Diseño y Análisis de Algoritmos

Tarea 6

Tony Santiago Montes Buitrago

Juan Carlos Marin Morales

Problema 1.

La alcaldía de Bogotá decidió comprar unas cámaras de vigilancia para la ciudad que tienen la capacidad de realizar video en 360 grados. Esto permite ubicarlas en intersecciones entre calles de manera tal que una cámara en una intersección permite vigilar al tiempo todas las calles que desemboquen a dicha intersección. Dado un mapa de calles e intersecciones de la ciudad, el distrito está interesado en saber el número mínimo de intersecciones en las que debe ubicar cámaras de manera tal que todas las calles queden vigiladas.

1. Especificación del Problema de Optimización

E/S	Nombre	Tipo	Descripción
E	G(V,E)	Grafo de la ciudad	Grafo no dirigido que con V representa las intersecciones y con E representa las calles.
S	m	nat	Mínimo numero de intersecciones en las que hay que colocar cámaras para tener todas las calles cubiertas

Para los problemas de optimización, decisión y verificación se realizan las siguientes definiciones previamente:

- Clase **G** que representa un grafo.
- **Predicados y Funciones:**

  - $CUBRE(x) \equiv (\forall e: E|: (\exists v: V|v \in x: v \in e))$
    "Para todo eje e existe un vértice v que pertenece a x, tal que v es un extremo de e"

Precondición

$$Q: \{true\}$$

Postcondición

$$R: \{s \in V^2 \wedge CUBRE(s) \wedge m = |s| \wedge (\forall a: V^2 |\ CUBRE(a): |a| \geq m)\}$$

"Entre todos los números de cámaras en intersecciones que cubren todas las calles, m es el menor"

2. Especificación del Problema de Decisión

E/S	Nombre	Tipo	Descripción
E	G(V,E)	Grafo de la ciudad	Grafo no dirigido que con V representa las intersecciones y con E representa las calles.
E	p	nat	Cota superior en el número de intersecciones donde poner cámaras
S	b	bool	¿Existe un numero de intersecciones menor a **p** donde poner cámaras tal que todas las calles queden cubiertas?

Precondición

$$Q: \{true\}$$

Postcondición

$$R: \{(\exists s: V^2 | CUBRE(s): |s| \leq p)\}$$

3. Especificación del problema de verificación

E/S	Nombre	Tipo	Descripción
E	G(V,E)	Grafo de la ciudad	Grafo no dirigido que con V representa las intersecciones y con E representa las calles.
E	x	Set of Intersección	Conjunto de intersecciones donde colocar las cámaras a verificar si cubre todas las calles.
E	p	nat	Cota superior en el número de intersecciones donde poner cámaras
S	b	bool	¿Existe un numero de intersecciones menor a **p** donde poner cámaras tal que todas las calles queden cubiertas?

Precondición

$$Q: \{|x| \leq p\}$$

Postcondición

$$R: \{CUBRE(s)\}$$

4. Algoritmo que soluciona el problema de verificación en tiempo polinomial.

```
func cubre(G(V,E), x: set) ret b: bool
var b:bool
var e:set
edges:= E
for e ∈ E
    if e.prev ∈ x ∨ e.next ∈ x
        edges = edges \ e
    fi
rof
b:= edges = ∅
ret
```

5. Estrategia de exploración de grafos

Estados:

Los estados son subconjuntos de intersecciones (Vértices) es decir V^2

Condición de Satisfactibilidad:

$$sat(x) \equiv CUBRE(x) \wedge |x| \leq p$$

Estado Inicial:

$$e_0 = \emptyset$$

Función Sucesor:

$$suc(x) = \{x \cup \{y\} | y \in V\}$$

Esta función de sucesor no induce a ciclos debido a que cada estado sucesor siempre tiene más estados que los estados anteriores.

Función de Viabilidad

$$viable(x) \equiv |x| \leq p$$

Si el estado (subconjunto) x no ha superado la cota p, es viable seguir por ese estado y sus sucesores; si supera la cota p, no es viable seguir por ese estado ni sus sucesores.

Problema 2.

Se diseñó un curso intensivo de un día que se va a dictar varias veces para atender a todos los estudiantes interesados. Se definieron unos días posibles en los que se va a dictar el curso y se pidió a los interesados en el curso que seleccionaran los días en los que tendrían disponibilidad para tomarlo. Se desea encontrar el mínimo número de días en los que se debe dictar el curso para atender a todos los estudiantes interesados

1. **Problema de Optimización:** Dadas unas fechas específicas en las que se dictará el curso y las listas de fechas seleccionadas por todos los estudiantes, escoger las fechas en los que se podrían dictar todas las clases tal que se minimicen.

E/S	Nombre	Tipo	Descripción
E	dts	set of Date	Conjunto de N fechas en las que se dictará el curso.
E	sel	array[0,M) of set of Date	Lista de conjuntos de fechas seleccionadas por los M estudiantes que tomarán el curso.
S	m	set of Date	Conjuntos de fechas finales, en las que todos los estudiantes pueden asistir, tales que minimizan el número de clases.

