18th Annual South Jersey Regional High School Programming Contest

hosted by the

Rowan University Computer Science Department

Saturday, 24 April 2004

Contest Problem

1 Background

At the South Jersey Regional High School Programming Contest, competitors are assigned anonymous unique ID codes, to prevent possible bias in the judging. At our 17th contest, held on 12 April 2003, there was a problem with awarding the 2nd place prizes. The ID codes *hspc0037* and *hspc0038* were incorrectly recorded as *hspc0027* and *hspc0028*, an error discovered during the annual post-contest audit. As might be expected, this led to significant extra work which should never have been necessary.

Had any of several conditions been different, the error would have been discovered, and the problem easily avoided. For example, had hspc0027 been paired with hspc0026, there would have been no team hspc0027+hspc0028, and the discrepency would have been noticed. But the registration system wasn't designed to avoid such overlaps; even it it had been, a last-minute substitution might cause an overlap to appear. A thorough effort to avoid errors must take such things into account. Error detection and correction being an important topic in computer science, the difficulties at last year's contest inspired this year's problem.

One method of preventing undetected errors is based on a concept known as 'Hamming distance', named after Richard W. Hamming, an engineer at AT&T, who described it in 1950[1]. The Hamming distance between two binary numbers is the number of bit positions which differ. For example, '110' and '100' have a Hamming distance of 1, since they differ in a single position. Similarly, '101' and '011' have a Hamming distance of 2. A Hamming distance of 0, obviously, means that two numbers are identical.

This concept can be applied to groupings of any length, with any set of symbols; the Hamming distance between $\phi\acute{o}\beta o\varsigma$ and $\phi\acute{o}\rho o\varsigma$ is 1. This concept also gives us a way to avoid errors. Many of last year's competitor IDs differed in only one position: *hspc0027* and *hspc0037* have a Hamming distance of 1. With pairs of strings that have a larger Hamming distance, it is impossible to incorrectly interpret one symbol and thus cause an undetected error: if one character in the ID is wrong, the result will be a nonexistent ID.

This concept can also be applied to groups larger than two. The Hamming distance of a group will be the *smallest* distance between any two elements. The Hamming distance of the set {0001 6784 0004} is 1, because that's the distance between the first and third elements. The second string, of course, has a Hamming distance of 4 with the first, and 3 with the second – but for the purposes of avoiding undetected errors, the first and third strings could be mistaken for each other with only a single error.

There are other difficulties which have to be addressed:

1. Some symbols may be confused when pronounced, such as *D* and *T*, or *H* and *8*. This can include *0* and *O*, as many people say 'oh' for 'zero'.

2. Some symbols may be confused when written, such as *S* and *5*, or *0* and *O*, or *1*, *I*, and *l*.

Your task for this contest is to write a program which will read in data sets of equal-length strings and report on the Hamming distance of each set, taking into account symbols which might be easily confused.

Your program will be given a number of data sets. Each data set will comprise a list of symbols to be used, a set of pairs indicating which symbols should be considered alike, and a set of strings to be compared. For each data set, your program will have to report on the number of strings, their length, and the Hamming distance of the strings.

[1] R. W. Hamming, 'Error Detecting and Error Correcting Codes', *The Bell System Technical Journal*, 26(2):147-160, April 1950.

2 Input/Output Specification

2.1 Input

For text input, your program should accept input in the following format:

1. A single integer, $\mathcal{D}$, where $1 \leq \mathcal{D}$, which specifies the number of data sets to be scanned.
2. $\mathcal{D}$ data sets, each of which is in this format:
   (a) A single integer, $\mathcal{S}$, where $1 \leq \mathcal{S} \leq 100$, which specifies the number of symbols to be used in the data set.
   (b) $\mathcal{S}$ characters on one line, separated by spaces, which are the characters to be used for this data set.
   (c) A single integer, $\mathcal{P}$, where $0 \leq \mathcal{P} \leq 100$, which specifies the number of symbol pairs which will be considered potentially confusing for this data set.
   (d) $\mathcal{P}$ lines, each with two characters separated by a single space, which define pairs of characters to be considered potentially confusing.
   (e) A single integer, $\mathcal{N}$, where $1 \leq \mathcal{N} \leq 100$, which specifies the number of strings in the data set, each of which will be the same length.
   (f) A single integer, $\mathcal{C}$, where $1 \leq \mathcal{C} \leq 80$, which specifies the number of characters in each string.
   (g) $\mathcal{N}$ lines, each with exactly one string of length $\mathcal{C}$.

