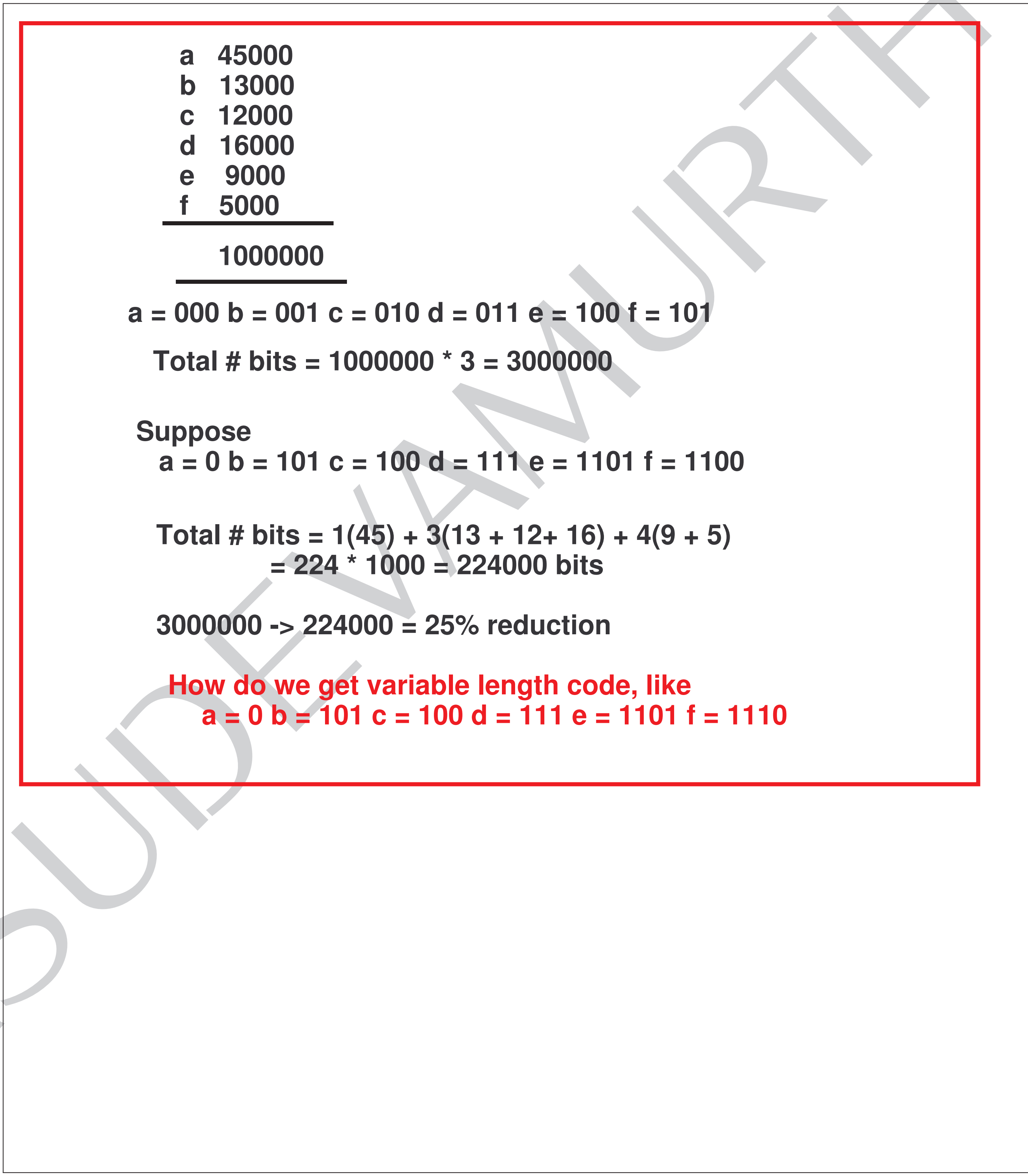
Chapter 16

Greedy Algorithms

16.1 Introduction

16.2 Hauffman coding



a	45000
b	13000
c	12000
d	16000
e	9000
f	5000
	1000000

a = 000 b = 001 c = 010 d = 011 e = 100 f = 101

Total # bits = 1000000 * 3 = 3000000

Suppose
a = 0 b = 101 c = 100 d = 111 e = 1101 f = 1100

Total # bits = 1(45) + 3(13 + 12+ 16) + 4(9 + 5)
= 224 * 1000 = 224000 bits

3000000 -> 224000 = 25% reduction

How do we get variable length code, like
a = 0 b = 101 c = 100 d = 111 e = 1101 f = 1110

Figure 16.1: Need for compression.

