

TDA (TB024 / 75.29 / 95.06): Parcial - 19 de Mayo 2025

Padrón: 106768

Alumno: Santiago Nahuel Ruiz Sylva

Hojas: 6

Considere en la resolución de los ejercicios incluir (cuando corresponda): Explicación de la solución, relación de recurrencia, pseudocódigo, análisis de complejidad temporal y espacial, análisis de optimalidad y un breve ejemplo para clarificar su idea. Contemple utilizar las definiciones y propiedades de cada metodología de resolución

Firma



Greedy: Se planeó un nuevo sendero en una montaña. Para ser habilitado requiere la construcción de refugios con guardaparques distribuidos sobre el mismo. Existen "n" proyectos. Cada proyecto propone construir un refugio en cierta ubicación sobre el sendero. Cada refugio "r_i" cubre una cantidad de kilómetros k_a_i antes y k_d_i después de donde está ubicado. Nos solicitan determinar la menor cantidad posible de refugios a seleccionar para cubrir todo el camino. Proponga un algoritmo greedy eficiente que lo resuelva.

Programación dinámica: Juan desea comprar perfumes para revender entre sus conocidos. Puede llevar "k" kilos antes de superar el límite de su bolso. Por cada uno de los "n" perfumes que le interesan conoce el precio de compra y el de reventa. Todos los perfumes cuentan con un peso unitario y stock ilimitado. Ayudar a Juan a maximizar la ganancia a obtener. Solucione el problema utilizando programación dinámica.

Redes de Flujo: Una red de satélites se construyó para permitir la comunicación entre una nave espacial y la tierra. Ciertos satélites pueden intercambiar mensajes entre otros. Algunos con la tierra y otros con la nave espacial. Contamos con la red y nos piden que midamos su robustez: cuantos

satélites en el peor de los casos se pueden romper que dejan incomunicada la nave con la tierra? ¿Cuáles?

Clases de Complejidad: Se realizará un nuevo festival de música y nos contratarán para definir a los artistas convocados. Existen "n" posibles artistas. Contamos con un presupuesto de "p" pesos. Cada uno de los artistas tiene honorarios de contratación. Entre ciertos artistas existen relaciones de amistad. Deseamos contratar "r" artistas sin superar el presupuesto y que al menos entre ellos existan "a" relaciones de amistad para fomentar las colaboraciones.. Demostrar que el problema es NP-C (utilizar el problema del Clique)

Fuerza Bruta: Contamos con una matriz de $n \times n$. En cada celda i encontramos un valor $v_i > 0$ asociado. Queremos seleccionar un subconjunto de celdas cuya suma sea la máxima posible. Pero no podemos seleccionar 2 celdas que sean adyacentes entre sí. Resolver utilizando branch and bound.

~~Problema de complejidad~~Cases de complejidad

Llámemos el problema de los artistas como P.A y queremos demostrar que es NP-C. Por ello debemos, demostrar que es NP y NP-H, para el primer caso debemos dar un certificado eficiente el cual ejecute en tiempo polinomial dentro un instante del problema y una supuesta solución.

o) Cert(artistas , presupuesto, emistados , r , e , s solución)

S: $\text{solucion} \neq r \Rightarrow O(1)$

return False

Porz zt in solucion $O(r)$

S: zt not in artistas $O(1)$

return False

$p = 0$

Porz zt in solucion $O(r)$

$p + \text{amt. honores}$

S: $p \geq$ presupuesto

return False

$\text{emistados} = 0$

Porz zt1 en solucion $O(r)$

Porz zt2 en solucion $O(r)$

S: $\text{zt1} = \text{zt2}$

continue

S: $(\text{zt1}, \text{zt2})$ en $\text{emistados} \circ (\text{zt2}, \text{zt1})$ en emistados

$\text{emistados} += 1$

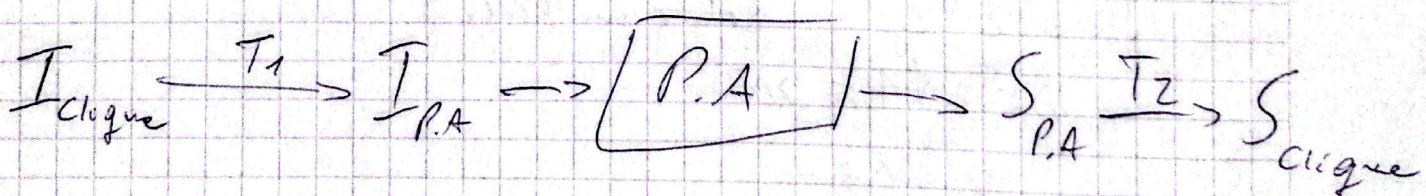
return $\text{emistados} \geq 2$

- Consideremos a solución como un conjunto.
- " " " a los vértices como un conjunto
- " " " a los lados como una lista.