Para los problemas de optimización, decisión y verificación se realizan las siguientes definiciones previamente:

- Clase **Date** que representa una fecha.
- **Predicados y Funciones:**

  - $E(r) \equiv (\forall s \in sel \mid : (\exists d \in r \mid : d \in s))$
    "Para todos los estudiantes, al menos una fecha seleccionada está en r"

Precondición

$$Q: \{(\forall\, s \in sel \mid (\forall\, d \in s \mid d \in dts))\}$$

"Todas las fechas seleccionadas por los estudiantes están dentro de las ofertadas"

Postcondición

$$R: \{(E(m) \wedge (\forall subd \in 2^{dts} \mid E(subd) : |m| \leq |subd|))\}$$

"Todo posible subconjunto de fechas, tal que para todos los estudiantes al menos una fecha seleccionada está en él; tiene una longitud mayor o igual a la de m; y m también cumple la condición"

2. **Problema de Decisión:** Dadas unas fechas específicas en las que se dictará el curso, las listas de fechas seleccionadas por todos los estudiantes *y un número de fechas*, decir si en un número menor o igual de fechas se podrían dictar todas las clases.

E/S	Nombre	Tipo	Descripción
E	dts	set of Date	Lista de N fechas en las que se dictará el curso.
E	sel	array[0,M) of set of Date	Lista de listas de fechas seleccionadas por los M estudiantes que tomarán el curso.
E	x	nat	Número de fechas máximas.
S	b	bool	Existencial sobre si existe o no un número *menor o igual a x* de fechas finales, en las que todos los estudiantes puedan asistir.

Precondición

$$Q\{(\forall\, s \in sel \mid (\forall\, d \in s \mid d \in dts))\}$$

"Todas las fechas seleccionadas por los estudiantes están dentro de las ofertadas"

Postcondición

$$R\colon\{(b \equiv (\exists\, subd \in 2^{dts} \mid E(subd) : |subd| \leq x\,))\}$$

"Existe algún subconjunto de fechas, cuya longitud sea menor o igual a x, tal que para todos los estudiantes al menos una fecha seleccionada está en él"

3. **Problema de Verificación:** Dadas unas fechas específicas en las que se dictará el curso, las listas de fechas seleccionadas para todos los estudiantes, un número de fechas y *una lista de fechas* que "podría" ser la solución, verificar si en las fechas seleccionados se podrían dictar todas las clases.

E/S	Nombre	Tipo	Descripción
E	dts	set of Date	Lista de N fechas en las que se dictará el curso.
E	sel	array[0,M) of set of Date	Lista de listas de fechas seleccionadas por los M estudiantes que tomarán el curso.
E	x	nat	Número de fechas máximas.
E	p	set of Date	Lista de fechas a verificar.
S	b	bool	Existencial sobre si existe o no un número *menor o igual a x* de fechas finales, en las que todos los estudiantes puedan asistir.

Precondición

$$Q\colon\{(\forall\, s \in sel \mid (\forall\, d \in s \mid d \in dts))\}$$

"Todas las fechas seleccionadas por los estudiantes están dentro de las ofertadas"

Postcondición

$$R: \{(b \equiv E(p) \land |p| \leq x)\}$$

"Para todos los estudiantes al menos una fecha seleccionada está en p; y p tiene hasta x fechas"

4. **Algoritmo que soluciona el problema de verificación en tiempo polinomial. (Lenguaje Python)**

```python
def f(dts:set, sel:list, x:int, p:list) -> bool:
    if len(p) > x: return False
    for s in sel:
        for d in p:
            if d in s: break
        else: break
        continue
    else: return True
    return False
```

5. **Estrategia de exploración de grafos:**

Estados:

$$E = 2^{dts}$$

Los estados son todas las posibles partes de *dts*, que esto son todos los subconjuntos de *dts*.

Condición de satisfactibilidad:

$$sat(s) \equiv E(s)$$

Para que un estado sea una respuesta del problema, debe cumplir que para todos los estudiantes al menos una fecha seleccionada está en el conjunto; **esto ya lo verifica la función E**.

Estado Inicial:

$$s_0 = \emptyset$$

Función Sucesor:

$$suc(s) = \{s \cup \{y\} \mid y \in dts\}$$

Todos los sucesores de un estado (subconjunto), es dicho subconjunto, agregándole cada una de las fechas en las que se dictará el curso (y). Esta función de sucesores **no induce ciclos**, ya que

Función de Viabilidad:

$$viab(s) \equiv |s| \leq x$$

Si el estado (subconjunto) s no ha superado la cota x, es viable seguir por ese estado y sus sucesores; si supera la cota x, no es viable seguir por ese estado ni sus sucesores.

Problema 3.

Juan quiere invitar a sus amigos a conocer su nuevo apartamento. Sin embargo, tiene la dificultad de que sus amigos son algo conflictivos y entonces sabe que varias parejas de amigos se han peleado entre ellos. Debido a esto, tomó la decisión de organizar dos reuniones. Desafortunadamente, Juan ya se dio cuenta que no es posible distribuir a sus amigos en dos grupos de tal manera que dentro de cada grupo no haya parejas de personas que se hayan peleado entre ellas. El objetivo entonces es desarrollar un programa que reciba los amigos de Juan y las parejas de amigos que tienen conflictos y distribuya los amigos en dos grupos, de tal modo que para la mayor cantidad posible de parejas con conflicto, los dos miembros de la pareja queden en grupos separados.

