Kody wykrywające i korygujące błędy transmisji

Sformułowanie problemu

M możliwych wiadomości należy przedstawić przy użyciu ***M*** słów kodowych. Wybrane słowa kodowe powinny umożliwiać wykrywanie i (ewentualnie) korekcję zadanej liczby błędów. Jeżeli do zakodowania ośmiu wiadomości wykorzystamy trzy bity (podejście standardowe) to powstanie kod z wiadomościami: 000, 001, 010, 011, 100, 101, 110, 111. Niestety kod ten nie jest w żaden sposób odporny na błędy. Przekłamanie na dowolnej liczbie pozycji zawsze generuje słowo ze słownika tego kodu. Błędne dane nie mogą być więc odróżnione od poprawnych. Przykład ten pokazuje, że aby kod spełniał postawione warunki musi posiadać dłuższe słowa kodowe.
Jednak to nie wszystko. Aby umożliwić wykrywanie (w tym przypadku pojedynczego) błędu transmisji wprowadzenie zmiany na jednym bicie poprawnego słowa kodowego powinno powodować powstanie słowa nie należącego do słownika. Dla ułatwienia wprowadźmy pojęcie odległości słów kodowych. Odległość między dwoma binarnymi słowami kodowymi o równej długości definiuje się jako liczbę pozycji na których słowa te się różnią. Na przykład odległość między słowami 000 i 111 wynosi 3, a między słowami 010 i 011 – wynosi 1. Odległość dowolnego słowa kodowego od słowa powstałego przez zmianę pojedynczego bitu wynosi 1.
Dla zilustrowania różnicy pomiędzy możliwością wykrycia i korekty błędu przyjmijmy, że osiem wiadomości kodujemy następująco: 0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111. Odległość pomiędzy dowolnymi dwoma słowami jest zawsze większa lub równa 2. (Sprawdzić!!!) Przypuśćmy, że na skutek błędu transmisji odebrano słowo 0111. Nie jest to słowo ze słownika, więc łatwo można wykryć, że powstał błąd. Nie można jednak stwierdzić jaki był nadawany sygnał. Odebrane słowo różni się o 1 aż od 4 słów (1111, 0011, 0101, 0110). Stąd wniosek, że dla umożliwienia korekty pojedynczego błędu odległość pomiędzy poprawnymi słowami kodowymi powinna wynosić co najmniej 3. Uogólniając można stwierdzić, że dla minimalnej odległości między słowami równej D_{min} można wykryć co najwyżej $\boldsymbol{D}_{min}$-1 błędów. Jeżeli $\boldsymbol{D}_{min}$ jest liczbą parzystą to można poprawić $\boldsymbol{D}_{min}/2 - 1$ błędów. Jeżeli zaś $\boldsymbol{D}_{min}$ jest liczbą nieparzystą można poprawić $(\boldsymbol{D}_{min}-1)/2$ błędów.

Metody tworzenia kodów

Najprostrzą metodą zwiększenia odległości między słowami jest powtarzanie. Używając metody powtarzania do przykładowego zestawu ośmiu wiadomości uzyskamy osiem słów kodowych: 000000, 001001, 010010, 011011, 100100, 101101, 110110, 111111. Minimalna odległość pomiędzy słowami wynosi 2. Stworzony kod sześciobitowy ma więc dokładnie takie same właściwości jak pokazany uprzednio kod czterobitowy. Można więc zaryzykować twierdzenie, że metoda powtarzania nie jest metodą wydajną.
Przedstawiony wcześniej kod czterobitowy był kodem algebraicznym. Został on wygenerowany poprzez dodanie czwartego bitu do wiadomości trzybitowej tak, by liczba jedynek w słowie była parzysta. Innymi słowy dodany bit był bitem parzystości. Metodę tę można uogólnić. Załóżmy, że wiadomości mają długość m bitów. Mamy zatem zbiór 2^m różnych wiadomości. Rozważmy zatem kod powstały poprzez dodanie n bitów parzystości. Każde słowo będzie miło długość $m+n$ i postać: $a_1\ a_2\ a_3 \ldots a_m\ c_1\ c_2\ c_3 \ldots c_n$, przy czym a_i oznacza bit wiadomości a c_i bit kontroli parzystości. Bity c_i są definiowane tak, by stanowiły bit parzystości dla pewnych bitów wiadomości. Stosując algebrę Bool'a i działania na macierzach warunek jaki muszą spełniać bity parzystości można zapisać następująco:

$$\boldsymbol{H}\,\boldsymbol{T} = 0$$

przy czym $\boldsymbol{T}$ jest $(m + n)$ wymiarowym wektorem:

$$\boldsymbol{T} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ . \\ . \\ . \\ a_m \\ c_1 \\ c_2 \\ c_3 \\ . \\ . \\ . \\ c_n \end{bmatrix}$$

