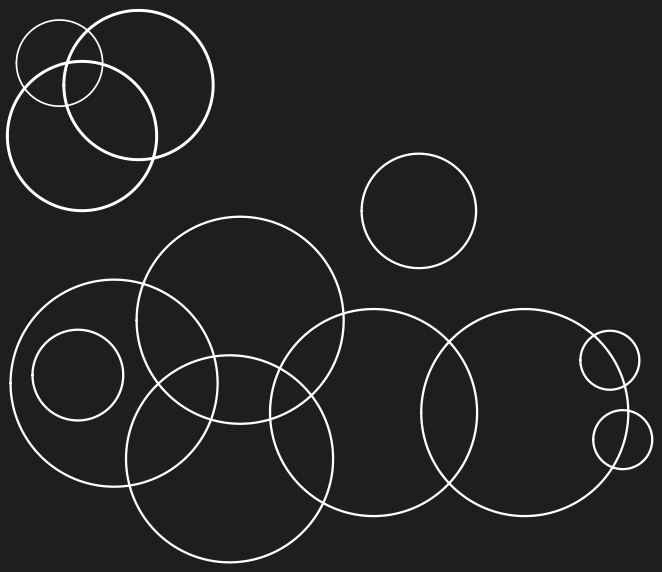
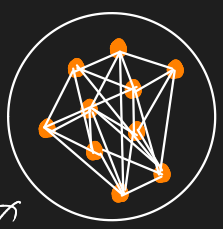


Caso borde.

$Q_{min} = \{a_1\} \Rightarrow Q_{min} \notin E_i$ (sin amigos)



Clique (grafo denso)

$V = \{\text{actores}: 1 \text{ to } n\}$

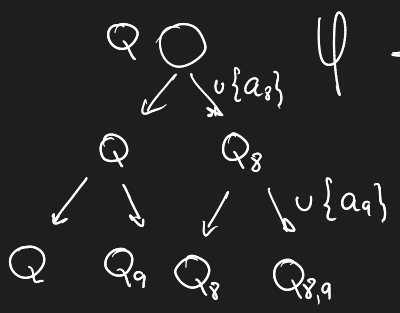
$E = \{\{1,2\}, \{1,3\}, \ldots, \{7,n\}\}$ (Notación "7n")

$p(a): V \longrightarrow \mathbb{N}_{>0}$

- Todas las soluciones parciales del árbol de backtracking son cliques de G.

  ↳ **"No puede haber nodos que sean respuestas no válidas: todas deben ser filtradas en el algoritmo de BT"**

- Cada nodo del árbol de bactracking representa un conjunto de soluciones (cliques) $\mathcal{Q}$ que se obtienen al visitar las hojas de su subárbol; todas las cliques de $\mathcal{Q}$ contienen a la solución parcial (clique) Q. Intuitivamente, en el proceso de extensión ya se tomó la decisión de que cada actor de Q esté en las soluciones que se generan a partir de este nodo.



$\Psi = \{Q, Q_8, Q_{8,9}, Q_9\}$

$Q \subset Q_i \quad \forall Q_i \in \Psi$

con $Q \triangle Q_i = \{a_i\}$ ↳ Actor agregado.

- Además de Q, cada nodo del árbol de backtracking tiene asociado un conjunto de actores K que es disjunto a Q. Intuitivamente, K contiene los actores sobre los que aún no se tomó ninguna decisión en el proceso de extensión. A partir de ahora, hablamos del nodo (Q, K) para referirnos a aquel nodo correspondiente a una solución parcial Q cuyo conjunto asociado es K.

$K = \{$ actores aún no considerados $\}$

- Algo. comienza con $K = V$ (todos)
  termina con $K = \emptyset$
- En cada llamada

$$K := V \setminus \{a_i\}$$

- En la 1° llamada $(K = V)$ no!

$\cup \{v_i\}$ con $v_i \in K$

2 posibles hijos (sucesores)

- En la 2°, ya elegí uno (#K veces!)
  quedan #K − 1 por probar en cada uno de los #K nodos anteriores.

⊕ Los nodos pueden describirse como

(Q, K)

Donde

Q contiene los actores del Clique actual

y K los actores que quedan por analizar

en el proc. de Backtracking (BT).

Obs: Dados el conjunto de actores V, los pares de amistades E, algún clique Q, y un conjunto K de actores, entonces con decir "(Q, K)" estoy describiendo completamente al nodo correspondiente, con conjunto asociado φ.

- Cada nodo del árbol de bactracking representa un conjunto de soluciones (cliques) $\mathcal{Q}$ que se obtienen al visitar las hojas de su subárbol; todas las cliques de $\mathcal{Q}$ contienen a la solución parcial (clique) Q. Intuitivamente, en el proceso de extensión ya se tomó la decisión de que cada actor de Q esté en las soluciones que se generan a partir de este nodo.

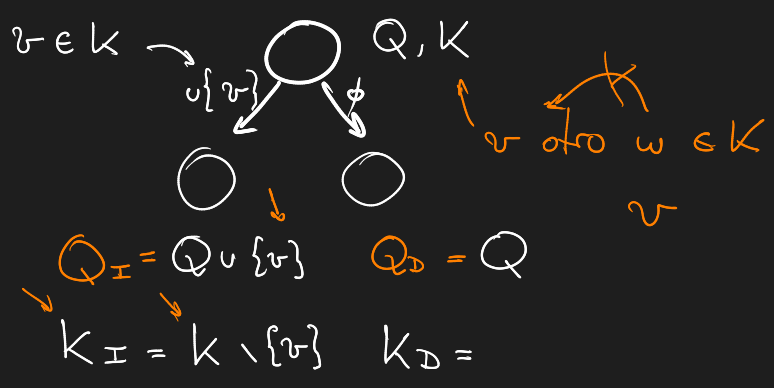
- En cada nodo (Q, K), el conjunto K satisface dos propiedades invariantes. Por un lado, cada $v \in K$ es amigo de todos los actores de Q, lo que implica que $Q \cup \{v\}$ es una clique de G. Por otro lado, cada $v \in K$ tiene un no-amigo en K, lo que implica que al menos una clique de G contiene a Q pero no a v.

