidero solamente llegan los expertos.

Las aristas salientes de la fuente serán ponderadas con el beneficio del proyecto representando en el vértice adyacente, ídem para las aristas entrantes al sumidero, las aristas que conectan cada proyecto con su experto requerido serán ponderadas con ∞. Este modo de representar el problema fue sacado del libro Algorithm Design Cap 7.11

Los proyectos y expertos en el siguiente formato de entrada se modelarían según el siguiente flujo de red:

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

Luego se modelarán como un grafo, tal como se muestra en la imagen más abajo.



Img 2 Formato de entrada modelado mediante un flujo de red

2.2. Diseño del Algoritmo

Para resolver el ejercicio se implementó el algoritmo de Ford-Fulkerson. A continuación se describe el pseudocódigo del algoritmo.

```
Ford-Fulkerson(G,s,t) {
    for(cada arco (u,v) de E) {
        f[u,v]=0;
        f[v,u]=0;
    }
    while(exista un camino p desde s a t en la red residual Gf) {
        cf(p) =min{cf(u,v):(u,v) está sobre p};
```

```
        for(cada arco (u,v) en p) {
            f[u,v]=f[u,v] +cf(p);
            f[v,u]=-f[u,v];
        }
    }
}
```

- El grafo que representa el flujo de red es modelado mediante una lista de adyacencias, y el grafo residual sobre el cual se va calculando el flujo máximo se calcula sobre la misma lista de adyacencias.
- Para hallar un camino hacia al sumidero (camino de aumento) se utiliza una búsqueda BFS, la cual se basa en analizar el nodo visitado y luego cada uno de sus hijos.
- Luego de hallar el camino de aumento, se calcula el cuello de botella, y se actualizan las aristas pertenecientes a camino dicho.
- Cuando ya no existen caminos de aumentos, quiere decir que se el flujo de grafo es máximo por lo que se realizar el corte minimo, tomando el subgrafo que contiene a la fuente y se toman de ahí los nodos que corresponden a los proyectos, estos son los proyectos que de llevarse a cabo maximizan las ganancias.

2.3. Complejidad

La complejidad computacional del algoritmo de búsqueda del flujo máximo depende de dos aspectos:

1. **El algoritmo de búsqueda de los caminos de aumento:** el algoritmo de búsqueda de caminos de aumento es el BFS cuyo tiempo de ejecución es O(V + E).
2. **La cantidad de proyectos, expertos y requerimientos de cada proyecto:** por cada Proyecto, Requerimiento, Experto habrá un camino al sumidero

Por lo cual el algoritmo tendrá orden aproximado de O(nV + nE) siendo n la cantidad total de Proyectos, Requerimientos y Expertos en el grafo.

En la imagen 3 pueden apreciarse los Proyectos, Requerimientos y Expertos.

- P1 -> E1 > Sumidero
- P1 -> E2 > Sumidero
- P2 -> E3 > Sumidero

Habiendo un total de 3 caminos posibles hasta el sumidero.