Homework4:

1.

2.

3.Huffman：

Before compression, the String”It is a sunny day today” needs 23\*8 = 184bits,

after compression, it needs 76 bits, the ratio of compression is 41.3%

Psuedocode:

define a method named printCode(HuffmanNode root, String s){

start;

if the left node of root is null and the right node of root is null and c of root is letter:

output :c of root + s;

end;

else:

use this method:

printcode(root.left, s+”0”);

printcode(root.right,s+”1”);

end;}

Define char[] chararray；

Define int[] charfreq ；

Define n = chararray.size()

Define PriorityQueue<HuffmanNode>q;

for(i =0; i<n;i++){

define: HuffmanNode hn,

make hn.c = chararray[i];

make hn.data = charfreq[i];

put this hn into q

}

Define a HuffmanNode root is null;

when (the size of q more than 1):

loop:

Define HuffmanNode x = the head of q;

let it deQueue;

Define HuffmanNode y = the head of q;

let it deQueue;

Define HuffmanNode f;

make data of f = data of x + data of y;

let c of f is ‘-‘;

let left node of f is x, right node of f is y

and make root is f;

put f into q again;

}

when loop ends;

printCode(root, “”);

ends;

}

}

In this algorithm, we use the priorityQueue for storing the Node and we make the nodes in order according to their frequency, Every iteration we need to poll two smallest nodes in Queue and combine their frequency together in one Node , then offer it into queue, and encode Huffman code by this way, until the last Node in the queue, then we will finish the encoding process.

