

Programa 1

Ian Israel García Vázquez

15 de septiembre de 2022

1. Problema de Alcanzabilidad

Dada una gráfica no dirigida $G = (V, E)$, con dos vértices distinguidos s y t . ¿Existe un camino que no repite vértices de s a t en G ?

1.1. Forma canónica

Ejemplar genérico: Gráfica simple $G = (V, E)$

Enunciado de optimización: Determinar el camino C en G de s a t cuyo conjunto de vértices $|V_c| \leq |V|$.

Ejemplar genérico: Una gráfica simple $G = (V, E)$

Pregunta de decisión: ¿Existe un camino C en G con los vértices distinguidos s, t ?

Algoritmo:

```
proceso generaRuta(Gráfica G, Vertice S, Vertice T):
  IMPRIME(G)
  contadorDeLecturas=0
  Stack ruta=[]
  Vertice V = S;
  ruta.agrega(V)
  mientras TIENEVECINOS(V):
    mientras V está en ruta:
      contadorDeLecturas++
      V=ESCOGE_VECINO_AL_AZAR[V]
      if contadorDeLecturas >= NoVecinos(V):
        break
    if V no está en ruta:
      ruta.agrega(V)
      contadorDeLecturas=0
    if T == V ó contadorDeLecturas>=NoVecinos(V):
      break

  IMPRIME(ruta)
  return ULTIMOAGREGADO(ruta)==T
```

```

iangarcia@2806-107e-0017-14ce-73fe-5c23-3033-283b:~/Documentos/Complejidad_C/Programal
iangarcia@2806-107e-0017-14ce-73fe-5c23-3033-283b ~$ python Programal.py
{([1, [5, 13, 3, 6, 7, 10, 12, ]], ([2, [3, 6, 4, 9, 5, 10, 7, 8, 12, 11, ]], ([3, [1, 2, 6, 21, 7, 4, 5, 10, 13, ]], ([4, [2, 3, 5, 6, 7, 12, ]], ([5, [1, 2, 21, 3, 10, 4, 6, 9, 12, ]], ([6, [2, 3, 9, 4, 11, 8, 5, 1, 14, 10, 12, 21, ]], ([7, [2, 3, 1, 10, 4, 8, 14, 9, 12, 13, 21, ]], ([8, [2, 6, 7, 10, 12, 13, 14, ]], ([9, [2, 6, 13, 7, 5, 10, 12, 14, ]], ([10, [2, 5, 6, 7, 8, 12, 11, 9, 1, 13, 21, 3, 14, ]], ([11, [6, 10, 2, 12, 21, ]], ([12, [2, 10, 11, 7, 5, 6, 4, 1, 9, 8, 21, 14, ]], ([13, [1, 9, 10, 8, 7, 3, ]], ([14, [6, 7, 12, 8, 9, 10, 21, ]], ([21, [3, 5, 10, 12, 14, 7, 6, 11, ]], )
Camino: deque([5, 2, 9, 7, 13, 8, 1, 6, 3, 4, 12])
False
iangarcia@2806-107e-0017-14ce-73fe-5c23-3033-283b ~$ python Programal.py
{([1, [2, 7, 11, 9, 10, 5, 6, 8, ]], ([2, [1, 8, 5, 3, 7, 10, 4, 9, ]], ([3, [2, 6, 10, 21, 11, 9, ]], ([4, [10, 6, 11, 2, 8, 7, 21, ]], ([5, [1, 2, 8, 10, 11, ]], ([6, [1, 3, 4, ]], ([7, [1, 2, 10, 4, 21, ]], ([8, [2, 4, 5, 1, 11, 21, ]], ([9, [1, 3, 2, 10, ]], ([10, [1, 2, 3, 4, 5, 7, 9, 11, ]], ([11, [1, 3, 4, 10, 5, 8, 21, ]], ([21, [3, 7, 4, 8, 11, ]], )
Camino: deque([5, 2, 8, 1, 3, 10, 7, 21])
True
iangarcia@2806-107e-0017-14ce-73fe-5c23-3033-283b ~$ python Programal.py
{([1, [11, 9, 3, 10, 7, 6, 4, 12, 21, ]], ([2, [11, 3, 6, 21, 4, 5, 12, 8, 9, ]], ([3, [1, 2, 7, 5, 11, 12, 21, ]], ([4, [2, 6, 7, 8, 21, 10, 11, 1, ]], ([5, [2, 3, 8, 10, 11, 12, 6, ]], ([6, [1, 2, 4, 5, 21, 7, 9, 11, ]], ([7, [1, 3, 4, 6, 8, 9, 11, 12, ]], ([8, [2, 4, 5, 12, 7, ]], ([9, [1, 6, 21, 2, 10, 11, 7, 12, ]], ([10, [1, 4, 5, 9, 12, 21, ]], ([11, [1, 2, 4, 5, 9, 6, 3, 7, 21, ]], ([12, [2, 5, 8, 21, 10, 1, 9, 7, 3, ]], ([21, [2, 4, 6, 9, 12, 1, 3, 10, 11, ]], )
Camino: deque([5, 8, 12, 10, 9, 1, 4, 2, 3, 6, 21])
True
iangarcia@2806-107e-0017-14ce-73fe-5c23-3033-283b ~$ python Programal.py
{([1, [11, 8, 21, 12, 4, 2, 3, ]], ([2, [1, 6, 12, 4, 3, 8, 7, ]], ([3, [2, 1, 6, 11, 10, 8, 9, 5, 12, 21, ]], ([4, [1, 2, 11, 21, 9, 5, 7, ]], ([5, [3, 4, 12, 6, 10, 21, ]], ([6, [2, 3, 12, 21, 9, 5, 11, ]], ([7, [4, 10, 2, ]], ([8, [1, 2, 3, 11, 10, 12, 21, ]], ([9, [3, 4, 6, 10, 12, 21, ]], ([10, [3, 7, 8, 11, 12, 9, 5, ]], ([11, [1, 3, 4, 8, 10, 6, 21, 12, ]], ([12, [1, 2, 5, 6, 8, 10, 9, 11, 3, 21, ]], ([21, [1, 4, 6, 11, 12, 8, 3, 9, 5, ]], )
Camino: deque([5, 6, 12, 3, 2, 1, 10, 9, 21])
True
iangarcia@2806-107e-0017-14ce-73fe-5c23-3033-283b ~$ python Programal.py
{([1, [4, 2, 7, 21, 9, 3, 6, 5, 8, ]], ([2, [1, 6, 3, 21, 8, 9, ]], ([3, [1, 2, 7, 8, 9, 21, ]], ([4, [1, 9, 5, 21, ]], ([5, [6, 21, 9, 4, 1, 7, ]], ([6, [1, 2, 5, 8, 9, ]], ([7, [1, 5, 3, 21, ]], ([8, [6, 2, 21, 9, 1, 3, ]], ([9, [1, 4, 5, 6, 8, 3, 2, ]], ([21, [1, 2, 5, 8, 7, 3, 4, ]], )
Camino: deque([5, 1, 3, 8, 6, 21])
True
iangarcia@2806-107e-0017-14ce-73fe-5c23-3033-283b ~$

```

En esta imagen tenemos la gráfica generada y las soluciones que nos otorgan el programa, así como su evaluación, nótese que para este caso debemos nuestra $s = 5$ y $t = 21$

2. Problema 3SAT

3-SAT

Una variante del problema de satisfacibilidad, dadas n -variables y m -clausulas, determinar si la proposición es satisfacible, en este caso cada clausula deberá conformarse de al menos tres variables.

2.1. Forma canónica

Ejemplar genérico: Conjunto de Clausulas $C_1 * C_2 \cdots C_i$, tal que cada clausula se conforma de k -variables, con $k = 3$

Pregunta de decisión: ¿Es satisfacible el conjunto de k clausulas si cada clausula debe componerse de al menos 3 clausulas?

Algoritmo:

```

PROCESO 3SAT(LC <- Lista de clausulas):
    boolean evaluacion;
    boolean fst_time=true;
    for Clausula from LC:
        fst_time=true
        if fst_time then:
            evaluacion=Clausula.ValorBooleano
            fst_time=false
        else:
            evaluacion&=Clausula.ValorBooleano
        endif
    return evaluacion

```

```
Actividades Terminal 15 de sep 16:46
iangarcia@fedora:~/Documentos/Complejidad_C/Programa1

iangarcia@fedora:~/Documentos/Complejidad_C/Programa1 x iangarcia@fedora:~ x
iangarcia@fedora ~/Documentos/Complejidad_C/Programa1 Program1.2 python 3SAT.py
Ejemplar [+y(False), q(True), r(True)], +[-y(True), h(True), r(True)], +[e(True), h(True), r(True)], +[s(True), b(False), r(True)], +[q(True), n(False), r(True)]
Respuesta: True
iangarcia@fedora ~/Documentos/Complejidad_C/Programa1 Program1.2 python 3SAT.py
Ejemplar [+e(True), f(True), -i(False)], +[-e(False), f(True), -i(False)], +[l(True), f(True), -i(False)], +[-f(False), z(True), -i(False)], +[-f(False), y(False), -i(False)]
Respuesta: False
iangarcia@fedora ~/Documentos/Complejidad_C/Programa1 Program1.2 python 3SAT.py
Ejemplar [+p(False), b(False), m(True)], +[f(False), n(True), m(True)], +[w(True), b(False), m(True)], +[-w(False), -n(False), m(True)], +[-b(True), n(True), m(True)]
Respuesta: True
iangarcia@fedora ~/Documentos/Complejidad_C/Programa1 Program1.2 python 3SAT.py
Ejemplar [+b(True), f(False), -j(True)], +[-f(True), r(True), -j(True)], +[o(False), x(True), -j(True)], +[c(True), -f(True), -j(True)], +[v(True), f(False), -j(True)]
Respuesta: True
iangarcia@fedora ~/Documentos/Complejidad_C/Programa1 Program1.2 python 3SAT.py
Ejemplar [+z(True), b(True), t(True)], +[b(True), m(False), t(True)], +[-b(False), m(False), t(True)], +[j(False), h(False), t(True)], +[h(False), -m(True), t(True)]
Respuesta: True
iangarcia@fedora ~/Documentos/Complejidad_C/Programa1 Program1.2
```