UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias, 2023-2

Complejidad Computacional

Programa 01

Zamora Cruz Diego Arturo — 316249560

Zamora Cruz Diego Arturo
Cuenta: 316249560

1. Alcanzabilidad

- **Forma canónica**
  - Ejemplar genérico: Sea $G = (V, E)$ una gráfica no dirigida.
  - Pregunta de decisión: ¿Existe un camino que no repite vértices de s a t con $s, t \in V$ de G?
  - Enunciado de optimización: Determinar un camino que no repita vértices de s a t con $s, t \in V$ de G.
- **Instrucciones** Para ejecutar la solución al problema 3Sat, dentro de la carpeta *src/* ejecutar el siguiente comando para compilar el programa

```
$ javac *.java
```

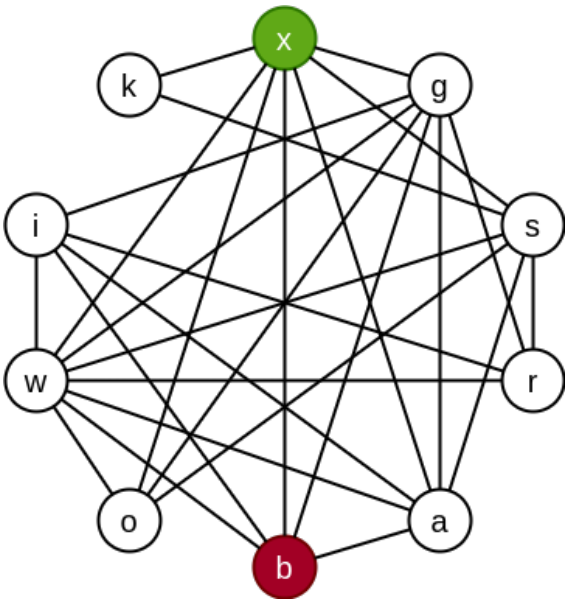
A continuación ejecutar el programa con el comando

```
$ java Programa1 -r
```

Y se obtendrá una salida como la siguiente

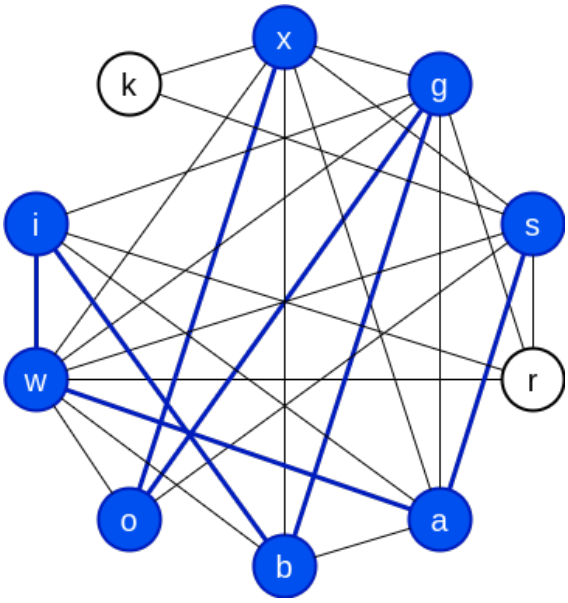
```
Ejemplar de Alcansabilidad:
Vertices: [V:x, V:g, V:s, V:r, V:a, V:b, V:o, V:w, V:i, V:k]
Aristas: [(V:x, V:g), (V:x, V:w), (V:x, V:o), (V:x, V:b), (V:x, V:s), (V:x, V:a), (V:g,
V:a), (V:g, V:r), (V:g, V:o), (V:r, V:w), (V:r, V:s), (V:a, V:s), (V:a, V:w), (V:b, V:g),
(V:b, V:a), (V:o, V:s), (V:w, V:s), (V:w, V:o), (V:w, V:g), (V:w, V:b), (V:i, V:r), (V:i,
V:w), (V:i, V:a), (V:i, V:g), (V:i, V:b), (V:k, V:s), (V:k, V:x)]
Vertice s: V:x
Vertice t: V:b
Candidato a solucion:
[V:x, V:o, V:g, V:b, V:i, V:w, V:a, V:s]
Si es solucion para el ejemplar
V:x -> V:o -> V:g -> V:b
```

Donde el primero valor sera la entrada del problema, en particular, la entrada anterior tendría la siguiente equivalencia

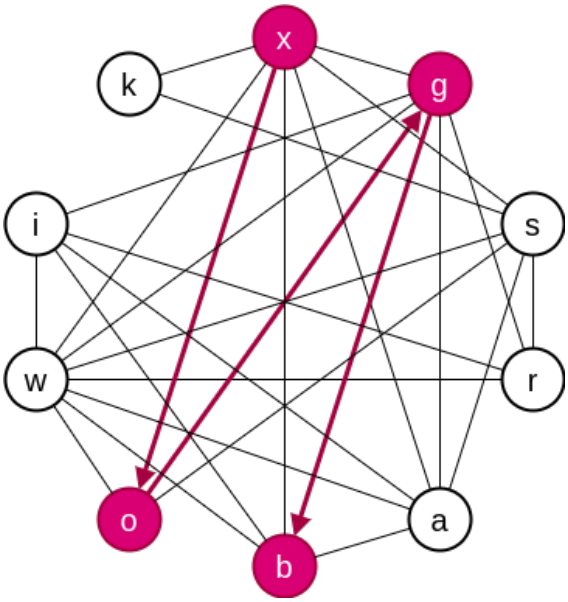


Zamora Cruz Diego Arturo
Cuenta: 316249560

El candidato a solución se interpretaría de la siguiente forma
Es una lista de las vértices que trazan una ruta, es decir



Finalmente, la fase verificadora indicara si el candidato a solución es un camino desde el vértice s al vértice t.