1. **C/u de todos los actores de K es amigo de todos los actores de Q**
2. **Pero no todos los amigos de K son amigos entre sí. De hecho, para cada actor de K hay un no-amigo en K.**

distinto, no? no! debería!

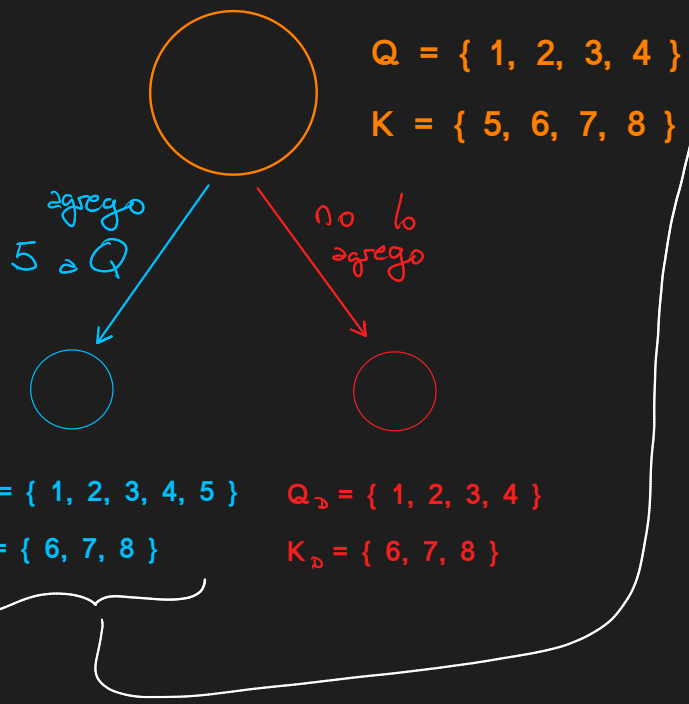
ver abajo

- La extensión del nodo (Q, K) genera dos subárboles a partir de un actor $v \in K$. En uno de estos subárboles, cada hoja se corresponde a una clique que contienen a $Q \cup \{v\}$. En el otro subárbol, cada hoja representa una clique que contiene a Q y no a v.



- **Ayuda:** recordar que al extender el nodo (Q, K) a partir de un vértice $v \in K$ se obtienen dos nodos hijo (Q_I, K_I) y (Q_D, K_D). Sin perdida de generalidad, supongamos que $v \in Q_I$, i.e., el subárbol de (Q_I, K_I) representa a cliques que contienen a $Q \cup \{v\}$. En cambio (Q_D, K_D) representa a las cliques que contienen a Q y no a v. Notar que, por invariante, ni Q_I ni K_I pueden contener actores que sean no-amigos de v. Por otra parte, K_I tampoco puede contener actores que sean amigos de todos los actores en $Q_I \cup K_I$; estos últimos pertenecen a todas las cliques que contienen a $Q \cup \{v\}$ y por lo tanto se pueden agregar a Q_I. Pensar cómo computar el resto de los parámetros de cada nodo, siguiendo un razonamiento similar.

"para cada $v \in K$, el único no-amigo de v, es sí mismo."



Por invariante de K_I :

- 6 es amigo de todo Q_I, lo mismo 7 y 8
- Pero entonces 6, 7 y 8 son amigos de 5, pues está en Q_I
- Pero entonces, volviendo un nodo arriba, 5 sí era amigo de todos los elementos de K, cuando sabemos que siempre debía haber un no-amigo, o sea, 5 no tenía no-amigos en K, pues todos eran sus amigos.

Respuesta: Debo sacar de K_I TAMBIEN al elemento w que es no-amigo de v

Poda:

- Para reducir el espacio de búsqueda, se poda cada nodo (Q, K) tal que $p(Q) + \sum_{v \in K} p(K) \leq p(S)$ para la clique S de mayor influencia encontrada hasta el momento.

if suma_restante < max_hasta_ahora
return

- El árbol se recorre recursivamente, avanzando primero en profundidad; pensar qué hijo conviene visitar primero en cada nodo.

?

- Antes de empezar, los actores se ordenan por algún criterio (ver abajo). En cada extensión, el candidato v es el primer vértice del orden que pertenece a K.

2. Considerar al menos dos opciones para ordenar a los actores antes de ejecutar el algoritmo: de mayor a menor influencia y de menor a mayor influencia.

Quiero sumar primero las influencias más grandes, y luego las más chicas? En qué dirección sumo en la recursión? cuando armo el árbol de recursión? o cuando devuelvo todas las llamadas?

3. Realizar un experimento con los casos de test de ambas opciones y reportar los resultados, justificando los mismos a través de analizar los efectos de la poda.

4. La operación de extensión define una función recursiva cuya descripción formal puede ser compleja. Sin definir formalmente esta función, explicar por qué la misma no tiene la propiedad de superposición de subproblemas.

llamados totales ~~>>~~ # llamados válidos
(post - poda)

Reunión con el grupo.

empiezo