Before encode

I 11001

t 1111

01

i 0011

s 000

01

a 101

01

s 000

u 0010

n 1110

n 1110

y 100

01

d 1101

a 101

y 100

01

t 1111

o 11000

d 1101

a 101

y 100

The length of Huffman Code is 76

I : 01001001

t : 01110100

: 00100000

i : 01101001

s : 01110011

: 00100000

a : 01100001

: 00100000

s : 01110011

u : 01110101

n : 01101110

n : 01101110

y : 01111001

: 00100000

d : 01100100

a : 01100001

y : 01111001

: 00100000

t : 01110100

o : 01101111

d :01100100

a : 01100001

y : 01111001

After Encode

I 11001

t 1111

01

i 0011

s 000

01

a 101

01

s 000

u 0010

n 1110

n 1110

y 100

01

d 1101

a 101

y 100

01

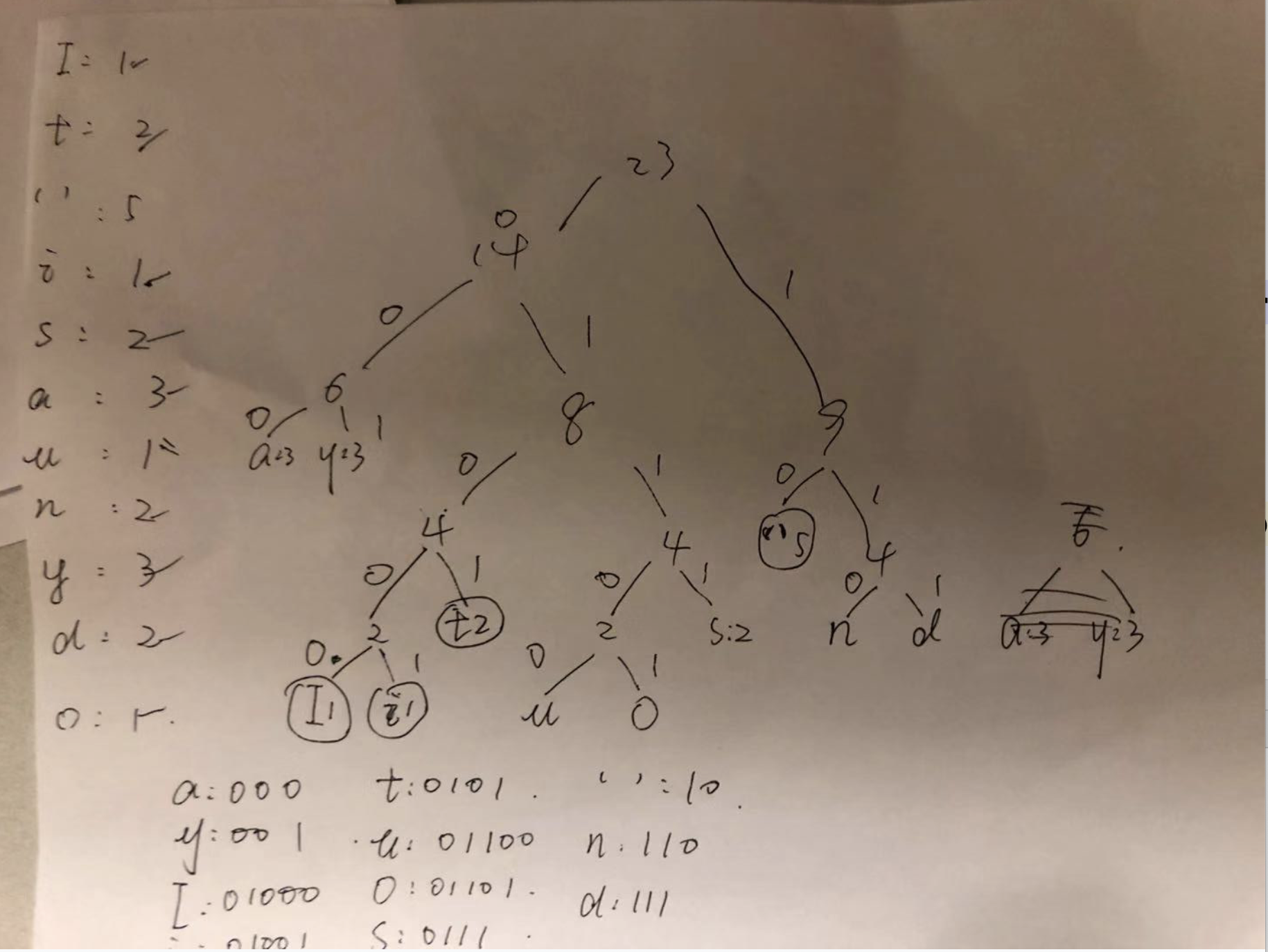
t 1111

o 11000

d 1101

a 101

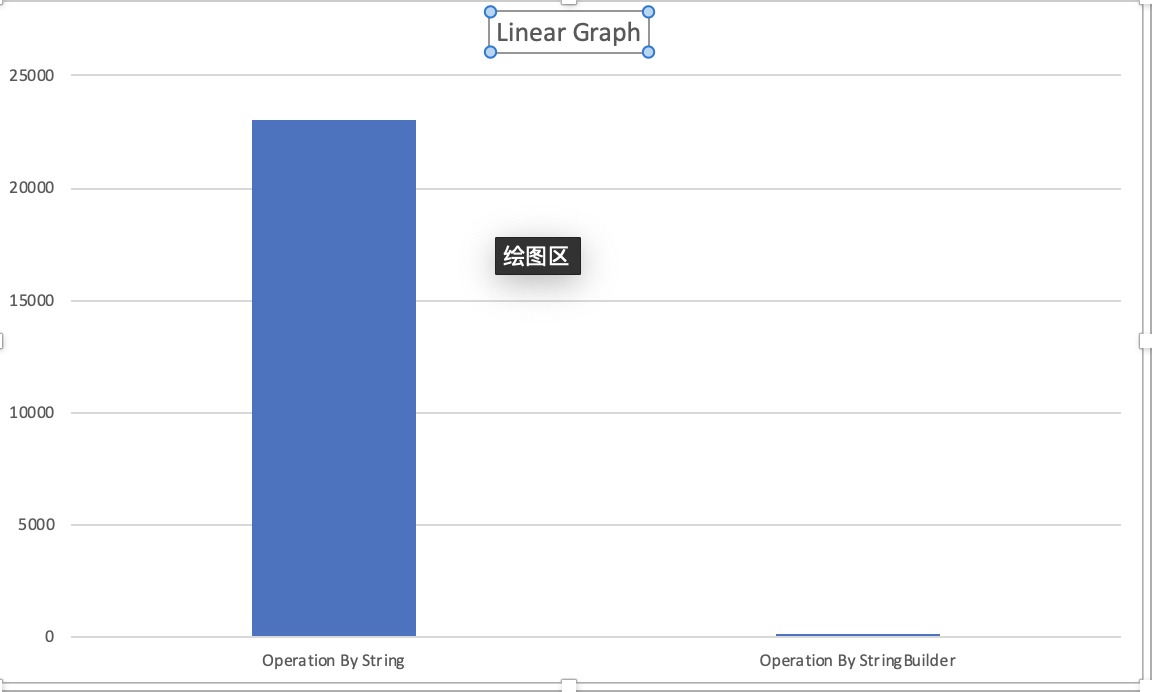
y 100

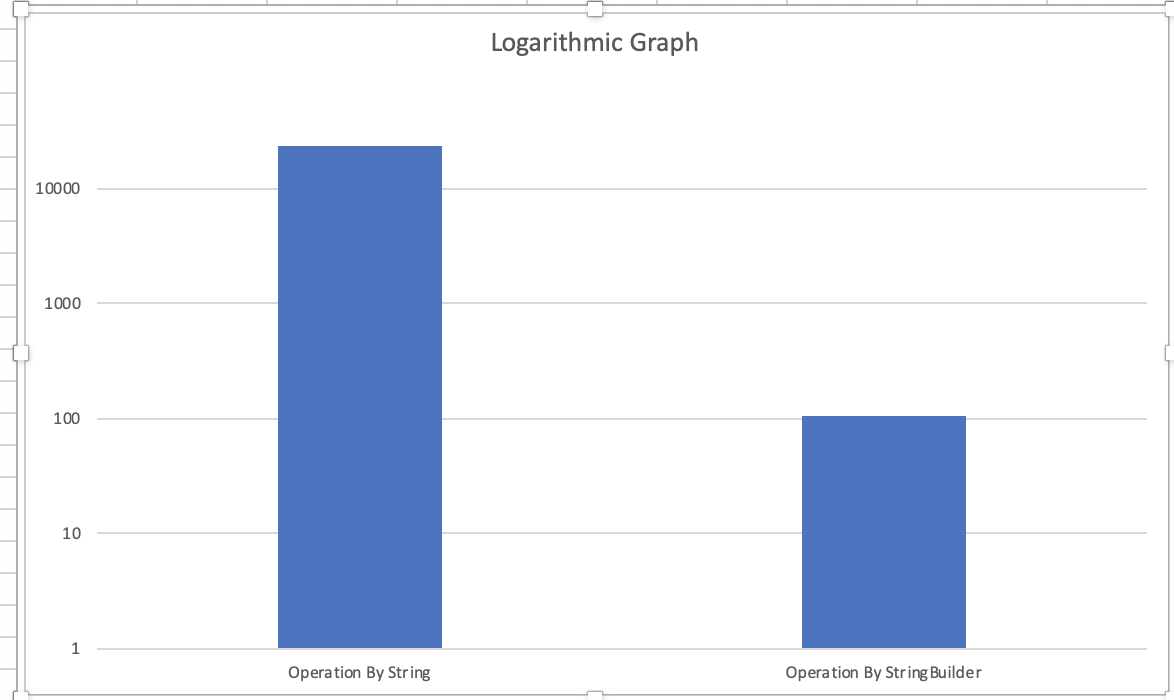


4.The time Complexity of Operation in String is the times of iteration\*O(N^2)

The time Complexity of Peration in StringBuilder is the times of iteration \*O(N);

:N is length of String given





The Y-axis- ms.

5. The User class implements Comparable Interface, and has 3 private members: String name, int id, Date birth. The constructor is general mode, distribute value of each parameter to translated members.

