Dasar-Dasar Pemrograman 2

Tugas Pemrograman 1

Hamming Code

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Kode-Kode Privad

Salman

Alkisah, terdapat sepasang pemuda pemudi bernama Salman dan Dina, mereka sangatlah dekat dan seringkali berbagi keluh kesah masing masing secara privat. Namun, karena dinamika tempat mereka berkuliah terbilang cukup sulit dan sibuk, mereka seringkali bersama teman-teman mereka untuk mengerjakan tugas perkuliahan sehingga waktu mereka untuk mereka berbagi keluh kesah semakin sedikit.

Melihat permasalahan tersebut, mereka berusaha untuk mencari cara agar mereka dapat membagikan keluh kesah mereka masing masing secara privat walaupun mereka sedang bersama teman teman mereka. Mereka berpikir "apakah kita berkomunikasi dengan sistem bilangan biner?!" tapi setelah mereka berpikir lebih lanjut, itu saja tidak cukup karena teman-teman mereka juga memahami angka biner. Akhirnya, mereka memutuskan untuk membuat sebuah sistem kode enkripsi biner bernama "Hamming Code".

Mereka berdua telah bisa membuat bahasa tersebut beserta aturannya. Salah satu dari mereka akan mengubah pesan yang akan dia sampaikan ke dalam bentuk biner lalu mengenkripsikannya (*encode*) dengan sistem Hamming Code. Satunya lagi akan menerjemahkan (*decode*) kode yang diterimanya ke dalam bentuk biner dari pesan aslinya. Sistem kode tersebut juga memiliki kelebihan, sistem kode tersebut bisa mendeteksi serta mengoreksi apabila penerima kode mendapatkan kode yang salah karena adanya gangguan ketika proses pengiriman kode seperti suasananya yang berisik, Salman yang salah mendengar karena mengantuk, teman-teman yang iseng dan lain lain.

Akan tetapi, peraturannya cukup sulit untuk diimplementasikan apabila hanya menggunakan akal. Oleh karena itu, mereka merasa kesulitan. Mereka ingin meminta bantuan teman-teman mereka, tetapi mereka berpikir "Kalau minta bantuan mereka *mah* percuma aja, *ngerti ngerti* juga *dong* mereka bahasanya".

Mereka pun memutuskan untuk meminta bantuan kepada Dek Depe. Namun, ternyata Dek Depe belum mengambil mata kuliah PSD maupun DDAK, sehingga ia tidak paham mengenai Hamming Code. Dek Depe bertanya kepada Pak Es De, "Siapakah orang-orang yang sudah paham mengenai Hamming Code?"

Pak Es De bilang, ada suatu keluarga bernama keluarga Meong yang sudah paham mengenai Hamming Code. Keluarga Meong berisi orang-orang hebat, di antaranya adalah kalian, peserta mata kuliah DDP 2. Akhirnya, Dek Depe meminta kalian untuk membuatkan program untuk Salman dan Dina yang dapat mengubah bahasa biner dari pesan yang ingin disampaikan menjadi pesan dengan sistem "Hamming Code" dan sebaliknya agar mereka bisa berkomunikasi dengan bahasa tersebut. Akankah mereka bisa berbagi keluh kesah melalui cara tersebut? Jawabanya ada di tangan kalian, orang-orang hebat.

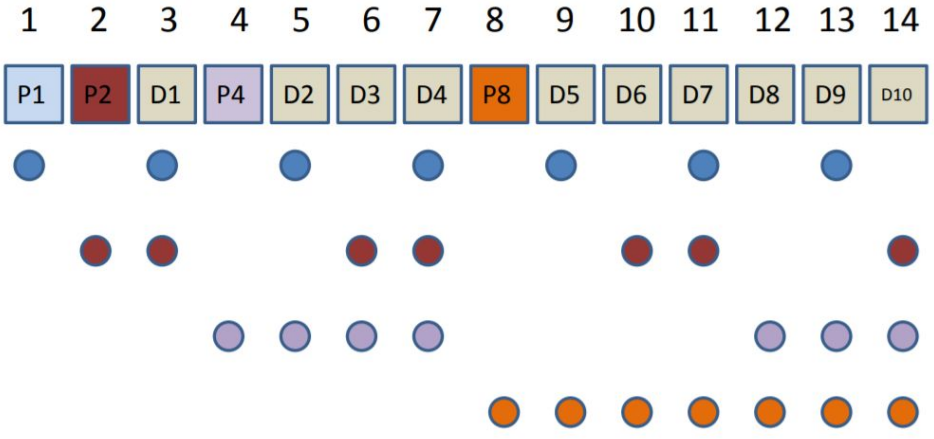
(Credit cerita: GBP dan SMA)

Hamming Code

Hamming Code adalah sebuah *code* yang dapat digunakan untuk memperbaiki *single-bit errors*. Artinya, jika **tepat satu bit** dari beberapa bit yang diterima ternyata salah, kita bisa mengetahui dan mengembalikan data yang benar menggunakan Hamming Code. Hamming Code bekerja seperti berikut.

