Trabajo Práctico

75.29 Teoría de Algoritmos I

FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Profesores:

Víctor Podberezski

1° Cuatrimestre 2020

Integrantes:

NOMBRE Y APELLIDO	PADRON
Hernán Arroyo García	91257
Ruben Jimenez	92402
Federico Rodrigez Longhi	93336
Emanuel Condo	94773

1. El productor agropecuario

Problema

Se quiere elegir un subconjunto $S \in \{1, ..., N\}$ de cultivos mutuamente compatibles el cual sembrar durante los trimestres $[1, ..., T]$, tal que:

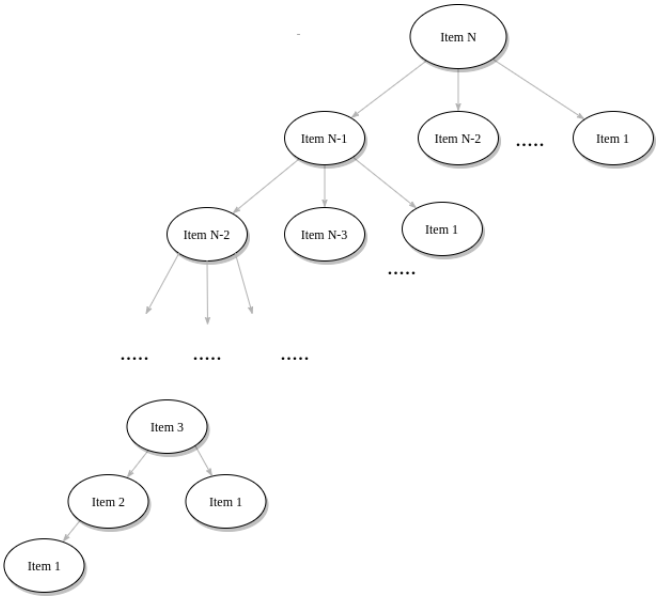
$\sum_{i \in S} S_i \rightarrow ganancia\, sea\, Máxima$

Propuesta

Primero leyendo el archivo de entrada se cargaran todos los item en memoria. Se arma un árbol de decisión, donde se tiene:

- Un nodo $i \in S$ por cada cultivo.
- Una arista $(i, j) \in SxS$, sí despues del cultivo i puedo sembrar j.
- Cada item tiene tantos nodos hijos como cultivos anteriores validos tenga.
- La altura del arbol es acotada por la cantidad maxima de periodos T.

Luego se evalúan todos los casos posibles para calcular el óptimo de cada item, se tendra que recorrer tantos caminos como nodos hijos tenga y a su vez nodos hijos de nodos hijos. Por lo que en el peor de los casos se tienen 2^(n-2) caminos, para n $>$ 1.

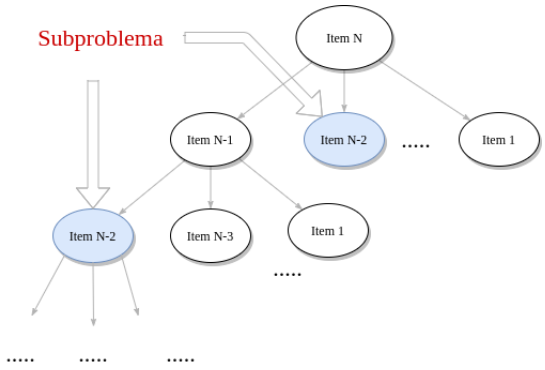


Simplificamos esto usando programación dinámica. Almacenamos las soluciones optimas de los subproblemas que se repiten en el árbol de decisión.

Para calcular el optimo, iteramos los items para obtener el valor óptimo de cada item basandose en el valor máximo que hay entre los óptimos que tienen los items que lo preceden. El resultado será una estructura que contiene el óptimo de cada item. Para encontrar el optimo que nos interesa, sólo resta elegir el máximo entre ellos.
Para mostrar todos los items del Optimo. Dado el Optimo de un item, se muestra el item y se va iterando sobre sus precedentes en busca del optimo a mostrar, así sucesivamente "h" veces (siendo h<=T) hasta que lleguemos a la raiz.

