第三次代码作业：二叉树

1. 根据马丁路德金的演讲原文《I have a dream》构建 Huffman 编码树，并对 dream 以及其他单词（自行选择）进行编码。

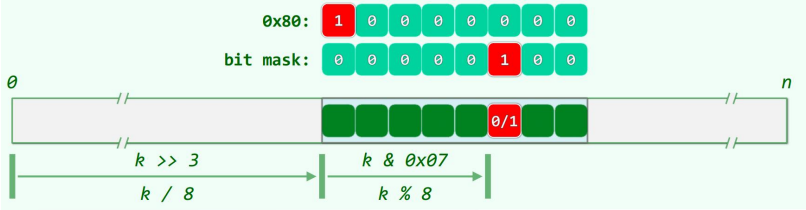
（1）实现二叉树数据结构 BinTree

（2）基于 BinTree 构建 HuffTree

（3）借助位图类 Bitmap 定义 Huffman 二进制编码串类型 HuffCode

（4）实现 Huffman 编码算法

参考：位图（Bitmap）是一种特殊的序列结构，可以动态地表示由一组（无符号）整数构成的集合。其长度无限，且每个元素的取值均为布尔型（初始均为 false），支持的操作接口主要包括：set（添加）、clear（删除）、test（判别）。如下图，借助整数移位和位运算得以高效实现。




```
class Bitmap { //位图 Bitmap 类
private:
	unsigned char* M;
	Rank N, _sz; //位图空间 M[]，N*sizeof(char)*8 个比特中含_sz 个有效位
protected:
	void init( Rank n )
	{ M = new unsigned char[N = ( n + 7 ) / 8]; memset( M, 0, N ); _sz = 0; }
public:
	Bitmap( Rank n = 8 ) { init( n ); } //按指定容量创建位图（为测试暂时选用较小的默认值）
	Bitmap( char* file, Rank n = 8 ) { //按指定或默认规模，从指定文件中读取位图
		init( n );
		FILE* fp = fopen( file, "r" ); fread( M, sizeof( char ), N, fp ); fclose( fp );
		for ( Rank k = 0, _sz = 0; k < n; k++ ) _sz += test(k);
	}
	~Bitmap() { delete[] M; M = NULL; _sz = 0; } //析构时释放位图空间
	Rank size() { return _sz; }
	void set ( Rank k ) { expand( k ); _sz++; M[k >> 3] |= ( 0x80 >> ( k & 0x07 ) ); }
	void clear ( Rank k ) { expand( k ); _sz--; M[k >> 3] &= ~ ( 0x80 >> ( k & 0x07 ) ); }
	bool test ( Rank k ) { expand( k ); return M[k >> 3] & ( 0x80 >> ( k & 0x07 ) ); }
	void dump( char* file ) //将位图整体导出至指定的文件，以便对此后的新位图批量初始化
	{ FILE* fp = fopen( file, "w" ); fwrite( M, sizeof ( char ), N, fp ); fclose( fp ); }
	char* bits2string( Rank n ) { //将前 n 位转换为字符串——
		expand( n - 1 ); //此时可能被访问的最高位为 bitmap[n - 1]
		char* s = new char[n + 1]; s[n] = '\0'; //字符串所占空间，由上层调用者负责释放
		for ( Rank i = 0; i < n; i++ ) s[i] = test( i ) ? '1' : '0';
		return s; //返回字符串位置
	}
	void expand( Rank k ) { //若被访问的 Bitmap[k]已出界，则需扩容
		if ( k < 8 * N ) return; //仍在界内，无需扩容
		Rank oldN = N; unsigned char* oldM = M;
		init( 2 * k ); //与向量类似，加倍策略
		memcpy_s( M, N, oldM, oldN );
		delete[] oldM; //原数据转移至新空间
	}
};
```