You need not do error-checking on the input. There will be nothing on the first line except a single integer. For each data set, there will be nothing on the first line except a single integer, and nothing on the next but the correct number of characters separated by spaces. The third line will have only a single integer, and the following lines only two characters separated by a single space. After the symbol set is defined, there will be nothing on the next two lines except for the string count and string length, and each line of data will have exactly $\mathcal{C}$ characters. There will be no blank lines.

You may choose to have your program read the input from the keyboard, or ask the user for a filename and then read the file. Users of GUI-based programming environments may prefer to use text boxes into which the strings can be entered, and buttons to begin their calculation. Any reasonable variation in the spirit of the problem is acceptable.

2.2 Output

For each data set, your program must generate output according to the following description:

1. The text **Data set *S*:**, indicating which data set is being reported on.
2. A line with the text **Number of strings: *N***, indicating how many strings were read.
3. A line with the text **Length of strings: *L***, indicating how long each string was.
4. A line with the text **Hamming distance: *H***, reporting on the minimum number of differences between any two pairs of strings.

Your output does **not** have to duplicate the sample output as regards spacing or use of upper/lower case. Your output should be neat, but need not exactly match the sample.

3 Sample Data

3.1 Sample Input

```
3
10
0 1 2 3 4 5 6 7 8 9
2
1 9
5 9
3
4
0921
3145
4628
20
A B C D E F G H I J K L M N O P Q R S T
6
B D
B P
B T
D P
D T
P T
4
5
ABCDE
FGHIJ
KLMNO
ABCDE
2
0 0
1
0 0
2
4
0000
0000
```

3.2 Sample Output

```
Data set 1:
    Number of strings: 3
    Length of strings: 4
    Hamming distance:  3

Data set 2:
    Number of strings: 4
    Length of strings: 5
    Hamming distance:  0

Data set 3:
    Number of strings: 2
    Length of strings: 4
    Hamming distance:  0
```

(Your floppy disk has a copy of the sample input in the file named **sample.txt**.)

4 Test Data

Run your program on this input and make screenshots showing the output. Your program will also be tested on data known only to the judges.

```
4
2
0 1
0
5
8
01010010
01101111
01110111
01100001
01101110
13
0 1 2 3 4 5 6 7 8 9 F M R
2
1 9
5 9
8
9
F0329M41R
87F5R3960
12468R789
9873428F3
M651F9308
238F07RM7
R46M7827F
39146RM24
4
5 S 2 Z
2
5 S
2 Z
4
50
S55ZZS2SZ222ZZ55SSZSZS2S5Z5ZSSSZ22SZ25SSZZ25ZSSZ22
SSZSZZS552SSSZS255222S255222Z5ZS2SSS2SZ5Z2ZSSZ22ZZ
S5S22ZZ225252SSZZS55SSZ55Z2S5ZSS2S222ZSZ5Z55Z52SS2
ZZ2S5ZZ2ZZ25Z5S22255S55S2S2ZZ55ZS255SS22Z2ZS2Z5255
11
A B F H I L M O U R X
0
9
70
LABAUIRRXOUHHRHOFABRMOMRFRHAXIOXULIIMRIHOBFAFIXMOFOXLXBRXXULUAIFHMRHUO
IRFLLURFOBXHLXOBUFBFHIUBOLLUUBBAAMMAFUFMIOBLXARLFBLFMOURIUOULLHOIXMFBM
IHUALLBARLFIUIHFIIMHUILFHLMXLUOXMAROBMMRUBUOLFOIOOUAOXHUUFMHXXIFAAARLH
HHHXULLUROURFFLFXRHMXMRBXMOBOBRLOIBHHUORLFIMAUUFLAXHHHXXHMIMBUIULBXRIM
MIIUOMXAABBFHLUFXRIOHRXUMIIUAIXULIXHFBXRFUBHILBHXMHRUXMMIXFRMUHOIXBIFI
IIFIXLXORFIRLMBFRABBXHHBHIMOHLLMRLRALIFHLUXXLRFOFMMLFMHOFUXMIRBOAOUXLR
AHRBMXFIAXUBUUFLLFRMBFLLAIRXRFFXORRIAABHXOFHOFRAHBXMOMFBMFURLFUMLLBULM
HHBXBRRXLIBFHHBLXFHHBMBRBRRRUMMLLAMFIUORIAUMHFBLARFFIUOHUOFXXAOHHMFHLF
OIHFXRHBHBLFMFBRHIHALRMFXMMOMOFRILFUULRFHRFFIALOMHBUFHXMHLARLUIOAMRFMM
```

