Trabajo Final

Teoria de la Computacion

Federico Vareika, Guillermo Golpe

2 de Julio 2025

Ejercicio 1

1.4 Reducción `reduceAtoB`

Para obtener `reduceAtoB`, debemos encontrar un plan de reducciones que reduce **3-SAT** al problema **B**. En nuestra investigación logramos obtener el siguiente plan de reducciones:

$$\textbf{3-SAT} \leq_p \text{Subset Sum} \leq_p \text{Knapsack} \leq_p \mathbf{B}$$

Reducción 3-SAT $\leq_p$ Subset Sum

Cabe aclarar que esta reducción no fue ideada por nosotros, sino que tomamos como referencia la reducción mostrada en [1].

Para realizar esta reducción tenemos que moldar el problema **3-SAT** para que se pueda reducir de manera trivial a **Subset Sum**.

Tomamos el problema **3-SAT** como un conjunto de variables $x_1, ..., x_n$ y un conjunto de clausulas $c_1, ..., c_r$. Para cada variable x_i, se puede decir que tenemos dos opciones, que esta se evalue a `True` o que se evalue a `False`, llamaremos a estas opciones como a_i y b_i respectivamente. Si se toma la opcion b_i, entonces podemos concluir que cada clausula que tenga a la variable x_i negada se evaluará a `True`.

Si ahora tomamos estas opciones como numeros decimales de manera que:

$$a_i = b_i = 10^{n-i}$$

Entonces podemos formar la siguiente tabla tomando $n = 3$:

	x_1	x_2	x_3
a_1	1	0	0
b_1	1	0	0
a_2	0	1	0
b_2	0	1	0
a_3	0	0	1
b_3	0	0	1

En **3-SAT** solo podemos tomar una opcion por variable, y debemos definir todas las variables. Dado esto, si sumamos todas las opciones elegidas, y llamamos esta suma k:

	x_1	x_2	x_3
a_1	1	0	0
b_1	1	0	0
a_2	0	1	0
b_2	0	1	0
a_3	0	0	1
b_3	0	0	1
k	1	1	1

Se eligen las opciones en verde como ejemplo.

Teniendo esto y usando el hecho que si elegimos a_i entonces cada clausula que contenga x_i no negada se va a evaluar a `True`, podemos extender esta tabla. Para el ejemplo anterior tomamos la siguiente expresion:

$$\underbrace{(x_1 \vee \neg x_2 \vee x_3)}_{c_1} \wedge \underbrace{(\neg x_1 \vee \neg x_2 \vee \neg x_3)}_{c_2}$$

Y derivamos la tabla:

	x_1	x_2	x_3	c_1	c_2
a_1	1	0	0	1	0
b_1	1	0	0	0	1
a_2	0	1	0	0	0
b_2	0	1	0	1	1
a_3	0	0	1	1	0
b_3	0	0	1	0	1
k	1	1	1	0	3

Se puede ver en esta tabla como indicamos con un 1 las clausulas que se evaluan a `True` al elegir la opcion en esa fila, e indicamos con 0 el resto. Tambien extendimos k para incluir la suma de las opciones marcadas.

Podemos saber que se cumple **3-SAT** con las opciones elegidas sii se elige solo una opcion de cada variable 1 y todas las clausulas son evaluadas a `True` al menos una vez 2.

Estas reglas se ven cumplidas en los digitos de k:

$$digitoK(i) = \begin{cases} 1 & \text{si } i \in [0, n) & (1) \\ v & \text{si } i \in [n, n+r), \quad v \geq 1 & (2) \end{cases}$$

Para el ejemplo anterior, es claro que una solucion es x_1 = `True`, x_2 = `False`, x_2 = `True`. Por lo tanto, deberiamos poder comprobarlo con estas nuevas reglas:

	x_1	x_2	x_3	c_1	c_2
a_1	1	0	0	1	0
b_1	1	0	0	0	1
a_2	0	1	0	0	0
b_2	0	1	0	1	1
a_3	0	0	1	1	0
b_3	0	0	1	0	1
k	1	1	1	3	1

$$k = 11131 \Rightarrow k[i] = digitoK(i) \quad \forall i \in [0, n+r)$$

Podemos ver como nos estamos acercando a **Subset Sum**:

- Tenemos una entrada de numeros (a_i y b_i).
- Sumamos un subconjunto de estos numeros.
- Verificamos que esta suma cumple ciertas reglas.

El ultimo paso tendria que ser "verificamos que esta suma es igual a un numero k'. Para esto, debemos modificar las reglas originales:

$$digitoK'(i) = \begin{cases} 1 & \text{si } i \in [0, n) & (3) \\ 3 & \text{si } i \in [n, n+r) & (4) \end{cases}$$

Y agregamos nuevos elementos al conjunto de numeros. Estos van a ser dos por clausula, de tal manera que se podra 'rellenar' de cierta forma el digito de esta clausula. Llamaremos a estos numeros s_i y t_i. Podemos ver estos nuevos numeros en la siguiente tabla:

	x_1	x_2	x_3	c_1	c_2
a_1	1	0	0	1	0
b_1	1	0	0	0	1
a_2	0	1	0	0	0
b_2	0	1	0	1	1
a_3	0	0	1	1	0
b_3	0	0	1	0	1
s_1	0	0	0	1	0
t_1	0	0	0	1	0
s_2	0	0	0	0	1
t_2	0	0	0	0	1
k	1	1	1	3	3

Ahora podemos elegir los nuevos numeros para rellenar el ultimo digito de k:

	x_1	x_2	x_3	c_1	c_2
a_1	1	0	0	1	0
b_1	1	0	0	0	1
a_2	0	1	0	0	0
b_2	0	1	0	1	1
a_3	0	0	1	1	0
b_3	0	0	1	0	1
s_1	0	0	0	1	0
t_1	0	0	0	1	0
s_2	0	0	0	0	1
t_2	0	0	0	0	1
k	1	1	1	3	3

Y ahora:

$$k = 11133 \Rightarrow k[i] = digitoK'(i) \quad \forall i \in [0, n + r)$$

PPor lo tanto, ya podemos cambiar la tercer regla de 1.4 por:

- Verificamos que la suma es igual a $\underbrace{1...1}_{n}\underbrace{3...3}_{r}$

References

[1] Park City Mathematics Institute. *Lecture Notes B07*. 2007. URL: https://people.clarkson.edu/~alexis/PCMI/Notes/lectureB07.pdf.