3DM a Partition

Integrantes:
Airam Rafael Luque León
Lucas Hernández Abreu
Juan Salvador Magariños Alba
Alejandro García Perdomo

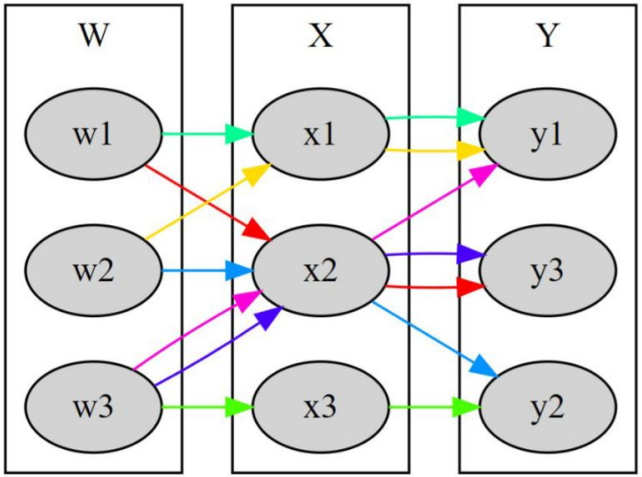
Índice

1. Problema de emparejamiento tridimensional (3DM)
2. Problema de la partición
3. Reducción de 3DM a Partition
4. Implementación
5. Referencias

1. Problema de emparejamiento tridimensional (3DM)

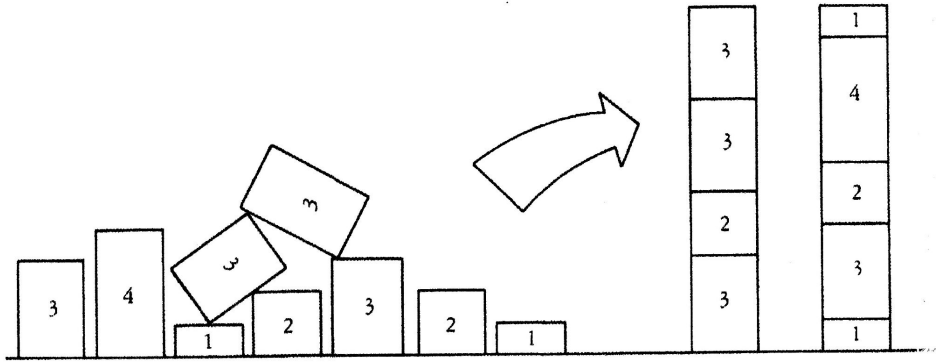
Sean los conjuntos M, W, X e Y tal que |W|=|X|=|Y|=q, y además, M ⊆ W x X x Y.

Buscamos encontrar un M' ⊆ M, tal que cubra todos los elementos de W, X y Y y solo los incluya una vez.



2. Problema de la partición (Partition)

Partimos de una lista de números L, la cual la tenemos que dividir en dos listas L_1 y L_2, tal que $\Sigma(L_1) = \Sigma(L_2)$



1. Problema de emparejamiento tridimensional (3DM)
2. Problema de la partición
3. **Reducción de 3DM a Partition**
4. Implementación
5. Referencias

3.1. Inicialización

Partimos de los conjuntos W, X, Y y M.

Los conjuntos W, X e Y deben tener la misma longitud. Mientras, el conjunto M, es una salida válida del problema 3DM con una cardinalidad propia.

$$W = \{w_1, w_2, ..., w_q\}$$
$$X = \{x_1, x_2, ..., x_q\}$$
$$Y = \{y_1, y_2, ..., y_q\}$$
$$M = \{m_1, m_2, ..., m_k\}$$

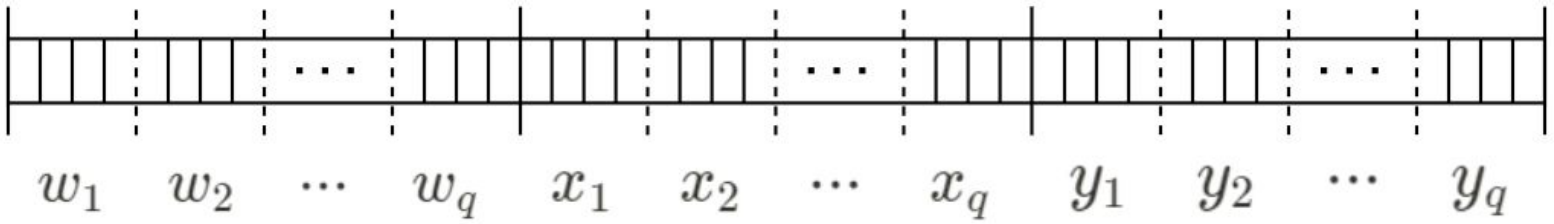
$$k = |M|$$
$$q = |W| = |X| = |Y|$$

3.2 Representación en binario

Necesitamos 3q zonas de tamaño p.