1. Cari **r**, yaitu banyaknya bit redundan (berlebih) yang akan ditambahkan dengan cara mencari **r** sedemikian hingga $2^r \geq m + r + 1$ dengan **m** adalah banyaknya bit data sebenarnya.
2. Letakkan bit-bit redundan di posisi dua pangkat, yaitu 1, 2, 4, 8, …
3. Cari nilai bit-bit redundan sedemikian hingga:
    a. Jumlahan bit-bit yang berada pada posisi 1, 3, 5, 7, 9, 11, 13, 15… adalah genap. Pola posisi: cek 1 bit, *skip* 1 bit.
    b. Jumlahan bit-bit yang berada pada posisi 2, 3, 6, 7, 10, 11, 14, 15, 18, 19… adalah genap. Pola posisi: cek 2 bit, *skip* 2 bit.
    c. Jumlahan bit-bit yang berada pada posisi 4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23, … adalah genap. Pola posisi: cek 4 bit, *skip* 4 bit.
    d. dan seterusnya, hingga tidak ada bit yang belum di-*cover*.

Gambaran untuk bagaimana bit-bit redundan dan bit-bit asli ditempatkan:



Sumber gambar: *slide* kuliah PSD

Keterangan: P1, P2, P4, dan P8 adalah bit redundan, sedangkan D1, D2, D3, D4, … , D10 adalah bit-bit asli dari data yang kita miliki.

Sebagai contoh, mari kita ambil data **10011010**.

1. Banyaknya data (m) adalah 8. Ambil nilai r = 4, karena 4 adalah nilai terkecil yang memenuhi pertidaksamaan $2^4 \geq 8 + 4 + 1$.
2. Letakkan bit-bit redundan tersebut di posisi dua pangkat, sebagai berikut :
   _ _ 1 _ 0 0 1 _ 1 0 1 0
3. Cari nilai dari bit-bit redundan.
   a. Posisi 1: ? _ 1 _ 0 0 1 _ 1 0 1 0. Yang ditandai merah adalah bit-bit yang kita cek saat ini dan jika dijumlahkan harus bernilai genap. Maka, bit redundan di posisi pertama ini bernilai 0.
   b. Posisi 2: 0 ? 1 _ 0 0 1 _ 1 0 1 0. Yang ditandai merah adalah bit-bit yang kita cek saat ini dan jika dijumlahkan harus bernilai genap. Maka, bit redundan di posisi kedua ini bernilai 1.
   c. Posisi 4: 0 1 1 ? 0 0 1 _ 1 0 1 0. Yang ditandai merah adalah bit-bit yang kita cek saat ini dan jika dijumlahkan harus bernilai genap. Maka, bit redundan di posisi keempat ini bernilai 1.
   d. Posisi 8: 0 1 1 1 0 0 1 ? 1 0 1 0. Yang ditandai merah adalah bit-bit yang kita cek saat ini dan jika dijumlahkan harus bernilai genap. Maka, bit redundan di posisi kedelapan ini bernilai 0.

Dari tahapan-tahapan tersebut, didapat hamming code untuk data **10011010** adalah **011100101010**.

Bagaimana jika kita memiliki sebuah hamming code dan ingin mendapatkan datanya?

Misalkan hamming code yang diterima adalah **01110010111**0**,** yaitu hamming code yang diterima pada hasil sebelumnya. Namun, bit ke-10 salah karena seharusnya 0, tetapi yang diterima 1.

Cek menggunakan cara serupa pada proses pembuatan hamming code (*encoding*). Jika data benar, maka semua penjumlahan pada P1, P2, P4, dan P8 yang bersesuaian masing-masing akan bernilai genap. Namun, pada *code* ini:

1. Cek bit redundan 1: **0111001011 10.** Penjumlahan bit yang ditandai merah adalah 4 (genap).
2. Cek bit redundan 2: **0 11 1001 01 11 10.** Penjumlahan bit yang ditandai merah adalah 5 (**ganjil**).
3. Cek bit redundan 4: **011 1001 0111 0.** Penjumlahan bit yang ditandai merah adalah 2 (genap).
4. Cek bit redundan 8: **01110010 1110.** Penjumlahan bit yang ditandai merah adalah 3 (**ganjil**).

Posisi yang bersesuaian dengan P2 dan P8 salah. Dengan demikian, kita tahu bahwa bit ke $2+8 = 10$ salah. Karena bit ke-10 pada *code* tersebut adalah 1, maka bit yang seharusnya adalah 0. Dengan demikian, *code* yang benar adalah **011100101010**. Untuk mendapatkan data aslinya, cukup hilangkan bit-bit redundan yang berada pada posisi 1, 2, 4, 8, ... (dua pangkat). Jadi, **data** pada *code* **011100101010** adalah 10011010.

Pada tugas pemrograman kali ini, kalian diminta membuat sebuah program yang dapat mengubah data menjadi *code* dan mengubah *code* menjadi data semula. Implementasikan kedua fungsi tersebut berturut-turut pada *method* `encode` dan `decode`.

Contoh:
`HammingCode.encode("10011010")` mengembalikan `"011100101010"`.
`HammingCode.encode("1011001")` mengembalikan `"10100111001"`.
`HammingCode.decode("011100101110")` mengembalikan `"10011010"`.
`HammingCode.decode("01110110101")` mengembalikan `"1001101"`.
(mendeteksi error pada bit ke-6 dan otomatis memperbaikinya)
Catatan: kedua *method* tersebut **mengembalikan String, bukan mencetak (print) String**.

Untuk mengetahui kebenaran program yang kamu buat, ada dua cara yang dapat dilakukan. Pertama, coba tiap kasus yang ingin kamu uji menggunakan program main yang telah disediakan. Cara yang lain, kamu dapat menjalankan

```
# di Windows
> gradlew.bat test
# di Linux/Mac
$ ./gradlew test
```

dan melihat apakah ada test yang gagal.

Penilaian:

60% kebenaran program

30% demonstrasi program

10% kerapian dan dokumentasi

+5% bonus kesesuaian kode dengan aturan Checkstyle yang diberikan