

Welcome to Algorithms and Data Structures! - CS2100

Grafos dispersos

Qué es un grafo disperso?

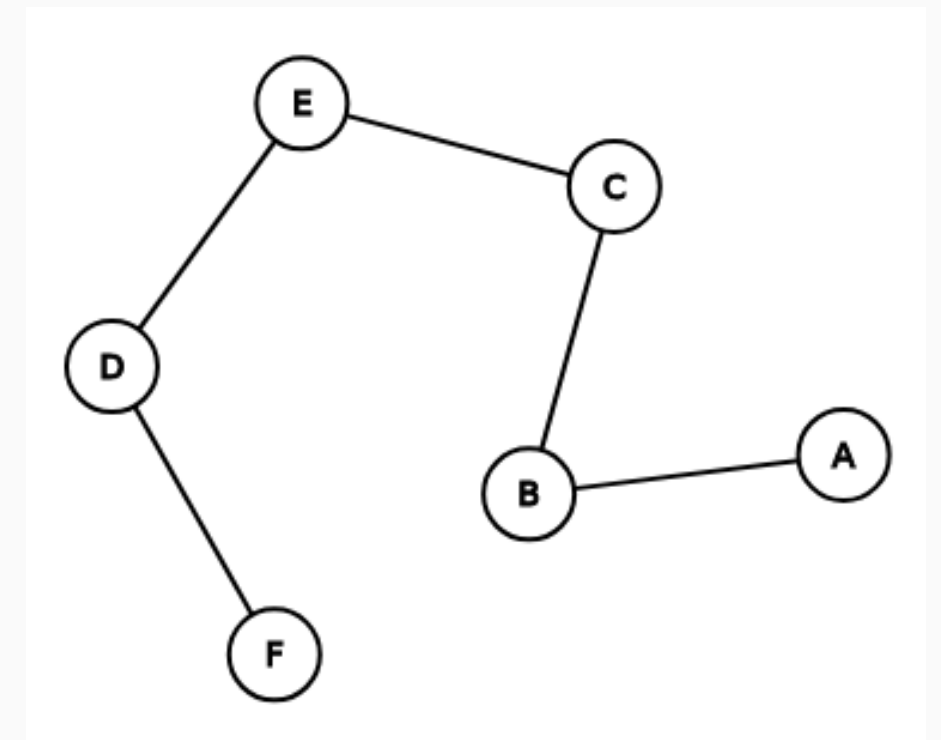
Cuando la mayoría de sus vértices no están conectadas por aristas

Cómo podemos representar un grafo?

Listas y matrices de adyacencia

Cuál sería el problema de usar matrices de adyacencia para grafos dispersos?

Que la mayoría de valores serían 0 (o algún valor que represente ausencia)



Matrices dispersas

Problema:

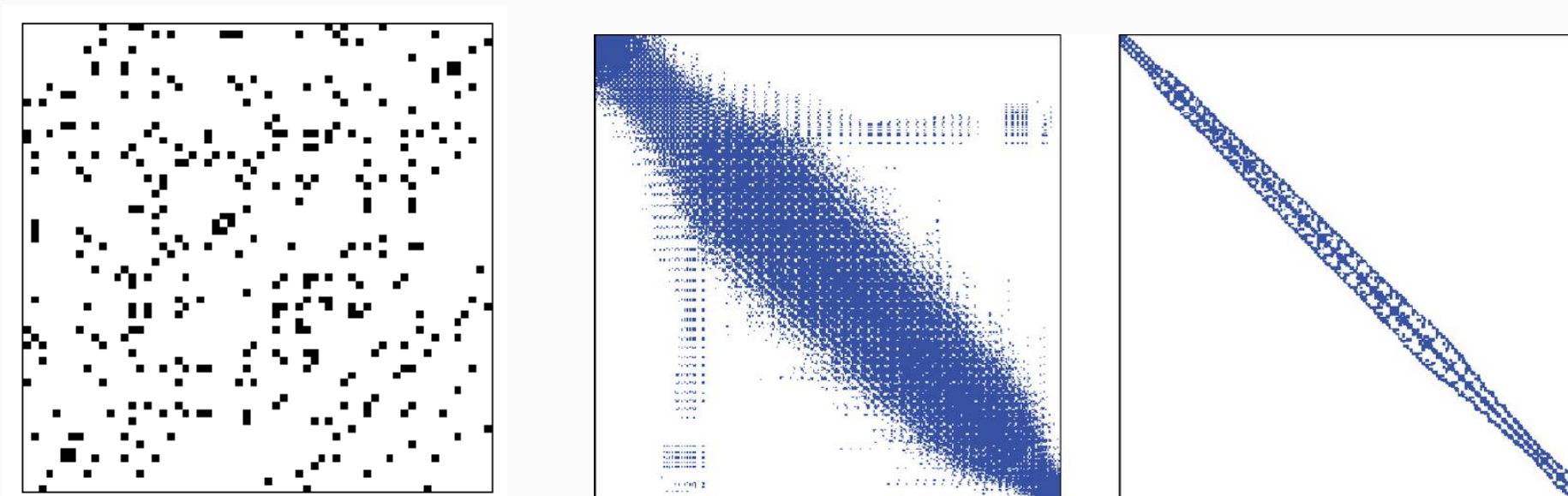
- En el caso de ser matrices muy grandes se ve que se tiene una gran cantidad de espacio desperdiciado
- Las matrices son bastante eficientes en el acceso de datos, por tanto se busca poder utilizarlas sin desperdiciar memoria
- Muchas aplicaciones reales tienden a ser matrices dispersas de gran tamaño (e.g. redes sociales)

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 & 3 & 4 & 0 \\ 0 & 5 & 0 & 6 & 0 & 0 & 0 \\ 0 & 7 & 8 & 9 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 4 & 5 & 0 \\ 0 & 0 & 6 & 0 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 9 \end{pmatrix}$$

Matrices dispersas

Son estructuras ampliamente usadas en computación científica, sobretodo en optimización a gran escala, teoría de redes y grafos

Dado que este tipo de matrices ocurren de manera muy natural, se han desarrollado distintas maneras de representarlas.



Matrices dispersas (array compacto)

Qué soluciones se les ocurren para poder evitar almacenar los ceros en nuestra matriz?

Quizás un array compacto? Cada celda almacena el índice de la columna y fila (de manera ordenada), y la data.

Cuál sería el problema con esta implementación?

Su falta de eficiencia para las operaciones básicas de inserción, eliminación, etc. Al ser un array los tiempos de acceso son en $O(n)$.

| | | | | |
|----------|----------|----------|----------|-----------|
| 1, 1 (5) | 1, 3 (1) | 2, 2 (9) | 4, 1 (3) | 4, 4 (11) |
|----------|----------|----------|----------|-----------|

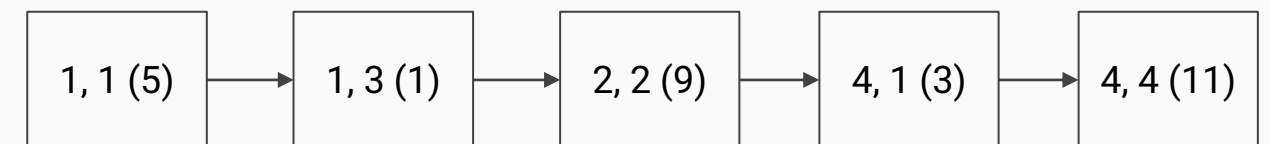
Matrices dispersas (listas enlazadas)

Qué soluciones se les ocurren para poder evitar almacenar los ceros en nuestra matriz?

Quizás una lista simplemente enlazada? Cada nodo almacena el índice de la columna y fila (de manera ordenada), y la data

Cuál sería el problema con esta implementación?

Su falta de eficiencia para las operaciones sobre matrices. Al ser una lista los tiempos de acceso son en $O(n)$. Las operaciones como multiplicación, suma tardarían mucho.



Matrices dispersas (coordenado)

Qué soluciones se les ocurren para poder evitar almacenar los ceros en nuestra matriz?

Formato coordenado, o array triple. Se utilizan tres vectores, en donde se guarda la posición de la fila, la posición de la columna, y el valor. Los problemas son que la asignación sigue siendo lenta, y ocupa mucha memoria cuando es denso.

```
int triplet[3][size];  
...  
if (matrix[i][j] != 0) {  
    triplet[0][k] = i;  
    triplet[1][k] = j;  
    triplet[2][k] = matrix[i][j];  
    k++;  
}
```

| Filas | Columnas | Valor |
|-------|----------|-------|
| 4 | 4 | 5 |
| 1 | 1 | 5 |
| 1 | 3 | 1 |
| 2 | 2 | 9 |
| 4 | 1 | 3 |
| 4 | 4 | 11 |

Matrices dispersas (hashing)

¿Qué tal si usamos hash/map?

Almacenar los elementos en un hash usando un función de transformación de índices.

$\text{Matriz}[i][j] \rightarrow \text{hash}[\text{trans}(i,j)]$ $\text{trans}(i,j) = i * m + j$

¿Cuál sería el problema con esta implementación?

Operaciones en matrices están enfocadas a la localización e iteración. Por lo tanto, para una implementación eficiente de las operaciones matriciales se debe iterar en mismo hash table.

El hash table ocupa más espacio de lo necesario. Rehashing. Colisiones.

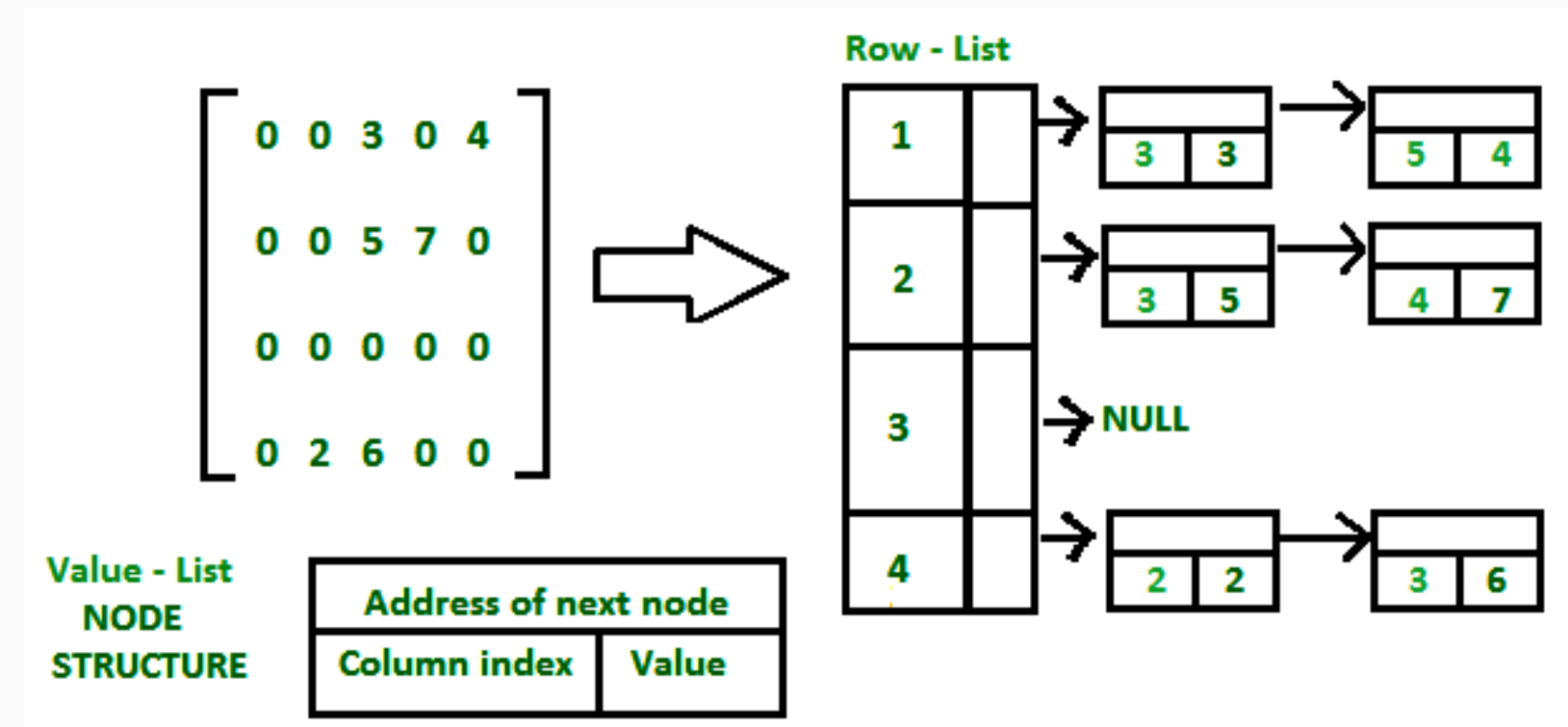
Matrices esparza (lista enlazada por fila)

Qué tal usar las listas enlazadas de otra manera?

Similar a como trabajamos con las listas de adyacencia, se podría tener una lista que mantenga los índices de las filas.

Cada fila tendría una lista enlazada para almacenar los elementos

Este formato posee dos ventajas. Primero: es más fácil poder ubicar los elementos, ya que podemos ir directo a su fila. Segundo, el valor de la fila ya no se tendrá que repetir en cada nodo



Matrices esparza (representación enlazada)

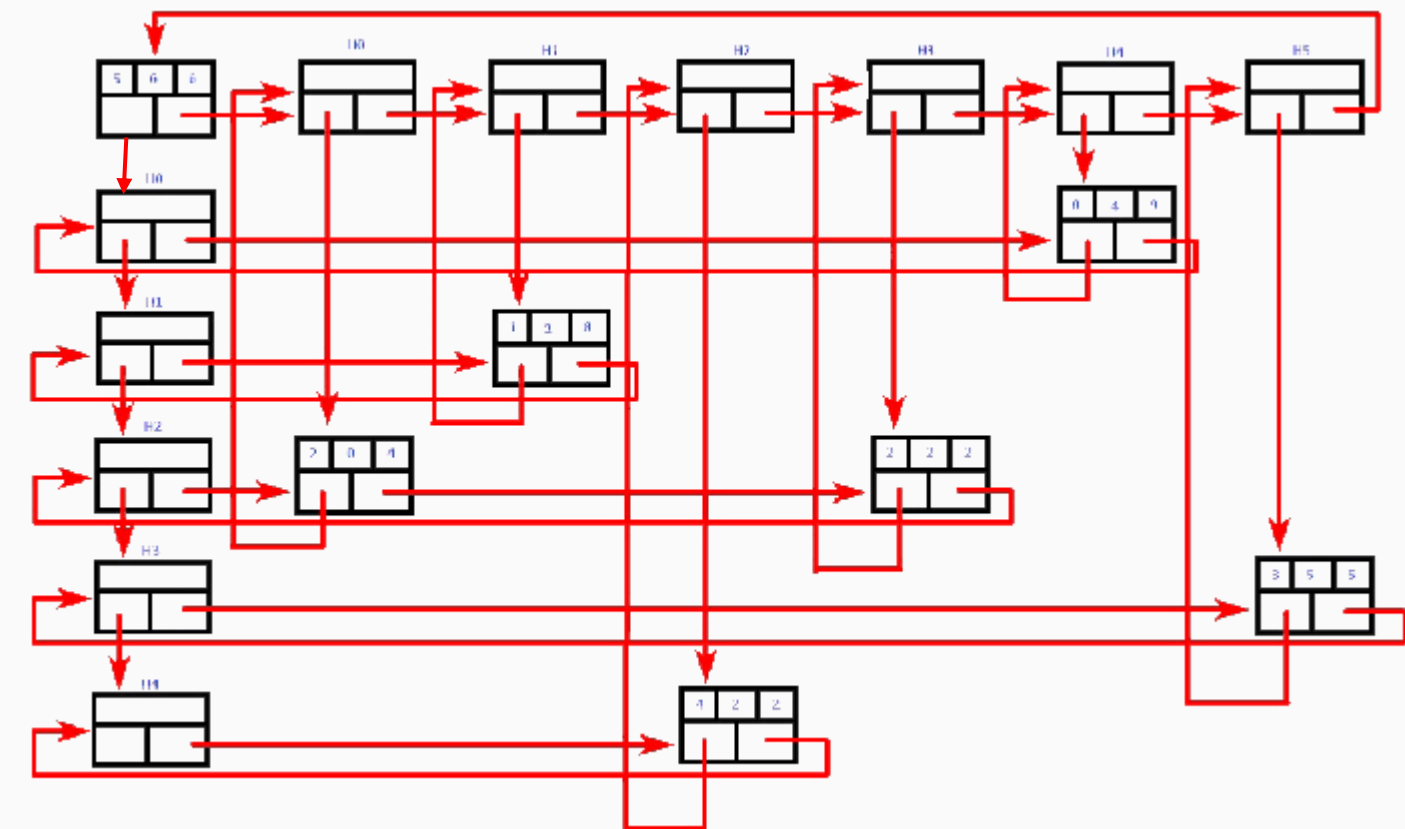
Se utilizan tres tipos de nodos:

1. Detalle (opcional)
2. Cabecera
3. Elemento

El nodo detalle suele contener la cantidad de filas y columnas de la matriz, y punteros hacia abajo y al siguiente

Las cabeceras tienen su posición en la fila o columna y un puntero hacia abajo y al siguiente

El elemento contiene la posición, su valor no nulo y un puntero hacia abajo y al siguiente

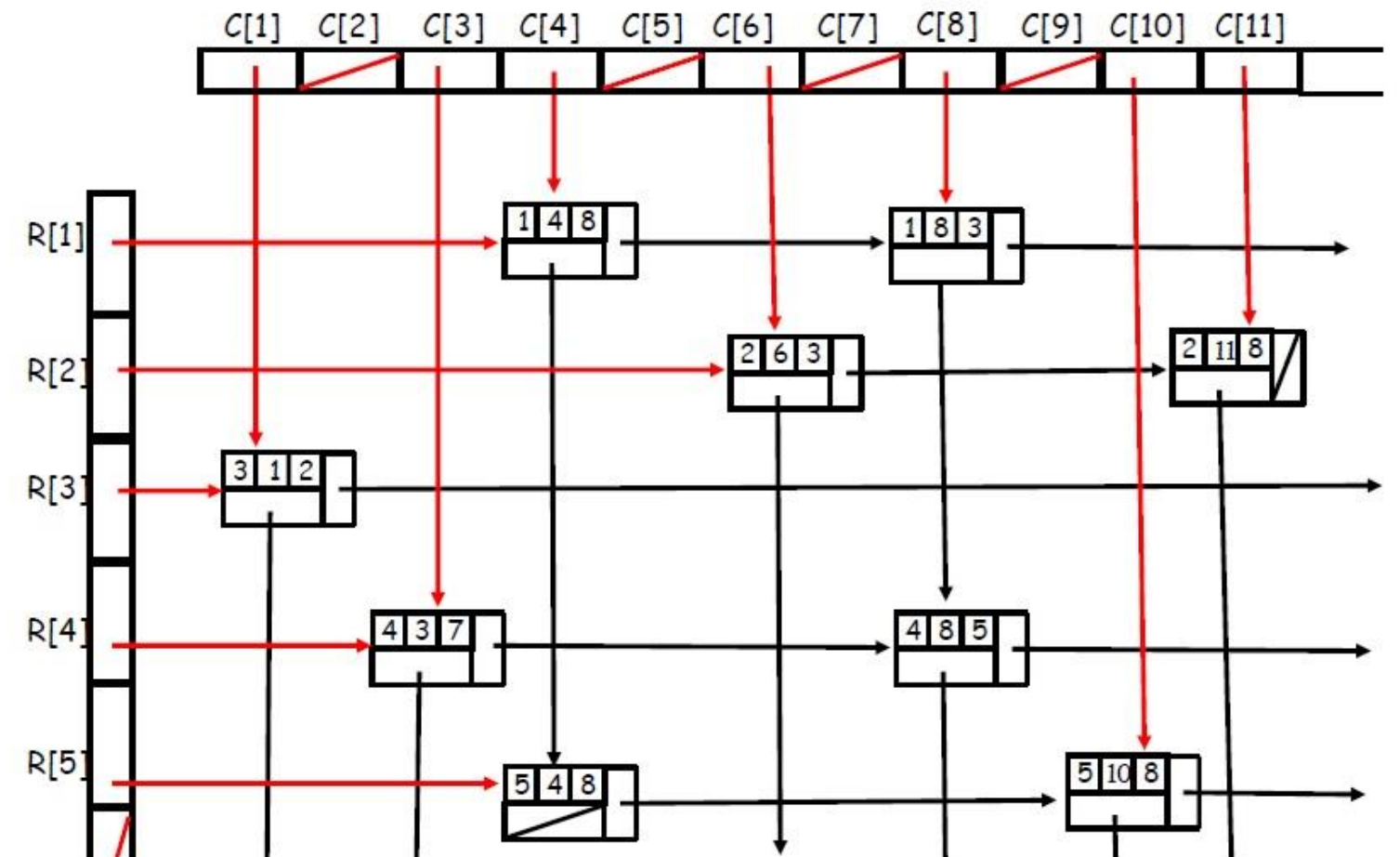


Matrices esparza (representación enlazada)

Implementación

```
template <class T>
class Node {
public:
    T data;
    int posX;
    int posY;
    Node<T>* next;
    Node<T>* down;
};
```