Respecto a la complejidad podemos observar que la mayor es $O(r + r^2 \cdot m)$ con 'r' la cantidad de vértices en la solución y 'm' la cantidad de aristas, por lo tanto $P.A \in NP$ por haber brindado un certificado eficiente que ejecuta en $O(r^2 \cdot m)$ y por ende polinomial.

Por demostrar que pertenece a $NP\text{-H}$ debemos reducir un problema $NP\text{-C}$ conocido. Queremos demostrar en este caso que Clique $\leq_p P.A$, es decir, que Clique es reducible polynomialmente a P.A.

Dada una instancia de Clique compuesta por $G(V, E)$ y un K entero buscamos resolver una transformación polynomial a una instancia de P.A para que este último obtenga la caja negra que resuelve P.A y posteriormente transformar la solución de la instancia producida de P.A en la solución de la instancia original de Clique.



Nuestra instrucción de P.A regresa los vertices , el presupuesto, las relaciones de simetría, $r \neq \infty$.

Transformación $T_1(G, K)$

$$\text{vertices} = []$$

$$\text{simétricos} = []$$

$$\text{Presupuesto } \checkmark \text{ en } V(G) \quad]O(v)$$

$$\text{vertices}.add(v)$$

$$\text{Presupuesto } v \text{ en } G\text{-adyacentes}(v) =]O(e)$$

$$\text{simétricos}.add((v, w))$$

$$r = K \quad]O(1)$$

$$p = K \quad]O(1)$$

$$z = \frac{K(K-1)}{2} \quad]O(1)$$

return P.A(vertices, simétricos, r, p, z)

La complejidad de la transformación T_1 es $O(vn)$
 \forall por la forma polinomial, el orden V de G es
 un vertice , mientras que estos que los van ser
 agregados, la cantidad de vértices para construir es igual
 al presupuesto, y la cantidad de simétricos es dada
 por $K(K-1)/2$ que es un clúster de tamaño
 K hay $K(K-1)/2$ simétricos.

Si la ejecución de P.A retorna true entonces
 necesariamente hay un clúster de tamaño K en G
 \exists que los r^2 vértices de la subconjunto comparten
 tal clúster de modo que la cantidad de simétricos
 en el mismo es igual a $r(r-1)/2 = z$ calculado en T_1 .

La transformación vertebral se tratará de recorrer
esos estados de la solución y que cumplan el
K-clique de la instancia inicial.

Transformación 2 (S)

Por cada $s \in S \setminus O(r)$
res.odd(S)

return s

La complejidad es entonces polinomial y finalmente
hemos demostrado que Clique \leq_p P.A, por ende
P.A \in NP-C.

Prog. Díjávez

Juan puede elegir simultáneamente un perfume a su bolso
mientras no excede el peso K tolerado por el mismo. Por
cada perfume puede calcular si genera menor peso
la operación ~~compra~~^{compras} - compra - revata. Lo que
hacemos es por cada perfume calcular si genera
menor peso la operación indicada, y también tener en
cuenta el peso contenido, de este forma confeccionamos
un nuevo arreglo donde en cada posición tendremos una
tupla de las formas (perfume, generar, peso). Por
saber cuáles elegir e ir memorizando las elecciones
puedes comprobar un número de $(n+1) \times (K+1)$ inserciones
en cero en todos sus posiciones, cada columna representará
un peso máximo aceptado (es decir de 0 a K) y cada fila
un perfume que estrene, ordenando. Al echar a elegir
un nuevo perfume al bolso o no debemos considerar la

NOTA

generar que hay en el bolso el agregar este perfume más la generación del mismo y la generación por el mismo K multiplicado por el perfume anterior. Tenemos entonces la siguiente ecuación de recurrencia:

$\text{Opt}[i][j] = \max(\text{Opt}[i-1][j-p_i] + g_i, \text{Opt}[i-1][j])$

Donde i es el elemento actual (perfume que estoy excluyendo), j el peso actual que estoy evaluando, p_i el peso del perfume i y g_i la generación del mismo. Claro que si $p_i > j$ entonces no lo puedo agregar y entonces considero el efecto reforzado por el perfume anterior en tal j.