Zamora Cruz Diego Arturo
Cuenta: 316249560

- **Ejecución**

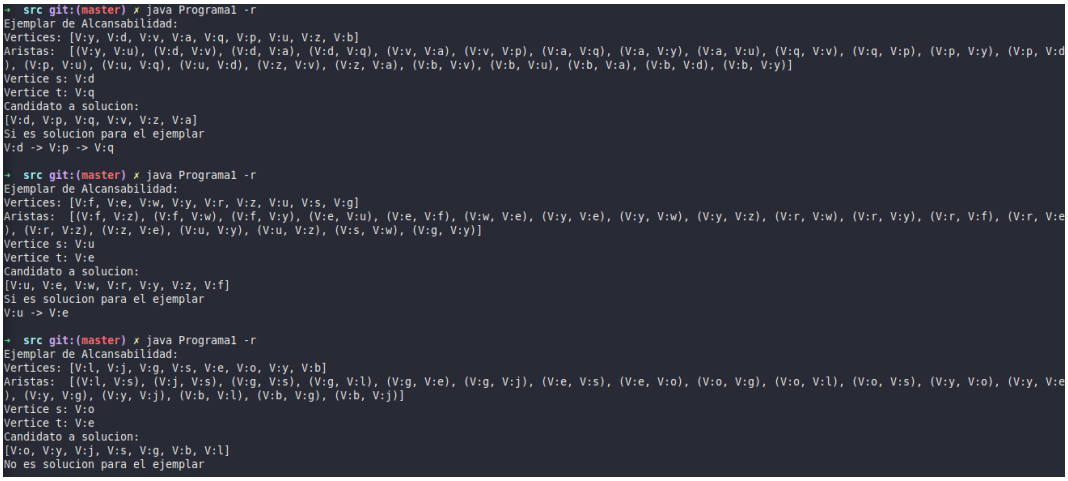


Figura 1: Captura de la ejecución con ejemplares aleatorios no. 1, 2 y 3

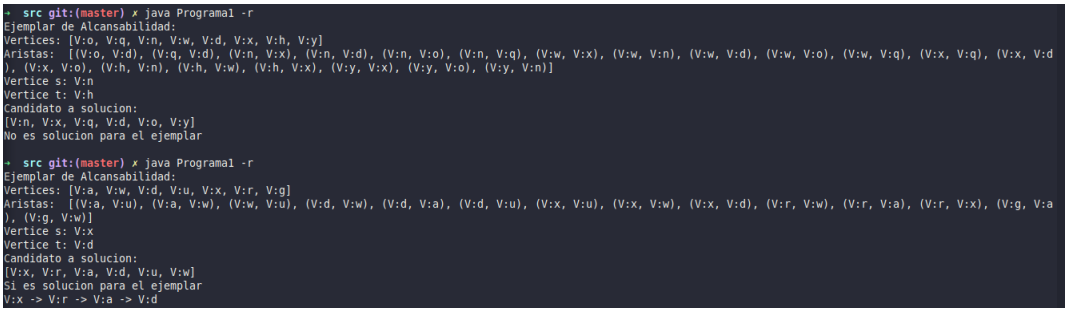


Figura 2: Captura de la ejecución con ejemplares aleatorios no. 4 y 5

- **Algoritmo**

Algorithm 1 solucionReachability

Input: G : Graph; s, t : Vertex
{PreCond:
G es una gráfica no dirigida y $s, t \in V$ de G}

```
1:  var camino, disponibles : List¡Vertex¿;
2:  var auxV ← s;
3:  G.setVerticeVisitado(auxV, true);
4:  camino.add(auxV);
5:  adyacentes ← G.getAdyacentes(auxV);
6:  for vertice in adyacentes do
7:     if vertice.getVisitado() == False then
8:        disponibles.add(vertice);
9:     end if
10: end for
11: while disponibles.size() > 0 do
12:    // Tomamos un valor del conjunto {0, ..., disponibles.size()}
13:    tomar ← ndChoice(0, disponibles.size());
14:    adyacente ← disponibles.get(tomar);
15:    auxV ← adyacente;
16:    G.setVerticeVisitado(auxV, True);
17:    camino.add(auxV);
18:    adyacentes ← G.getAdyacentes(auxV);
19:    for vertice in adyacentes do
20:       if vertice.getVisitado() == False then
21:          disponibles.add(vertice);
22:       end if
23:    end for
24: end while
25: for vertice in camino do
26:    if vertice.equasl(t) then
27:       return camino;
28:    end if
29: end for
30: return null;
```

{PostCond:
camino es una lista finita sin elementos repetidos, de vértices v
tales que siguiendo la secuencias de v_is tendremos el camino desde s hasta t}

2. 3-Sat

Forma canónica

- Ejemplar genérico: Sea E una expresión lógica en FNC donde cada clausula contiene exactamente 3 variables.
- Pregunta de decisión: ¿Existe una asignación de valores para todas las variables $v \in E$, tal que E se satisface?
- Enunciado de optimización: Determinar la asignación de valores para cada variable $v \in E$, tal que E se satisface.