Override a method named equals with the parameter Object “other” and return a boolean data:

if Object other is the User itself, return true directly;

and if other is referred to null or other is not instance of class User, return false directly;

if we can get here:

We define an User object named guest, which is cast from other:

then we have 3 requirements :

id of guest is equal to id of itself

name of id is equal to name of itself

birth of id is equal to birth of itself:

return true;

if any requirement failed , this method will return false;

Override hashCode()method:

define a int data named result:

give the value of id to result

and let result multiple 31 then plus hashCode of member name;

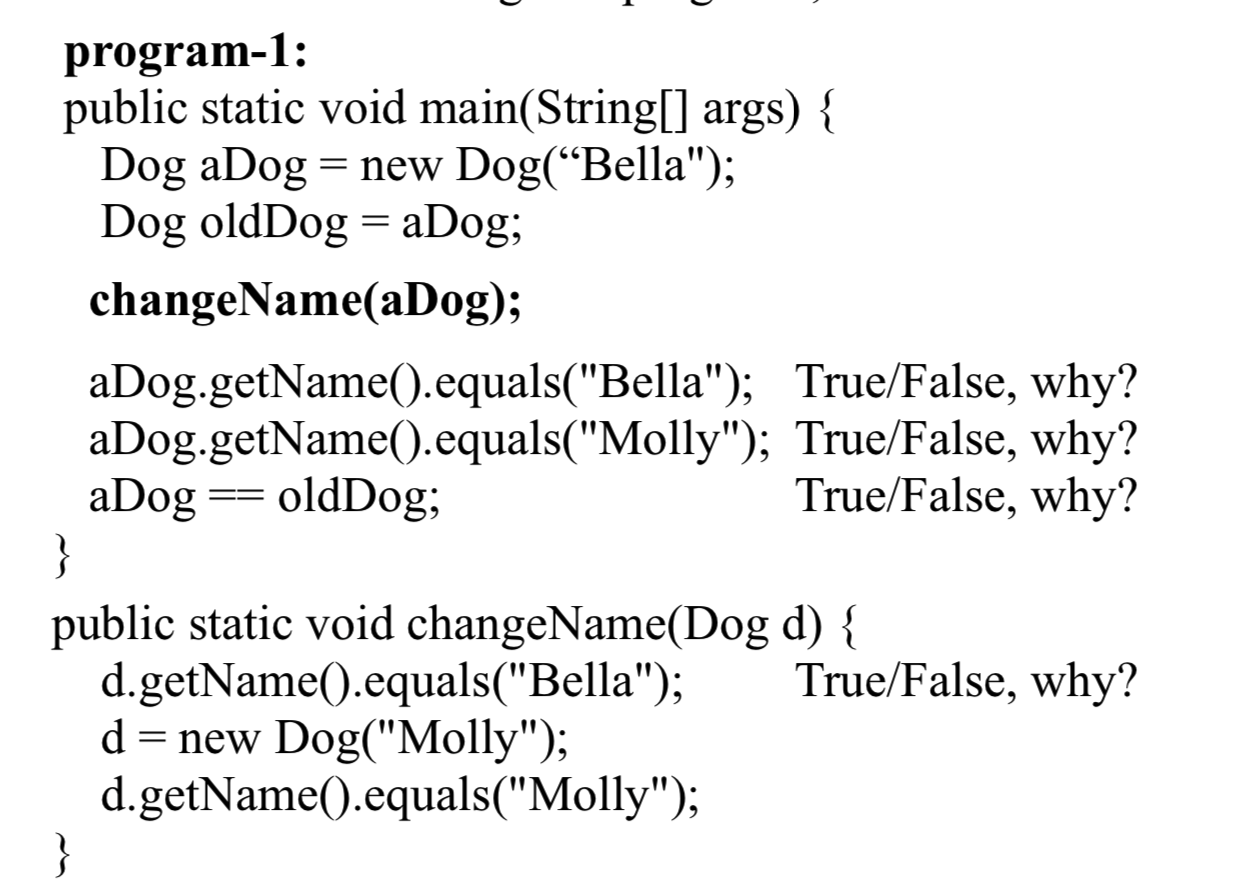
let result multiple 31 again, and puls the hashCode of member birth; then return result;

Override method compareTo(User o), return this.id – o.id;

means that if we have a collection of User, and we cannot define any order by user itself, so when we finish overriding method, we can set the default order by User’s id from small to high

in collection like arraylist;

8.