Bolsa Perfumes (perfumes, K)

- Ser p el arreglo de perfumes de la forma

(perfume, p_i , g_i) de largo n \rightarrow $\text{JO}(n)$

- Ser Opt la matriz de $(n+1) \times (m+1)$ inicializada en ceros $\rightarrow \text{JO}(n \cdot K)$

Por i de 1 a $n+1$

Por j de 1 a $K+1$

$$\text{perfume_actual} = p[i-1]$$

$$p_i = \text{perfume_actual}[\text{peso}]$$

$$g_i = \text{perfume_actual}[\text{generación}]$$

Si $p_i > j$:

$$\text{Opt}[i][j] = \text{Opt}[i-1][j]$$

Sino:

$$\text{Opt}[i][j] = \max(\text{Opt}[i-1][j-p_i] + g_i, \text{Opt}[i-1][j])$$

$i = n$
 $j = k$
 $res = []$

while $i > 0$ and $j > 0$

~~if $Opt[i:j] := Opt[i:j-1, p_i] + q_i$~~
~~res.append($p[i-1]$)~~
 ~~$j -= p_i$~~
~~Sino~~

$p_i = p[i-1][q_{i-1}]$

~~$S_i: Opt[i-1:j] \neq Opt[i:j]$~~

~~res.append($p[i-1]$)~~

~~$j -= p_i$~~

~~Sino~~

~~$i -= 1$~~

return res

De este forma el enfoque en la encauz de recurrencia si agregas el elemento me convierte mas que no hacerlo y (si lo hago) considera la generacion optima por el peso restante mas la generacion del perfume enunciado es que considera siempre el caso de volver a agregar el mismo perfume i.

La complejidad temporal es entonces $O(n \cdot K)$ dado que la mima se debe el enunciado de la mima de optimo, su recorrido por el calculo de los mismos y su iteracion "reversa" para reconstruir las elecciones. (el enunciado del enfoque p es $O(n)$ ya que se realizan 2 operaciones constantes como el calculo de la generacion y el recorrer el enunciado de perfumes original).

La complejidad de este algoritmo es $O(nk)$. Debido a que tiene la matriz en memoria y el largo del algoritmo (sabemos también tamaños el arreglo P que es $O(n)$ en memoria y el arreglo resultado).

Redes de flujo

Dede que en la red tenemos que tanto la nave como las tierra se comunican con satélites (los satélites tienen info. de la nave, de la tierra o de otros satélites, y viceversa) vamos a poder de la red que nos dany realizar ciertas modificaciones para que la misma sea una red de flujo y la cual podemos aplicar Ford Fulkerson.

Armar Red (G)

- Red = un grafo dirigido pesado igual a G (copy)
- Agrego a Red un nodo "S"
- Agrego a Red un nodo "T"
- Conecto el nodo "S" al nodo "nave" con una arista de peso igual a los 2dyz centes de la nave (en otros satélites)
- Conecto el nodo "Tierra" al nodo "T" con una arista de peso igual a los entradas de "Tierra"
- flujo-max, $Gr = \text{Ford Fulkerson}(\text{Red}, "S", "T")$

C = Encuentra Cente Maximo ($Gr, "S", "T"$)

return "Gr" si el peso entre si se rompen "+ flujo-max" "satélites" entonces se quedan incomunicados, ellos son " \star " C

* A cada arista en Red le pongo peso 1

Encontrar Corte Minimo (G , s , t) (G)

Sea V el conjunto de vértices de la fuente " s " obstante, medir el BFS en G en gráfico residual.

Sea T el conjunto de vértices del sunro "f1" obstante, medir el $V-U$.

Por tanto v es un vértice de T .