Instrucciones

Para ejecutar la solución al problema 3Sat, dentro de la carpeta *src/* ejecutar el siguiente comando para compilar el programa

```
$ javac *.java
```

A continuación ejecutar el programa con el comando

```
$ java Programa1 -t
```

Y se obtendrá una salida como la siguiente

```
Ejemplar de 3SAT:
[(!d,!a,b), (!p,!x,!x), (!b,!y,b), (!x,q,!y), (k,!e,l)]
Candidato a solucion:
[d, a, b, p, !x, y, q, k, e, l]
Si es solucion para el ejemplar
```

Donde el primero valor sera la entrada del problema, en particular, la entrada anterior tendría la siguiente equivalencia
$(\neg d \vee \neg a \vee b) \wedge (\neg p \vee \neg x \vee \neg x) \wedge (\neg b \vee \neg y \vee b) \wedge (\neg x \vee q \vee \neg y) \wedge (k \vee \neg e \vee l)$

El candidato a solución se interpretaría de la siguiente forma
Es una lista de las variables con su respectivo valor de verdad, es decir

- `v`: la variable `v` tiene como valor de verdad *True*
- `!v`: la variable `v` tiene como valor de verdad *False*

Finalmente, la fase verificadora indicara si el candidato a solución satisface a la expresión de entrada.

Zamora Cruz Diego Arturo
Cuenta: 316249560

- **Ejecución**

```
→  src javac *.java
→  src java Programa1 -t
Ejemplar de 3SAT:
[(!s,k,g), (!w,f,w), (b,g,!i), (z,o,!f), (r,!o,b)]
Candidato a solucion:
[!s, k, g, !w, !f, !b, !i, !z, !o, r]
No es solucion para el ejemplar

→  src java Programa1 -t
Ejemplar de 3SAT:
[(g,r,!x), (!z,a,!s), (e,!r,!z), (!w,!c,!a), (k,!x,g)]
Candidato a solucion:
[!g, r, !x, z, a, !s, e, !w, !c, !k]
No es solucion para el ejemplar

→  src java Programa1 -t
Ejemplar de 3SAT:
[(!y,a,w), (n,z,!a), (!t,q,w), (x,!e,a), (d,!w,n)]
Candidato a solucion:
[!y, !a, !w, n, !z, t, q, !x, e, !d]
No es solucion para el ejemplar

→  src java Programa1 -t
Ejemplar de 3SAT:
[(a,q,x), (j,d,c), (b,!i,z), (y,c,!z), (b,!b,d)]
Candidato a solucion:
[a, !q, x, !j, d, c, b, !i, !z, y]
Si es solucion para el ejemplar

→  src java Programa1 -t
Ejemplar de 3SAT:
[(!j,g,s), (r,e,!b), (!l,!l,e), (s,!l,a), (f,!p,j)]
Candidato a solucion:
[!j, !g, s, r, e, b, !l, a, !f, !p]
No es solucion para el ejemplar
```

Figura 3: Captura de la ejecución con 5 ejemplares aleatorios

- **Algoritmo**

Algorithm 2 solucion3Sat

Input: E : List
{PreCond:
E es una lista finita de triplas con variables booleanas}

```
1:  var variables : List;
2:  for clausula in E do
3:    if variables.contains(clausula.getX()) then
4:      variables.add(clausula.getX());
5:    end if
6:    ⋮ {Se repite para los elementos Y y Z}
7:  end for
8:  for i ← 0, i < variables.size(), i ← i + 1 do
9:    var aux ← variables.get(i);
10:   // Tomamos un valor del conjunto {0,1}
11:   if ndChoice(0, 1) == 1 then
12:     aux.setValue(True);
13:   else
14:     aux.setValue(False);
15:   end if
16: end for
17: for clausula in E do
18:   var varX, varY, varZ : boolean;
19:   for variable in variables do
20:     if variable.equals(clausula.getX()) then
21:       varX ← variable.getValue();
22:     end if
23:     ⋮ {Se repite para los elementos Y y Z}
24:   end for
25:   if clausula.getX().getValue() == false then
26:     varX ← !varX;
27:   end if
28:   ⋮ {Se repite para los elementos Y y Z}
29:   if (varX or varY or varZ) == False then
30:     return null;
31:   end if
32: end for
33: return variables;
```

{PostCond:
variables es una lista finita sin elementos repetidos, de variables v
tal que v con su respectivo valor son una asignación de valores validad para satisfacer la expresión E}