1.True: because aDog.Name is equal to String “Bella”

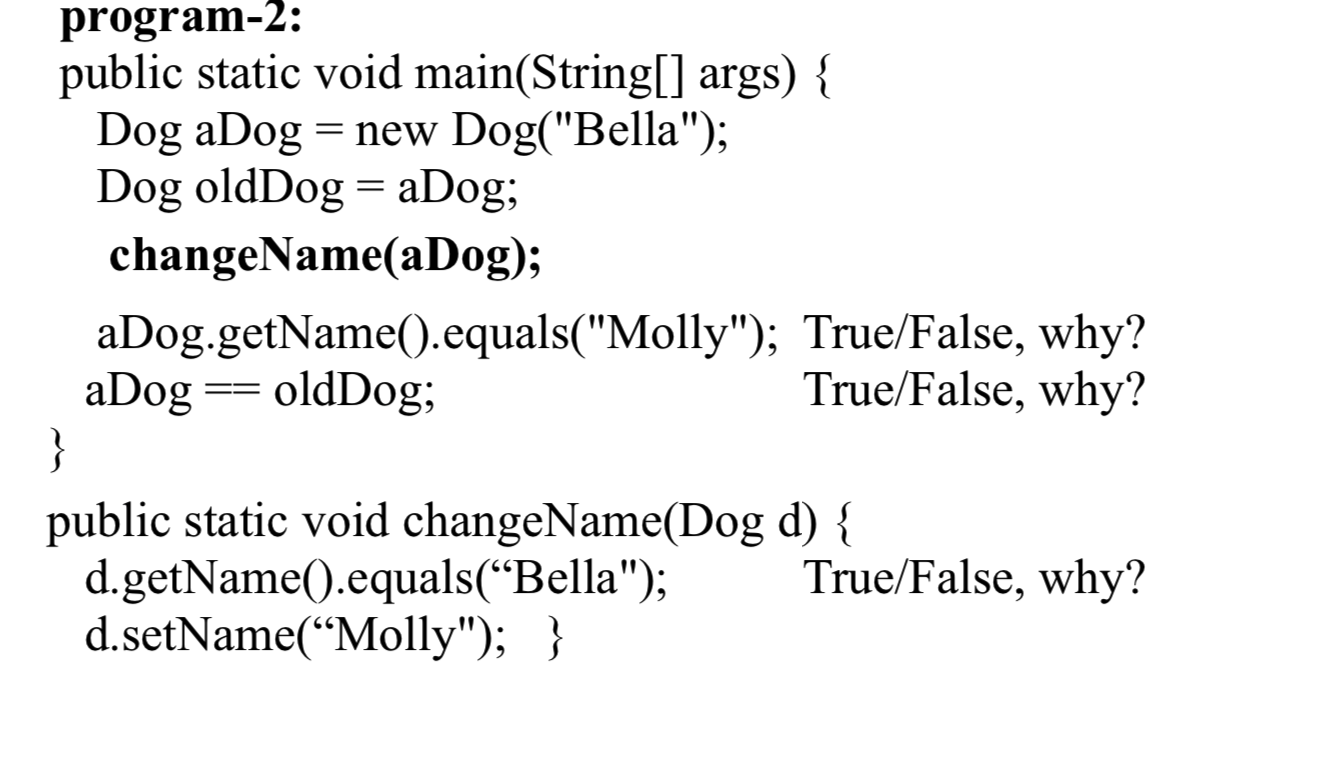


2.True: Because aDog has referred to a new address which memoried a new dog name “Molly

3..True because Only aDog’s value has been passed to Method ChangeName, so the address of aDog still keep the name”Bella”;

4.False: Same reason.

5.True: Because aDog & oldDog refer to one address in Computer memory.





1. True because adog.Name is equal to “Bella”
2. True because the memory of aDog’address has been changed. And after change the name of aDog is equal to “Molly”
3. True.Because two references refer to one address.