Subproblema del planteo

Como se mencionó anteriormente se tendrá un árbol por cada item, en el cual el cual los items que se encuentren en el trimestre más alto serán aquellos que tienen los árboles más profundos. En estos árboles tienen nodos en los cuales los óptimos se repiten, esos óptimos en lugar de recalcularlos se almacenarán para utilizarlos nuevamente.



Relación de recurrencia

A continuación se presenta la relación de recurrencia utilizada en la solución de programación dinámica.

$$OPT(i) = 0,\ i = 0$$
$$OPT(i) = profit[i] + Max\{OPT(previous[i])\},\ i > 0$$

En la relación de recurrencia se puede observar que el óptimo de cada item se basa en el óptimo de sus precedentes, por eso se define que el óptimo inicial es cero, para que el el óptimo previo al primer item tenga beneficio cero.

Pseudocodigo

```
Constructor items [1...n](O(n) lo desestimamos porque es la lectura del archivo)
Ordenar items por trimestre (O(nlog_2 n))
T = obtener cantidad maxima de trimestres (O(1))
Contruir estructura previous[1,...n] (lista de precedencias de cada item) (O(n.((n-1)+(n-2)+(n-3)+...+0)) < O(n^2))
Inicializar OPT [1...n] con 0, posOpt = 1
Para cada item en items: (O(n))
    optPrevious = 0
    Para cada itemPrevious en previous[posición item]:
(O((n-1)+(n-2)+(n-3)+...+0) < O(n))
        Si itemPrevious es el Opt: Actualizo optPrevious
    Fin Para
    OPT[item] = optPrevious + item.profit
    Si OPT[item] > OPT[posOpt]: Actualizo posOpt por posición item
Fin Para
Imprimir "Ganancia Maxima: " OPT[posOpt]
Mientras previous[posOpt] != null: (O(h))
    Imprimir item[posOpt]
    Para cada itemPrevious en previous[posicionOpt]
(O((n-1)+(n-2)+(n-3)+...+0) < O(n))
        Si itemPrevious es el Opt: Actualizo posOpt
    Fin Para
Fin Mientras
```

Complejidad temporal y espacial

Considerando que se tienen n items a sembrar y que existen t trimestres con t $<$ n.

- Para la evaluación de la complejidad temporal se debe considerar que inicialmente se realiza un ordenamiento para obtener el valor de trimestre más alto, esto se realiza con un costo de $O(nlog_2 n)$. Seguido se arma la estructura previus (lista de precedencias de cultivos), recorremos los iteme y por cada uno se toman los que lo preceden, n iteraciones y la serie $((n-1)+(n-2)+(n-3)+...+0))$, todo esto acotado por $O(n^2)$. Luego se recorre cada item, para ir calculando el Opt en cada de cada uno de estos, dentro se tiene un segundo loop donde se busca el Opt en su periodo anterior, pero para los items que lo preceden, los items que lo preceden se comportan como la serie $((n-1)+(n-2)+(n-3)+...+0))$ esto esta acotado por $O(n)$, entonces la complejidad es de $O(n^2)$. Para mostrar tenemos que dado la posicion de un item optimo tenemos que hacer h saltos (h$<$=T) y por cada salto tenemos la serie de previous de complejidad$O(n)$, entonces la complejidad es de$O(n.T)$. Sumando los

costos se tiene $O(nlog_2 n) + O(n^2) + O(n^2) + O(nT)$, lo cual está acotado por $O(n^2)$.

- Para la evaluación de la complejidad espacial se utilizan tres estructura de datos una para el almacenamiento de item, siendo este de tamaño n, otra para almacenar la lista precedencias de los items y Opt para guardar el valor optimo para cada item, por lo tanto se tiene un costo de $O(n)$, $O(k.n)$ y $O(n)$ respectivamente. Si sumamos todos obtenemos que O(n+kn+n) está acotado por O(n). Por lo tanto tiene una complejidad espacial lineal.