(Your floppy disk has a copy of the sample input in the file named **testdata.txt**.)

5 Notes

Richard Hamming's original paper is at: <http://www.engelschall.com/~sb/hamming/>. His geometric model is described at the beginning of Part II; the visualization in terms of a multidimensional cube makes plain the use of 'distance' to describe differences between two groupings. (NB: many citations of this paper incorrectly list it as being in Volume 29.)

In general, if one chooses strings of length $\mathcal{L}$ from an alphabet of size $\mathcal{S}$, there are $\mathcal{S}^{\mathcal{L}}$ unique arrangements. (Thus there are 100 possible 2-digit numbers, because $10^2 = 100$.) However, many of these arrangements will have a Hamming distance of only 1 (such as 10 and 11). In the case where there is an additional restriction of a minimum Hamming distance $\mathcal{D}$, where $\mathcal{D} > 1$, the number of possibilities drops to $\mathcal{S}^{\mathcal{L}-(\mathcal{D}-1)}$. (Note that if $\mathcal{D} = 1$, this formula is the same as $\mathcal{S}^{\mathcal{L}}$.) Thus, if we require a Hamming distance of 2 for 2-digit numbers, we cannot choose a set with more than 10 elements. Working out a proof of this claim may be a productive exercise.

A. K. Dewdney's book *The New Turing Omnibus - 66 Excursions in Computer Science* has essays on many subjects which may be of interest to competitors at this contest; Hamming distance is mentioned in essays 12 and 49, titled 'Error-Correcting Codes' and 'Shannon's Theory'.

Another standard method of error detection is a *checksum*, in which transmitted data are encoded numerically in some way, a mathematical operation is applied by the sender, and the data are sent along with the results of the calculation. The receiver independently performs the same mathematical operation on the data, and compares the result with the result as transmitted. If a discrepency is discovered, the receiver can request that the data be re-sent. Checksums, while somewhat burdensome for human use, are commonly used in computer communication. The Internet Protocol, used by every computer which connects to the Internet, specifies a checksum for each data packet. The IETF specification for the Internet Protocol can be found at <http://www.ietf.org/rfc/rfc0791.txt>.

After the widespread adoption of telecommunications in the military services, the use of spoken alphabets (also called 'phonetic alphabets') was adopted to reduce errors. Unfortunately, during World War II, there were problems with communication between various allied military forces, because there was no international standard. After the war, the allied forces adopted a phonetic alphabet using the words able, baker, charlie, dog, easy, fox, george, how, item, jig, king, love, mike, nutley, oboe, peter, queen, roger, sail, tare, uncle, victor, william, x-ray, yoke, zebra.

Some of these words were inappropriate for non-English speakers, and after several revisions a common alphabet was adopted by the International Civil Aviation Organization (ICAO), the International Telecommunications Union (ITU), and NATO. The standard phonetic alphabet uses the words alpha, bravo, charlie, delta, echo, foxtrot, golf, hotel, india, juliett, kilo, lima, mike, november, oscar, papa, quebec, romeo, sierra, tango, uniform, victor, whiskey, x-ray, yankee, zulu.

The problem of pronunciation extends to numbers as well, and the standard pronunciation guide for numbers most notably specifies *niner* for 9.

The Federal Aviation Administration has many publications on-line; a complete pronunciation guide can be found in table 2-4-1 of the publication on Air Traffic Control, in the section titled 'Radio Communications': <http://www1.faa.gov/ATpubs/ATC/Chp2/atc0204.html>.