Por cada w adyacente a v en G el gráfico augmental

Si w pertenece a T

encontrar_corte_min.odd((v, w))

return encontrar_corte_min

FindFuller (G , s , t)

Introducción $f(e) = 0 \nabla e \in E(G)$, $f(s-t) = 0$

Creo el gráfico G' copia de G

Muestro existe un camino de sumo de $s-t$ en G'

Sea p dicho camino de sumo

Sea w el cuello de botella de p

$f(s-t) + = w$

Aumentar los estados de pertenencia de w

Aumentar G'

return $f(s-t)$, G'

En nuestra red todos los caminos de sumo tendrán
cuello de botella 1 debido a que lo estable como por
toda parte antes de crear " S " y " T ", por lo tanto los
~~cambios de sombra~~ cambios de sombra existen en el centro
mismo también aparecerá 1 y los pares de vértices
que los comparten serán los satélites que si se rompen
desaparecerán en las comunicaciones disponibles entre nube y tierra.

Si contrariamente se rompen ~~alguna~~ de los arcos verticales que compone cada parte del corte mínimo, entonces la tierra y 1/2 nave quedan incomunicadas.

Complejidad temporal

- Creamos la nave real en $O(V+E)$
- Agregamos los nuevos nodos "s" y "t" en $O(1)$
- Agregamos los pesos a los arcos de "s" y "t" en $O(V)$
- Ejecutamos Ford Fulkerson en $O(E \cdot |X|)$ sacando ~~X~~ los seteables en los arcos entre 1/2 parte "nave" \rightarrow "seteable"
- Que se $E \cdot |X| = 1/2$ cantidad de caminos de sumo menor entre s y t y por ende $|X| =$ flujo mismo
- Encuentra el corte mínimo corta de rebajar en BFS desde "s" para encontrar V , luego T , la complejidad de BFS es $O(V+E)$
- Finalmente la complejidad temporal es $O(V+E+E \cdot |X|)$.

Complejidad espacial

- Coste del grafo reforzado en función del original, V, T y los arcos del corte mínimo, por ende $O(V+E)$.

Greedy

Podemos comprobar procesando los refugios de forma que obsequiamos en un arreglo n elementos de 1/2 forma ~~(K int, K fin)~~ donde calculamos en orden como K_{int} como ubicacion-ri - ~~ultimo~~ $\leftarrow K_{fin}$ como ubicacion-ri + K_{el-i} de modo que K_{int} es el kilometro

~~HallaCobertura (refugios)~~

~~• Sección de elegir de refugios de la forma
(r[i].Kd[i] <= r[i].Kf[i])~~

$r = \text{procesar_refugios}(\text{refugios})$ $\mathcal{O}(n)$

Ordeno r por K_{ini} ascendente $\mathcal{O}(n \cdot \log n)$

Si $r[0][K_{\text{ini}}] != \emptyset$ $\mathcal{O}(1)$

return false

res.append($r[0]$) $\mathcal{O}(1)$

Por i desde $1 \leq i \leq n$ $\mathcal{O}(n)$

~~BÚSQUEDA DE UN ELEMENTO~~

Ser ult_ref el último elemento de res $\mathcal{O}(1)$

Si $r[i][K_{\text{ini}}] > \text{ult_ref}[K_{\text{fin}}]$ $\mathcal{O}(1)$

return false

Si $\text{res}[i][K_{\text{ini}}] \leq \text{ult_ref}[K_{\text{fin}}] \text{ y } \text{res}[i][K_{\text{fin}}] > \text{ult_ref}[K_{\text{fin}}]$ $\mathcal{O}(1)$

$\text{res}[i][K_{\text{fin}}] = \text{ult_ref}[K_{\text{fin}}]$

res.append($r[i]$)

return res

$\text{procesar_refugios}(\text{err})$

$\text{res} = []$

Por orden r en err $\mathcal{O}(m)$

$\text{res.append}(\text{ubicacion} - r.Kz[i], r.ubicacion + r.Kde[i])$ $\mathcal{O}(m)$

return res

La complejidad temporal Análisis es $O(n \cdot \log n)$ debido que dará los $O(n)$, la mitad se debe al ordenamiento. La complejidad espacial es $O(n)$ por la creación del arreglo procesado, en este de que las propuestas se sitúan en orden como donde surgen y otra vez, el arreglo resultante será de largo ≈ 6 sinn. n.