1. **Especificación del problema de optimización**

E/S	Nombre	Tipo	Descripción
E	G(V,E)	Grafo de Amigos de Juan	Grafo no dirigido donde V representa cada amigo de Juan y E representa las parejas de amigos que tienen conflictos.
S	g1	set of Amigos de Juan	Uno de los conjuntos de amigos que Juan tiene que llevar un grupo, que junto a g2 tienen el mínimo de conflictos entre ellos
S	g2	set of Amigos de Juan	Uno de los conjuntos de amigos que Juan tiene que llevar un grupo, que junto a g1 tienen el mínimo de conflictos entre ellos

Para los problemas de optimización, decisión y verificación se realizan las siguientes definiciones previamente:

- Clase **G** que representa un grafo.
- **Predicados y Funciones:**
  - $d(a, W) = (+v{:}V \mid v \in W \wedge (\exists e{:}E \mid e =< a, v >){:}1)$
    "El número de conflictos de un vértice a dentro con otros vértices dentro del conjunto W"

  - $c(a) = (+w{:}V \mid w \in a{:}d(w, a))$
    "El número de conflictos en el conjunto a"

Precondición

$$Q{:}\{true\}$$

Postcondición

$$R{:}\left\{\ g1, g2 \in V^2 \wedge |g1| + |g2| = |V| \wedge \left(\forall a, b{:}V^2 \mid\mid a| + |b| = |V|{:}\, c(a) + c(b) \geq c(g_1) + c(g_2)\right)\right\}$$

"El numero de conflictos en los grupos de respuesta es menos a cualesquiera otros dos grupos posibles"

2. **Especificación del problema de decisión**

E/S	Nombre	Tipo	Descripción
E	G(V,E)	Grafo de amigos de Juan	Grafo no dirigido donde V representa cada amigo de Juan y E representa las parejas de amigos que tienen conflictos.
E	p	nat	Cota superior de conflictos permitidos dentro de los dos grupos

S	b	bool	¿Existen dos subgrupos de amigos tales que el total de conflictos dentro de los subgrupos sean menores a p?

Precondición

$$Q: \{true\}$$

Postcondición

$$R: \{(\exists v, w: V^2 \mid |v| + |w| = |V| : c(v) + c(w) \leq p)\}$$

3. Especificación del problema de verificación

E/S	Nombre	Tipo	Descripción
E	G(V,E)	Grafo de amigos de Juan	Grafo no dirigido donde V representa cada amigo de Juan y E representa las parejas de amigos que tienen conflictos.
E	g1	set of Amigos de Juan	Uno de los grupos de amigos tales que junto a g2 el número de conflictos es menor a **p**
E	g2	set of Amigos de Juan	Uno de los grupos de amigos tales que junto a g1 el número de conflictos es menor a **p**
E	p	nat	Cota superior de conflictos permitidos dentro de los dos grupos
S	b	bool	¿Los dos grupos g1 y g2 tienen un numero de conflictos internos menor a **p**?

Precondición

$$Q: \{|g_1| + |g_2| = |V|\}$$

Postcondición

$$R: \{c(g_1) + c(g_2) \leq p\}$$

4. Algoritmo que soluciona el problema de verificación en tiempo polinomial. (Lenguaje GCL)

```
1. func f(G(V,E): Graph, g1: set, g2: set, p:nat) ret b:bool
2. var b: bool
3. var x: nat
4. x:= 0
5. for u ∈ g1
6.     for e ∈ E
7.         if e.prev = u ∧ e.next ∈ g1
8.             x:= x+1
9.         fi
10.    rof
11.rof
12.for u ∈ g2
13.    for e ∈ E
14.        if e.prev = u ∧ e.next ∈ g2
15.            x:= x+1
16.        fi
```

```
17.     rof
18.rof
19.
20.b:= (x/2) ≤ p
21.ret
```

5. Estrategia de exploración de grafos

Estados:

Los posibles estados son las posibilidades que surgen de que cada amigo esté en un grupo u otro de la solución. Entonces cada estado está conformado por 2 conjuntos de amigos, donde se cumple que $g1 \cap g2 = \emptyset$.

Condición de satisfactibilidad:

$$sat(x) \equiv c(x.g_1) + c(x.g_2) \leq p$$

Estado Inicial:

$$g_1 = \emptyset\,, g_2 = V$$

Función Sucesor:

$$suc(x) = \{y \in Estados |\ (\forall v{:}\, V | x.g_1 \cup v \wedge x.g_2 \setminus v)\}$$

La función no genera ciclos ya que siempre se sacan elementos de un conjunto y se meten al otro. Entonces es imposible volver a un estado anterior con la función sucesor.

Función de Viabilidad:

No existe una función de viabilidad ya que cambiar de un estado no correcto a un sucesor no implica que un sucesor futuro no pueda ser satisfactorio.