Guia 3 Complejidad

Ejercicio 5 Suponiendo que P = NP, diseñar un algoritmo polinomial que dada una fórmula booleana ϕ encuentre una asignación que la satizfaga, si es que ϕ es satisfacible.

Por hipotesis SAT $\in$ P:

Existe un programa SAT(ϕ) que devuele V si se puede satisfacer ϕ, F caso contrario.

Puedo definir tambien SAT-R(ϕ,r,k) que toma k restricciones y restricciones es una lista de boleanos tq: r_i es el valor que debe tener la x_i variable de ϕ.

Por ejemplo SAT-R(ϕ,[True,False],2) verifica si ϕ es satisfactible con la primera variable siendo T y la segunda F.

Sigue perteneciendo a NP (y por hipótesis a P) ya que la unica diferencia es que cuando se verifica el certificado (valuacion tq la formula sea satisfactible) hay que chequear que las primeras k variables del certificado sean las mismas que las restricciones.

Tambien se podria escribir reduccion para ver SAT(ϕ) $\leq_p$ SAT-R(ϕ, r, k):

$$\phi \in \text{SAT} \iff f(\phi) \in \text{SAT-R}$$

```
f(phi):
    return < phi, [],0>
```

Con esto en mente defino un algoritmo:

```
def asignacion(phi):
    if SAT(phi):
        asignacion = []
        for i in range(|phi.variables|):
            if SAT-R(phi,asignacion + [True],i+1):
                asignacion.push(True)
            else:
                asignacion.push(False)
        return asignacion
    else:
        print("no hay asignacion valida")
```

Basicamente va dejando fijas las variables y chequea si fijando Verdadero se puede satisfacer, sino la fija falso. Hace eso con todas hasta obtener una asignacion valida.

El programa corre en tiempo polinomial, pues hace cantidad de variables iteraciones, que es polinomial respecto de ϕ. En cada iteracion corre SAT-R que por hipótesis es polinomial.

Ejercicio 6 Suponiendo que P = NP, diseñar un algoritmo polinomial que dado un grafo G retorne una clique de tamaño máximo de G.

Por hipotesis tengo una maquina polinomial Clique(g,k) que me dice si G tiene una clique de tamaño k.

Busco primero el tamaño maximo de clique en G. Para cada nodo n chequeo si G-n sigue teniendo clique de tamaño maximo. Si se mantiene significa que n es dispensable asi que lo saco del grafo. Cuando termine de recorrer solo quedan nodos “indispensables” que forman la clique asi que devuelvo los nodos restantes en el grafo.

```
def max_clique(g):
    tamaño = 0
    for i in range(|g.v|+1):
        if clique(g,i):
            tamaño++
        else:
            break

    for nodo in g.v:
        if(clique(g.sin_nodo(nodo),tamaño)): // si hay clique de tamaño maximo en el grafo s
            g.sacar_nodo(n)
    return g.v // retorno los que quedaron, es decir los que forman la clique
```

Calcular el tamaño maximo es polinomial, por cada nodo del grafo que es polinomial respecto a su tamaño realizo una operación polinomial (clique).

Despues Simplemente vuelvo a recorrer la lista y hago operaciones polinomiales.

Ejercicio 7 Sabiendo que CLIQUE es NP-completo, demostrar que SUBGRAPH ISOMORPHISM es NPcompleto

Quiero ver:

$$\langle g,k\rangle \in CLIQUE \iff f(\langle g,k\rangle) \in SUBGRAPHISOMORPHISM$$

F va a tomar G, k y deolver G, H_k

Con H_k un grafo completo de tamaño k.

Veamos que vale:

$$\langle g,k\rangle \in CLIQUE \iff \langle G,H_k\rangle \in SUBGRAPHISOMORPHISM$$

$\Rightarrow$)

$\langle g, k\rangle \in CLIQUE \Rightarrow$ G tiene subgrafo completo de tamaño k, lo llamo G_k

Notar que G_k es isomorfo a **cualquier** grafo completo de tamaño k.

$\Rightarrow G$ es un grafo y H_k es isomorfo al grafo inducido de G $G_k \Rightarrow \langle G, H_k\rangle \in SUBGRAPHISOMORPHISM$

$\Leftarrow$)

$\langle G, H_k\rangle \in SUBGRAPHISOMORPHISM \Rightarrow G$ grafo y H_k es isomorfo a un grafo inducido de G

$\Rightarrow$ G tiene un subgrafo de tamaño k completo $\Rightarrow \langle G, k\rangle \in CLIQUE$