**Pontificia Universidad Católica Madre y Maestra**

**Facultad de Ciencias de la ingeniería,**

**Departamento de Ingeniería en Sistemas y Computación**

**Diseño y análisis de Algoritmos**



Trabajo:

T3-2Sat

Materia:

Diseño y análisis de Algoritmos

ST-ISC-405-T-001

Estudiantes:

Francis Cáceres

2012-0559

Profesor:

Juan R. Núñez

Introducción

El formato del archivo es el siguiente. En cada caso, el número de variables y el número de cláusulas esel mismo, y este número se especifica en la primera línea del archivo. Cada línea subsiguiente especifica una cláusula a través de sus dos literales, con un número que indica la variable y un signo "-"denota lógica "no". Por ejemplo, la segunda línea del primer archivo de datos es "-16808 75250", que indica la cláusula ¬x16808 x75250.

Su tarea consiste en determinar cuál de los 6 casos son satisfiable, y que son unsatisfiable. En la respuesta , introduzca una cadena de 6 bits, donde el bit i-ésimo debe ser 1 si la instancia i-ésimo es satisfiable, y 0 en caso contrario. Por ejemplo, si usted piensa que los 3 primeros casos son satisfiable yel último 3 no son, entonces usted debe responder con 111000 en la respuesta.

DISCUSIÓN: Esta asignación es deliberadamente abierta, y se puede poner en práctica lo que el algoritmo 2SAT desea. Por ejemplo, 2SAT reduce al cálculo de los componentes fuertemente conexos de un gráfico adecuado (con dos vértices por variable y dos bordes dirigidos por cláusula, usted debe pensar en los detalles). Esto podría ser una opción especialmente atractiva para alguno que ha codificado un algoritmo SCC . Como alternativa, puede utilizar el algoritmo de búsqueda local aleatorizado de Papadimitriou. (El algoritmo de clases es probablemente demasiado lento como se ha dicho, por lo que es posible que desee hacer una o más simples modificaciones en él --- incluso si esto significa romper el análisis realizado en clases --- para asegurar que se ejecute en una cantidad razonable de tiempo.) Un tercer enfoque es a través de retroceso.

Desarrollo

En teoría de la complejidad computacional, el Problema de satisfacibilidad booleana (también llamado SAT) fue el primer problema identificado como perteneciente a la clase de complejidad NP-completo.

El problema SAT se trata de un problema donde interesa saber si una expresión booleana con variables y sin cuantificadores tiene asociada una asignación de valores para sus variables que hace que la expresión sea verdadera.

**import** random  
**import** math  
**def** main():  
 f=open(**"2sat4.txt"**,**"r"**)  
 num\_vars=int(f.readline())  
 inst={}  
 choices=[True, False]  
 tclauses={}  
 **for** i **in** range(1,num\_vars+1):  
 tclauses[i]=[]  
 **for** i **in** range(1,num\_vars+1):  
 s=f.readline().rstrip()  
 variables=s.split(**' '**)  
 **if** int(variables[0])<0: tclauses[0-int(variables[0])].append(variables)  
 **else**: tclauses[int(variables[0])].append(variables)  
 **if** variables[0]!=variables[1]:  
 **if** int(variables[1])<0: tclauses[0-int(variables[1])].append(list(reversed(variables)))  
 **else**: tclauses[int(variables[1])].append(list(reversed(variables)))  
 inst[str(i)]=random.choice(choices)  
 f.close()  
 **for** i **in** range(1,num\_vars+1):  
 **if** len(tclauses[i])==0: **del** tclauses[i]  
 oldnum=num\_vars+1  
 newnum=num\_vars  
 **while** oldnum>newnum:  
 oldnum=newnum  
 **for** tc **in** tclauses.keys():  
 **if** len(tclauses[tc])==0:  
 **del** tclauses[tc]  
 **else**:  
 first=tclauses[tc][0][0]  
 sameSign=True  
 **for** i **in** tclauses[tc]:  
 **if** i[0]!=first:  
 sameSign=False  
 **break**

El código siguente fue la solución:

**if** sameSign:  
 **for** tcc **in** tclauses[tc]:  
 var1=tcc[0]  
 var2=tcc[1]  
 **if** int(var1)<0: var1=str(0-int(var1))  
 **if** int(var2)<0: var2=str(0-int(var2))  
 var11=True  
 var22=True  
 **if** int(tcc[0])<0:  
 **if** inst[str(0-int(tcc[0]))]: var11=False  
 **else**: var11=True  
 **else**: var11=inst[tcc[0]]  
 **if** int(tcc[1])<0:  
 **if** inst[str(0-int(tcc[1]))]: var22=False  
 **else**: var22=True  
 **else**: var22=inst[tcc[1]]  
 **if not**(var11 **or** var22):  
 inst[var1]=**not** inst[var1]  
 **if** tcc[0]!=tcc[1]:  
 tclauses[int(var2)].remove(list(reversed(tcc)))  
 tclauses[tc].remove(tcc)  
 **if** len(tclauses[tc])==0:  
 **del** tclauses[tc]  
 newnum-=1  
 flag=False  
 **for** varc **in** tclauses.keys():  
 **if** len(tclauses[varc])==0: **del** tclauses[varc]  
 **else**:  
 **for** c **in** tclauses[varc]:  
 var1=c[0]  
 var2=c[1]  
 **if** int(var1)<0: var1=str(0-int(var1))  
 **if** int(var2)<0: var2=str(0-int(var2))  
 **if** c[0]!=c[1]:  
 tclauses[int(var2)].remove(list(reversed(c)))  
 **for** i **in** range(0,int(math.log(num\_vars,2))+1):  
 j=1  
 **while** (**not** flag) **and** j<(2\*num\_vars\*num\_vars):  
 flag=True  
 **for** varc **in** tclauses.keys():  
 **for** c **in** tclauses[varc]:  
 var1=c[0]  
 var2=c[1]  
 **if** int(var1)<0: var1=str(0-int(var1))  
 **if** int(var2)<0: var2=str(0-int(var2))  
 var11=True  
 var22=True  
 **if** int(c[0])<0:  
 **if** inst[str(0-int(c[0]))]: var11=False  
 **else**: var11=True  
 **else**: var11=inst[c[0]]  
 **if** int(c[1])<0:  
 **if** inst[str(0-int(c[1]))]: var22=False  
 **else**: var22=True  
 **else**: var22=inst[c[1]]

**if not**(var11 **or** var22):  
 **if** random.choice([1,2])==1: inst[var1]=**not** inst[var1]  
 **else**: inst[var2]=**not** inst[var2]  
 flag=False  
 **break  
 if** flag:  
 **print "Satisfiable"  
 break** j=j+1  
 **if not** flag: **print "Unsatisfiable"  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

Para el resultado, solo obtuve dos los cuales fueron Sat3 y Sat4 los cuales me dieron satisfabilidad mientras que los demás no arrojaron resultados por lo que una posible conclusión seria 001100:



