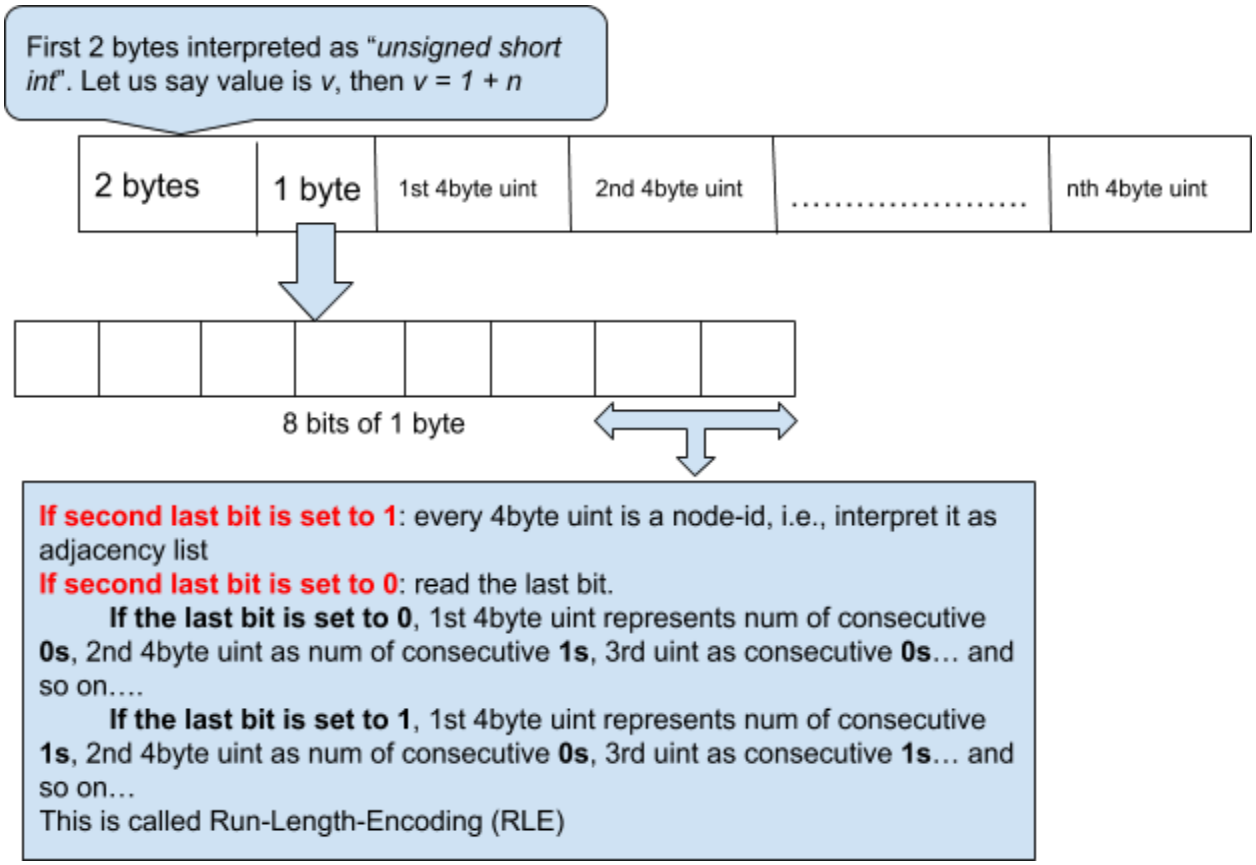
```
struct row {
	unsigned int rowid; //nodeid from the orig graph
	unsigned char *data; //pointer to adjacency list or adjacency
matrix row
	bool operator<(const struct row &a) const
	{
		return (this->rowid < a.rowid);
	}
};

/*
The *data pointer in row structure points to a char array that has
following representation
```




```
*/

typedef struct BitMat {
	list<struct row> bm;
	vector<struct row> vbm;
	unsigned int num_rows, num_totalBMs, num_columns, num_comm_so;
```

```
	unsigned int row_bytes, totalBMs_bytes, column_bytes,
common_so_bytes, dimension;
	unsigned long num_triples;
	unsigned char *rowfold; //row_bytearr
	unsigned char *colfold; //column_bytearr
	unsigned int last_op;

	void freebm(bool list = true, bool vector = true)
	{
		if (list) {
			for (std::list<struct row>::iterator it = bm.begin();
it != bm.end(); ){
				free((*it).data);
				it = bm.erase(it);
			}
		}
		if (vector) {
			for (std::vector<struct row>::iterator it =
vbm.begin(); it != vbm.end(); ){
				free((*it).data);
				it = vbm.erase(it);
			}
		}

		if (rowfold != NULL) {
			free(rowfold);
			rowfold = NULL;
		}

		if (colfold != NULL) {
			free(colfold);
			colfold = NULL;
		}
		num_triples = 0;
	}

	void reset(void)
	{
		for (std::list<struct row>::iterator it = bm.begin(); it
!= bm.end(); ){
			free((*it).data);
			it = bm.erase(it);
		}
```

```
            if (rowfold != NULL) {
                  memset(rowfold, 0, row_bytes);
            }
            if (colfold != NULL) {
                  memset(colfold, 0, column_bytes);
            }
            num_triples = 0;
      }

      ~BitMat()
      {
            for (std::list<struct row>::iterator it = bm.begin(); it
!= bm.end(); ){
                  free((*it).data);
                  it = bm.erase(it);
            }
            if (rowfold != NULL)
                  free(rowfold);
            if (colfold != NULL)
                  free(colfold);
      }

} BitMat;
```