La que será la cardinalidad de nuestros conjuntos de entrada.

$$p = [\log_2 k + 1]$$



3.3. EJEMPLO

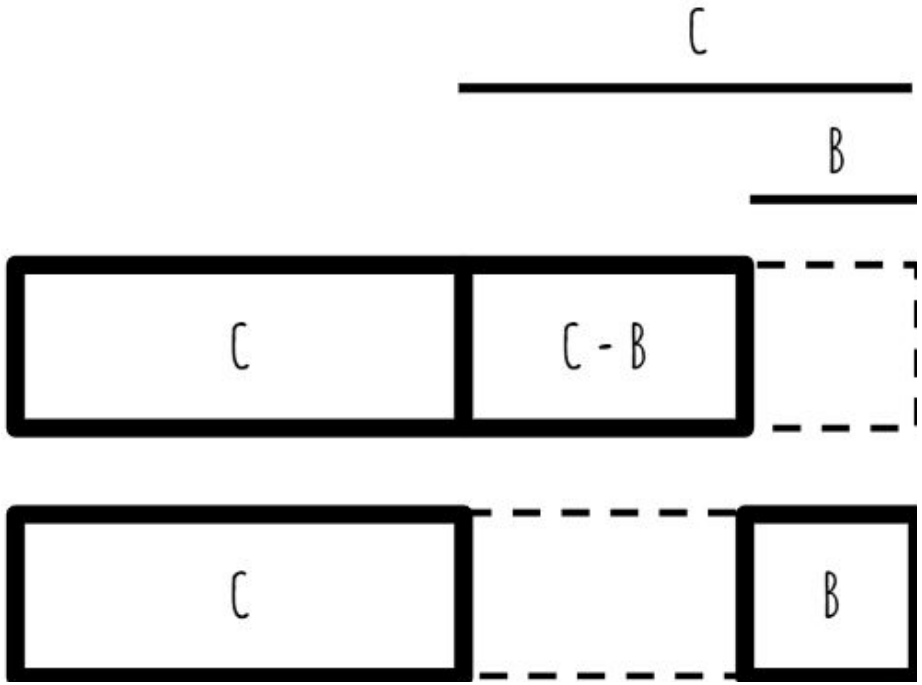
Supongamos unos conjuntos W, X e Y con cardinalidad 4.

	w_1	w_2	w_3	w_4	x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4	Decimal
$s(0)$	001	000	000	000	001	000	000	000	000	001	000	000	8592031808
$s(1)$	000	001	000	000	000	001	000	000	001	000	000	000	1074004480
$s(2)$	000	000	001	000	000	000	000	001	000	000	000	001	134221825
$s(3)$	000	000	001	000	001	000	000	000	000	000	001	000	16809992
$s(4)$	000	000	001	000	001	000	000	000	000	000	000	001	136314881
$s(5)$	000	001	000	000	000	001	000	000	001	000	000	000	1074004480
$s(6)$	000	000	001	000	000	001	000	000	001	000	000	000	134480384
C	001	010	011	001	010	011	001	001	011	001	001	010	11161867850
B	001	001	001	001	001	001	001	001	001	001	001	001	9817068105

3.4 Cálculo de b1 y b2

$$s(b_1) = 2 * C - B$$

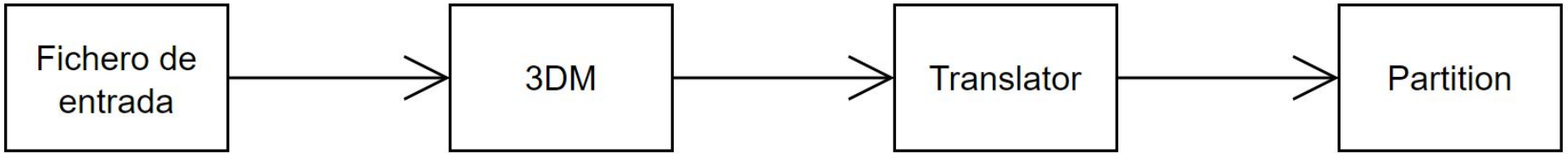
$$s(b_2) = C + B$$



1. Problema de emparejamiento tridimensional (3DM)
2. Problema de la partición
3. Reducción de 3DM a Partition
4. Implementación
5. Referencias

4.Implementación

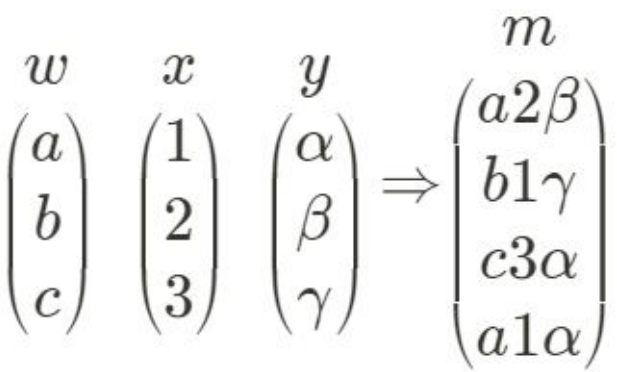
Se ha optado por desarrollar el programa en C# con la siguiente estructura:



4.1. Clase Program

```
public class Program {
    static void Main(string[] args) {
        ...
        _3DM instance3DM = new _3DM(inputFilePath);
        Partition instancePartition = Translator.Translate3DMToPartition(instance3DM);
        instancePartition.WriteToFile(outputFilePath);
    }
}
```

4.2. Fichero de entrada



JSON

```
{
  "wSet": ["a","b","c"],
  "xSet": ["1","2","3"],
  "ySet": ["α","β","γ"],
  "mSet": ["a2β","b1γ","c3α","a1α"]
}
```

4.3. Clase 3DM

```
public class _3DM {
    private uint sizeM_;
    private uint sizeWXY_;
    private string[] wSet_;
    private string[] xSet_;
    private string[] ySet_;
    private string[,] mSet_;
}
```

4.3. Clase 3DM

```
public class _3DM {
  public _3DM(string inputFileName) { ... }
  private void CheckValues(Dictionary<string, String[]> jsonMap) { ... }
  public uint GetMSize() { ... }
  public uint GetWXYSize() { ... }
  public int GetElementPositionInSet(string element, string setName) { ... }
  public string GetElement(int triplet, int position) { ... }
  public void Print() { ... }
}
```

4.4. Clase Partition

```
public class Partition {
    private ulong[] numbers_;

    public Partition(ulong[] numberList) {
        numbers_ = numberList;
    }

    public void WriteToFile(string outputFilePath)  {
        string jsonString = JsonSerializer.Serialize(numbers_);
        File.WriteAllText(outputFilePath, jsonString);
    }
}
```

4.5. Clase Translator

```
public class Translator {

  public static Partition Translate3DMToPartition(_3DM original_problem) {
    // nº de tripletas de M (k)
    uint sizeM = original_problem.GetMSize();
    // nº bits por elemento (p)
    double numberOfBits = Math.Ceiling(Math.Log(sizeM + 1.0f, 2f));
    // cardinalidad de los conjuntos W, X e Y (q)
    uint sizeWXY = original_problem.GetWXYSize();
    // Conversión a binario y cálculo de la suma total de los s(a)
    ulong sum = 0;
    List<ulong> numbers = new List<ulong>();
    const uint one = 1;

    ...
  }
}
```

4.5. Clase Translator

```
public class Translator {

  public static Partition Translate3DMToPartition(_3DM original_problem) {
    ...
    for (int triplet = 0; triplet < sizeM; ++triplet) {
      ulong newNumber = 0;

      // se extraen los elementos de la tripleta
      string firstElement = original_problem.GetElement(triplet, 0);
      string secondElement = original_problem.GetElement(triplet, 1);
      string thirdElement = original_problem.GetElement(triplet, 2);

      // se obtiene la posición de cada uno en el conjunto
      int firstPosition = original_problem.GetElementPositionInSet(firstElement, "w");
      int secondPosition = original_problem.GetElementPositionInSet(secondElement, "x");
      int thirdPosition = original_problem.GetElementPositionInSet(thirdElement, "y");

      // se crea el número binario que corresponde a la tripleta, y su valor se añade al total
      ulong first = (ulong)(Math.Pow(2, (numberOfBits * (3 * sizeWXY - firstPosition - 1))));
      ulong second = (ulong)(Math.Pow(2, (numberOfBits * (2 * sizeWXY - secondPosition - 1))));
      ulong third = (ulong)(Math.Pow(2, (numberOfBits * (sizeWXY - thirdPosition - 1))));

      newNumber = first + second + third;
      numbers.Add(newNumber);
      sum += newNumber;
    }
    ...
  }
}
```

4.5. Clase Translator

```
public class Translator {

  public static Partition Translate3DMToPartition(_3DM original_problem) {
    ...
    // Cálculo de B
    ulong matchingChecker = 0;
    for (int currentSet = 2; currentSet >= 0; currentSet--)  {
      for (int currentElement = (int)sizeWXY - 1; currentElement >= 0; currentElement--) {
        int shift = (int)(((currentSet * numberOfBits * 3) + ((2 - currentElement) * numberOfBits)));
        matchingChecker |= one << shift;
      }
    }

    // Cálculo de b1 y b2
    ulong b1 = 2 * sum - matchingChecker;
    ulong b2 = sum + matchingChecker;

    // Creación del Partition
    numbers.Add(b1);
    numbers.Add(b2);

    return new Partition(numbers.ToArray());
  }
}
```

5. Referencias

Garey, M. R., & Johnson, D. S. (2000). Computers and Intractability: A Guide to the Theory of NP-completeness (22.a ed.). W. H. Freeman and Company

NP-completeness Proff for 3DM, VC, CLIQUE, HC and PARTITION