Algorithms

Hamming Distance Handout

1 Introduction

Hamming distance is a metric for comparing two binary strings of equal length, we will discuss hamming distance in depth below. It has applications all over the field of computer

The minimum hamming distance problem is defined as follows: Given an array S of k binary strings of length n. Find the string y such that for $\forall x \in S$ you minimize the maximum hamming distance $H(x, y)$.

2 Example

2.1 Hamming Distance

Given two binary strings of length 10 we can calculate the hamming distance by counting the number of bits that are different. Given the two strings below:

```
0100111001
1100110101
```

The hamming distance is 3, because the binary strings differ in 3 places, at indices 0, 6, and 7.
For the example where the strings are exactly the same:

```
0100111001
0100111001
```

The hamming distance is 0.
For the example where the strings are complements (opposite) of each other:

```
0100111001
1011000110
```

The hamming distance would be 10, the length of the strings.

2.2 Minimum Hamming Distance

Given the following set of 3 binary strings of length 4:

```
{1100, 1011, 1101}
```

Find the string that minimizes the maximum hamming distance over all strings. For this example the optimal string is

```
1101
```

which has a maximum hamming distance of 1.

3 Solution Methods

3.1 Exhaustive

Generate every possible binary string of length n. Then for each string in that list you find the maximum hamming distance for that string over all strings in your given set. store the string that minimizes the maximum hamming distance. Repeat until every possible string has been tested.

3.2 Greedy Heuristic

Initialize an array of zeros of length n. Treat the values in the binary strings as separate integers. For each string in the given set add the value at index to it's respective index in the array, continue until all values have been used. Then divide each value by the number of strings you were give. This should force the values in the array to be in the range [0,1]. Now iterate through the array and if the value is strictly less than 0.5 set it to 0, otherwise set it to 1. now calculate the maximum hamming distance for this new binary string.

4 Pseudocode for exhaustive and greedy-heuristic approaches

Algorithm 1: Exhaustive Approach

Data: k the length of the binary strings,
n the number of binary strings,
S the given set of binary strings
Result: *ham* the minimum max hamming distance,
minStr the binary string which minimizes the maximum hamming distance over the set

```
allStrings= All possible binary strings of length k;
ham=∞;
minString="";
for string1 in allStrings do
    currMax = 0;
    for string2 in S do
        hamDist=HammingDistance(string1, string2);
        if hamDist > currMax then
            currMax=hamDist;
        end
    end
    if currMax < currMin then
        currMin = currMax;
        minStr=string1;
    end
end
```

Algorithm 2: Greedy-Heuristic Approach

Data: k the length of the binary strings,
n the number of binary strings,
S the given set of binary strings
Result: *ham* the minimum max hamming distance,
minStr the binary string which minimizes the maximum hamming distance over the set

```
newString = array of zeros of length k;
for string in S do
    for i in [0,k) do
        newString[i]+=string[i];
    end
end
for value in newString do
    if value/n ≥ 0.5 then
        value =1;
    else
        value=0;
    end
end
minStr=string(newString);
ham= 0;
for string in S do
    hamDist=HammingDistance(minStr, string);
    if hamDist > ham then
        ham=hamDist;
    end
end
```