2 Exportación Minera

Resolución del problema

Primero vamos a definir algunas cosas:

1. Tenemos un conjunto R de n cantidad de regiones.
2. Cada región i tiene un costo ci y una ganancia gi.
3. Vamos a definir pi como el retorno económico de la región i. La misma se calcula pi = gi - ci.
4. **Ganancia de A**: Sea A un conjunto de regiones i, llamamos Ganancia de A a:

$$G(A) = \sum_{i \in A} pi$$

5. Podemos calcular el tope de ganancia $C = \sum_{\forall i / p_i > 0} pi$. Esta será la ganancia máxima que podríamos llegar a tener, si no hubieran restricciones de precedencias.
6. Además tenemos restricciones para explotar ciertas regiones. Vamos a decir que si la región i necesita que se explote la región j para acceder a ella entonces **la región j precede a la región i**.

Con esto podemos construir un grafo de precedencias, por ejemplo:

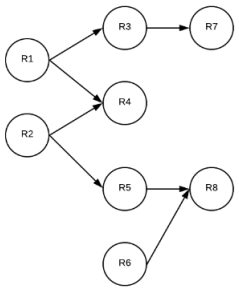


Figura 1: Grafo de precedencias

Vemos en este caso, por ejemplo la región 1 y 2 precede a la región 4, la región 3 precede a la 7.
Con esta información sabemos entonces varias cosas:

1. La solución de este problema es un conjunto A que contiene las regiones que se van a explotar.

2. Si una región es seleccionada dentro de la solución, entonces todas sus precedentes tienen que estar incluidas en la solución.

3. La ganancia G(A) de cualquier conjunto $A \subset R$ nunca va a superar al tope de ganancia C.

Entonces para encontrar la solución a este problema vamos a hacer lo siguiente:

1. Vamos a construir un grafo dirigido de la siguiente manera:

   - Agregamos un nodo s (como fuente) y un nodo t (como sumidero).
   - Agregamos un nodo por cada región.
   - Por cada nodo i con pi $>$ 0 agregamos un eje s-i con capacidad pi.
   - Por cada nodo i con pi $<$ 0 agregamos un eje i-t con capacidad -pi.
   - Por cada relación j precede a i agregamos un eje i-j con capacidad C+1.

   Como ejemplo supongamos que tenemos el grafo de la Figura 1 con los siguientes valores:

Region	1	2	3	4	5	6	7	8
$p_i = g_i - c_i$	5	-5	-3	1	-7	2	10	5

   Tope de Ganancia: C = 23.

   Con estos valores construiriamos el siguiente grafo:

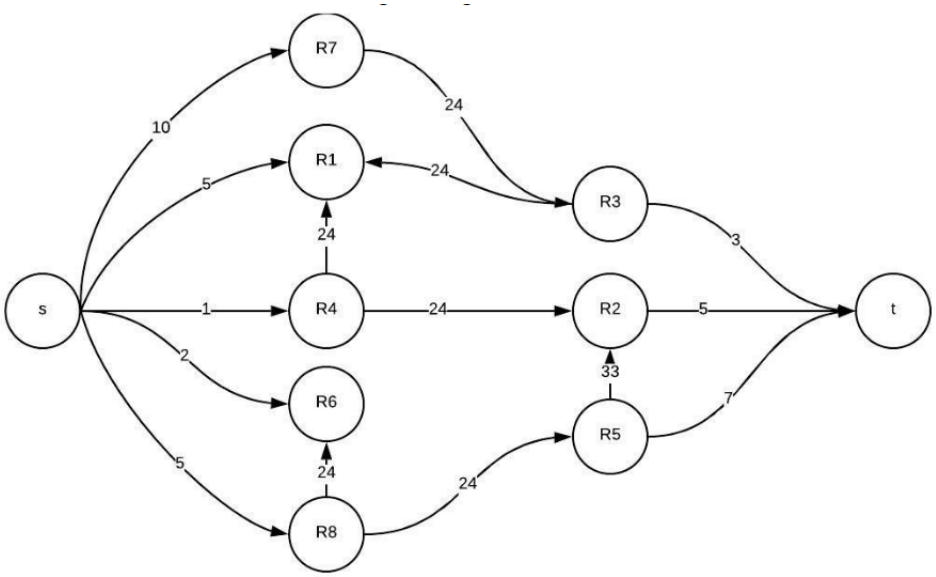


Figura 2: Grafo de reducción del problema

2. Obtenemos el corte mínimo

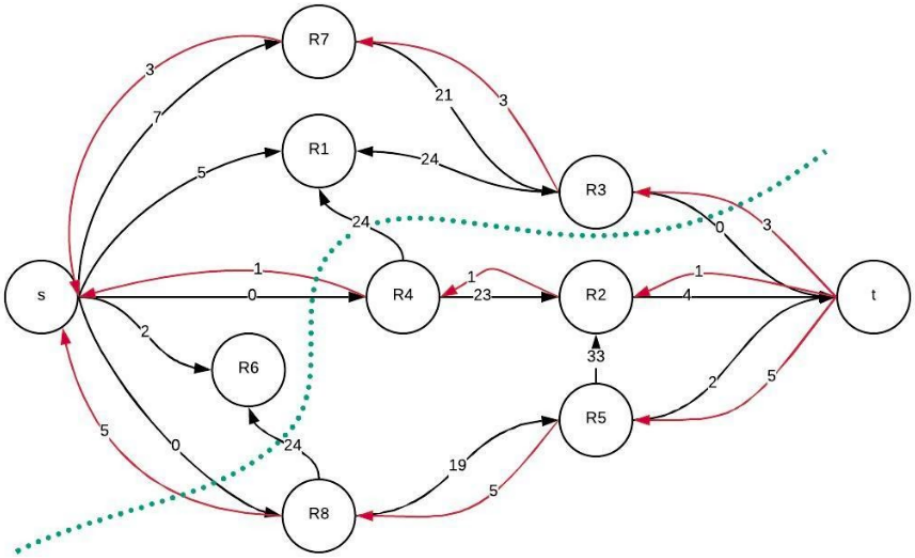


Figura 3: Corte mínimo

3. Si llamamos C(A', B') al corte mínimo: Las regiones en A' - {s} corresponden a las regiones a explotar para maximizar la ganancia.

   En este ejemplo obtenemos el corte mínimo dejando dos conjuntos A' = {s, R1, R3, R6, R7} y B' = {R2, R4, R5, R8, t}.

   Entonces, A' - {s} nos dará la ganancia máxima, esto es la suma del retorno económico de las regiones {R1, R3, R6, R7} que es igual a 14.

   Podemos hacer algunas verificaciones rápidas, por ejemplo:

   - $14 <$ C $= 23$, que es el tope de ganancia.
   - $14 > \sum_{\forall i} pi = 8$, ganancia que obtuvieramanos por explotar todas las regiones .
   - C - G(A) = 23 - 14 = 9 corresponde a c(A', B') = 9.

Justificación solución óptima

El problema de encontrar las regiones a explotar maximizando la ganancia, la redujimos a un problema de flujo.
A continuación probamos que calculando el flujo máximo y corte mínimo, obtenemos las regiones que maximizan la ganancia.
Sea C(A',B') el corte min.
Si A son las regiones en el corte mínimo, entonces:

- A' = A U {s}
- B' = (R-A) U {t}

Si R cumple las restricciones de precedencia, entonces:

- El corte min C(A',B') no tendrá ejes i-j que crucen de A' a B'.

  Ya que, la capacidad de los ejes i-j es mayor al corte c({s}, R U {t}) (porque estas capacidades las definimos con un valor por arriba del tope de ganancia máxima).

Sabemos que hay 3 tipo de ejes entre A' y B':

- Los que representan precedencias "E"

  Dijimos que los ejes "E" no están en el corte min

- Los que salen de "s" (Regiones con ganancia positiva)

  $$\sum_{i \notin R / p_i > 0} p_i$$

- Los que ingresan a "t" (Regiones con ganancia negativa)

  $$\sum_{i \in R / p_i < 0} -p_i$$

Por lo tanto, podemos calcular:

$$c(A', B') = \sum_{i \in R / p_i < 0} -p_i + \sum_{i \notin R / p_i > 0} p_i$$

Usando tope máximo de ganancia:

$$c = \sum_{i \in R / p_i > 0} p_i$$

Podemos reescribir el corte min:

$$c(A', B') = \sum_{i \in R / p_i < 0} -p_i + (c - \sum_{i \in R / p_i > 0} p_i)$$

$$c(A', B') = c - \sum_{i \in R} p_i)$$

Como R es una selección factible:

$$Ganancia(A) = \sum_{i \in R} p_i$$

Tenemos que:

$$c(A', B') = c - Ganancia(A)$$

Como c es constante, entonces al calcular el corte min, estoy maximizando la ganancia. Entonces al calcular el flujo máximo y obtener el corte min, estoy resolviendo el problema de maximizar la ganancia de las regiones a explotar.