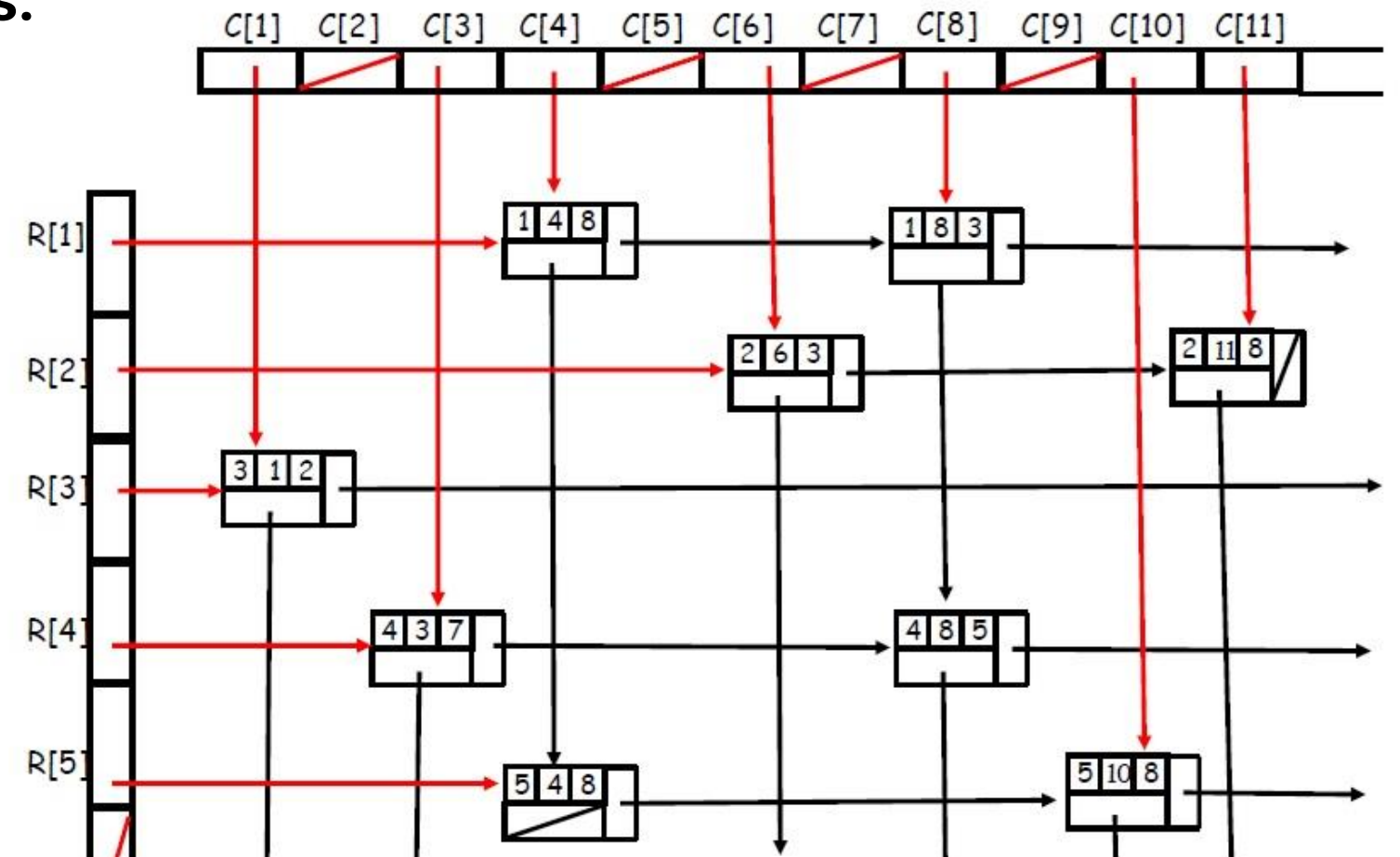
```
template <class T>
class SparseMatrix {
private:
    vector<Node<T>*> x;
    vector<Node<T>*> y;
    int columns;
    int rows;
};
```



Matrices esparza (representación enlazada)

Diseñar el pseudocódigo para las siguientes operaciones:

1. Insertar, eliminar y buscar
2. Obtener valor con sobrecarga de (i, j)
3. Comparar dos matrices
4. Suma y restar de matrices
5. Transpuesta de una matriz
6. Multiplicar dos matrices
7. Constructor y destructor



Welcome to Algorithms and Data Structures! - CS2100