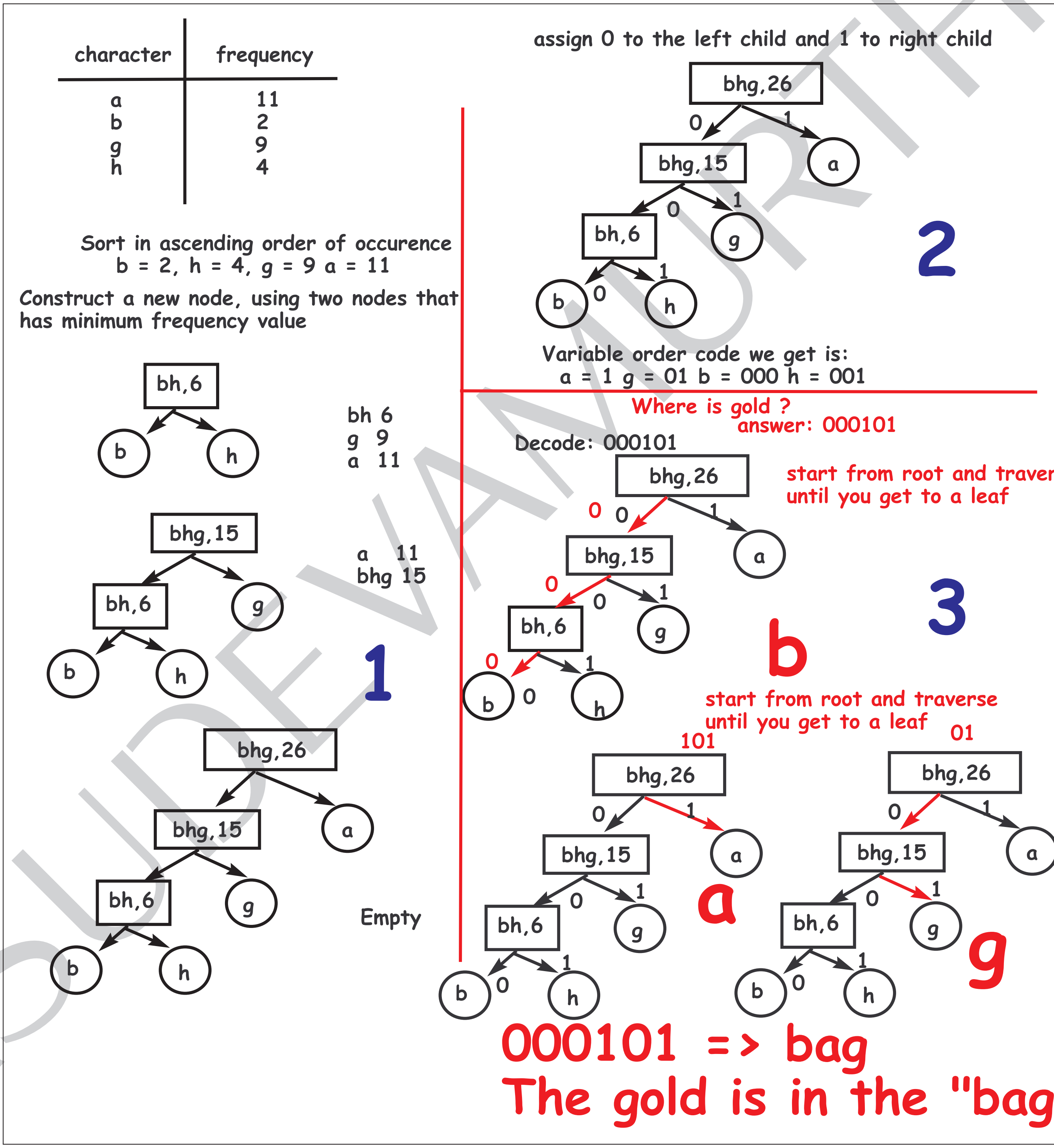


Figure 16.2: Compression using greedy algorithm