$\boldsymbol{H}$ jest $n \times (n + m)$ wymiarową macierzą o postaci:

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & ... & h_{1m} & 1 & 0 & 0 & 0 & ... & 0 \\ h_{21} & h_{22} & ... & h_{2m} & 0 & 1 & 0 & 0 & ... & 0 \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ h_{n1} & h_{n2} & ... & h_{nm} & 0 & 0 & 0 & 0 & . & 1 \end{bmatrix}$$

Operacje bitowe są zawsze wykonywane modulo 2. Elementy h_{11} do h_{nm} przybierają wartości 0 lub 1. Oglądając rozwinięcie równania wektorowego:

$$\begin{array}{c} h_{11}\, a_1 + h_{12}\, a_2 + \ldots + h_{1m}\, a_m + c_1 = 0 \\ h_{21}\, a_1 + h_{22}\, a_2 + \ldots + h_{2m}\, a_m + c_2 = 0 \\ . \quad . \quad . \quad . \\ . \quad . \quad . \quad . \\ . \quad . \quad . \quad . \\ h_{n1}\, a_1 + h_{n2}\, a_2 + \ldots + h_{nm}\, a_m + c_n = 0 \end{array}$$

Elementy $h_{j1} \ldots h_{jm}$ można zinterpretować jako określające które bity wiadomości są brane pod uwagę przy obliczaniu bitu parzystości c_j.
Dla kodu algebraicznego o 4-bitowych wiadomościach i trzech bitach kontrolnych macierz $\boldsymbol{H}$ zdefiniujmy jako:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Słowa kodowe wyglądają następująco:

0000	0000000
0001	0001111
0010	0010011
0011	0011100
0100	0100101

0101	0101010
0110	0110110
0111	0011001
1000	1000110
1001	1001001
1010	1010101
1011	1011010
1100	1100011
1101	1101100
1110	1110000
1111	1111111

Przy odbiorze słowa kodowego oblicza się jego iloczyn z macierzą ***H***. Jeżeli wynik tej operacji jest równy 0, to otrzymane słowo jest poprawne. Jak znaleźć miejsce wystąpienia błędu?
Oznaczmy wektor wiadomości nadawanych przez ***T*** i zdefiniujmy wektor błędu przez ***E***, jako zawierający jedynki na wszystkich pozycjach, na których występuje błąd. Wektor wiadomości odebranych ma zatem postać:

$$\boldsymbol{R} = \boldsymbol{T} + \boldsymbol{E}$$

Odbiornik sprawdzając otrzymaną wiadomość dostaje:

$$\boldsymbol{H}\,\boldsymbol{R} = \boldsymbol{H}\,(\boldsymbol{T} + \boldsymbol{E}) = \boldsymbol{H}\,\boldsymbol{T} + \boldsymbol{H}\,\boldsymbol{E} = \boldsymbol{H}\,\boldsymbol{E}$$

Iloczyn ***H E*** należy porównać z kolumnami macierzy ***H***. Numer kolumny mającej postać identyczną z iloczynem ***H E*** wskazuje numer bitu na którym wystąpił błąd. Uwaga! Aby wspomniana metoda mogła służyć korekcie błędów należy konstruować macierz w ten sposób, by nie miała identycznych kolumn, ani kolumny zerowej. Jeżeli błąd wystąpi na więcej niż jednej pozycji, to iloczyn ***H E*** przyjmie postać sumy odpowiednich kolumn macierzy ***H***. Można więc wyciągnąć wniosek, że do korekcji dwóch błędów kolumny macierzy dodatkowo muszą mieć taką postać, by żadna z kolumn nie była sumą dwóch innych. Łatwo widać (Sprawdź!), że przykładowa macierz nie spełnia tego warunku.

Zadanie

1. Opracować kod korygujący pojedynczy błąd bitowy dla wiadomości ośmiobitowych (1 bajt)
2. Opracować kod korygujący podwójny błąd bitowy dla wiadomości ośmiobitowych (1 bajt)
3. Napisać program przekodowujący dowolny plik do postaci zakodowanej jednym z opracowanych kodów (przygotowanie do transmisji) i odkodowywujący do postaci pierwotnej (odtworzenie danych po transmisji) z korekcją powstałych błędów. Dla ułatwienia zadania można kodować dane z upakowaniem na granicy 1 bajtu, czyli słowa o długości od 9 do 16 bitów jako 2 bajty, 17-24 bity jako 3 bajty itd. Operacje kodowania/odkodowania powinny być uruchamiane niezależnie od siebie, tak by można było poprzez ręczną modyfikację pliku zasymulować powstanie błędów transmisji.