Complejidad temporal y espacial

- La complejidad temporal va a depender del algoritmo utilizado para resolver el camino mínimo. Si utilizamos **Ford-Fulkerson** entonces la complejidad temporal será O(mC) donde m es la cantidad de ejes del grafo.

Además tenemos que agregar la complejidad de obtener el corte mínimo, esto se puede calcular en O(m). Analizando un poco nuestro problema, estas complejidades están calculadas para el grafo construido. ¿Cuantos ejes tenemos? En principio 1 por cada nodo y luego sumamos uno por cada precedencia, es decir que para el grafo G' que construimos para la resolución del problema tenemos que m' = m + n (la cantidad de nodos más la cantidad de ejes originales). En el caso de Ford-Fulkerson el valor de C está dado por la suma de todos los flujos que salen de s (fuente). En el caso de este problema este valor de C corresponde a nuestro tope de ganancia (que, paradójicamente, también notamos C).

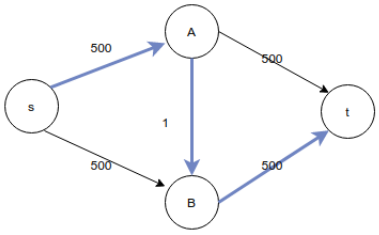
Finalmente tenemos que la complejidad temporal utilizando Ford-Fulkerson es de O((m+n)C).

- Para la complejidad espacial tenemos que tener en memoria los ejes con sus capacidades recordemos que la cantidad de ejes del grafo G' es m+n. Además para construir el grafo residual necesitamos a lo sumo 2(m+n) ejes. Por lo tanto la complejidad espacial para este algoritmo es de O(m+n).

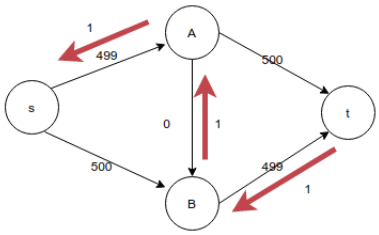
¿Es eficiente la solución?

El algoritmo usado **Ford Fulkerson** (si bien la complejidad temporal es pseudo polinómica de O(c(m+n)) tenemos que la elección del camino de aumento no esta definida (no hay un algoritmo), por eso se acota por el peor caso (incrementos de una unidad en el flujo hasta llegar a "c"). Por esta razón podemos decir que el algoritmo es aceptable, pero hay casos donde puede tardar mucho, donde tendremos que usar otras opciones. A continuacion vemos un caso patologico, el cual se podría mejorar.

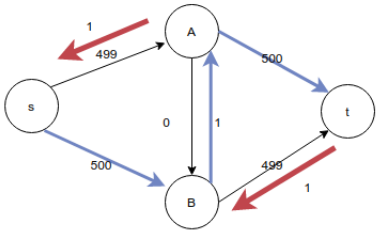
- Vemos que en el peor de los casos la elección del camino de aumento produce incrementos en 1 del flujo, esto se repite 1000 veces.
- Pero la mejor solución es hacer dos saltos de flujos de 500.



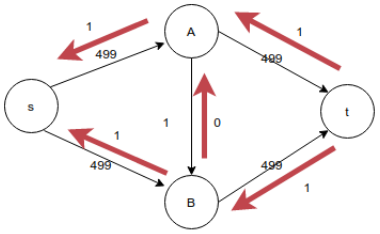
Grafo residual Gf.
Camino de aumento s,A,B,t (flujo min: 1)



Grafo residual Gf.



Grafo residual Gf.
Camino de aumento s,B, A,t (flujo min: 1)



Grafo residual Gf.

Hay algoritmos que mejoran el calculo de Flujo maximo:

- Algoritmo de Edmonds-Karp ($O(V^2E)$), siendo E=cantidad de ejes, V=cantidad de nodos.