16.3 Dijkstra's algorithm

16.4 Scheduling problem

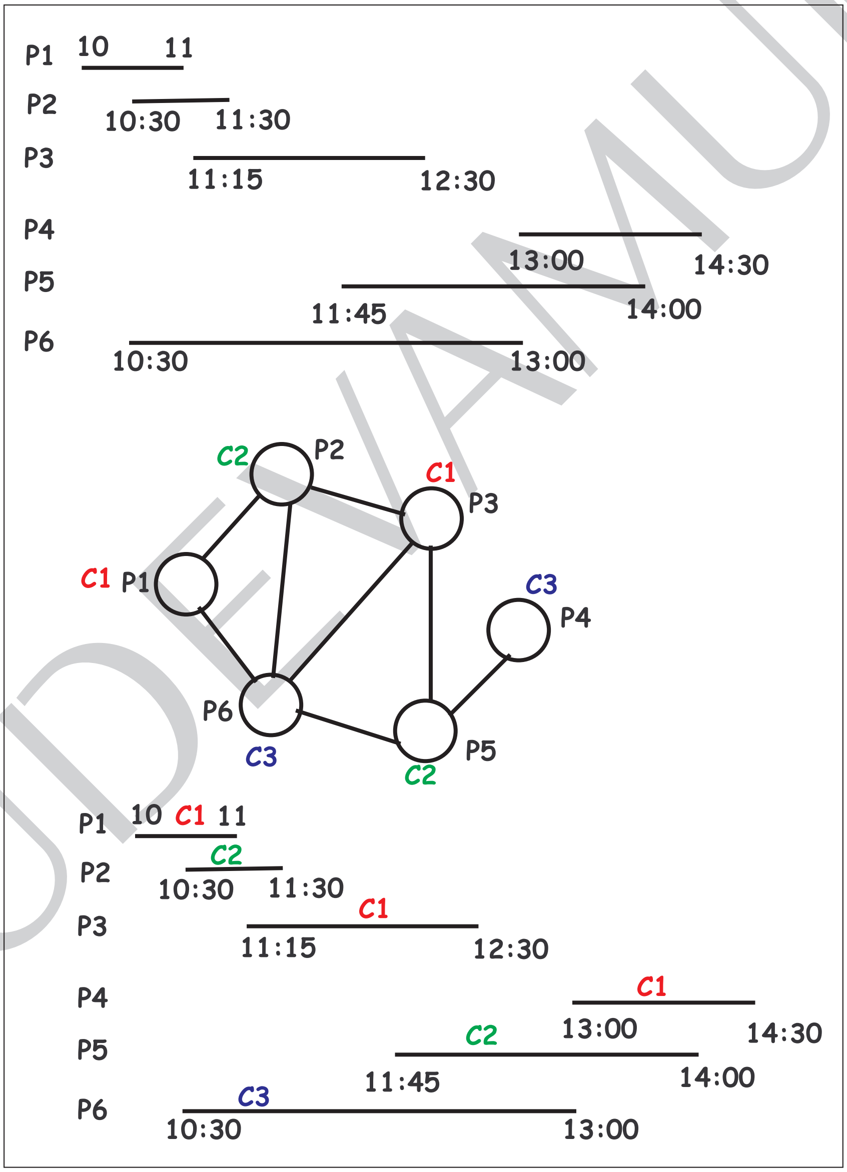
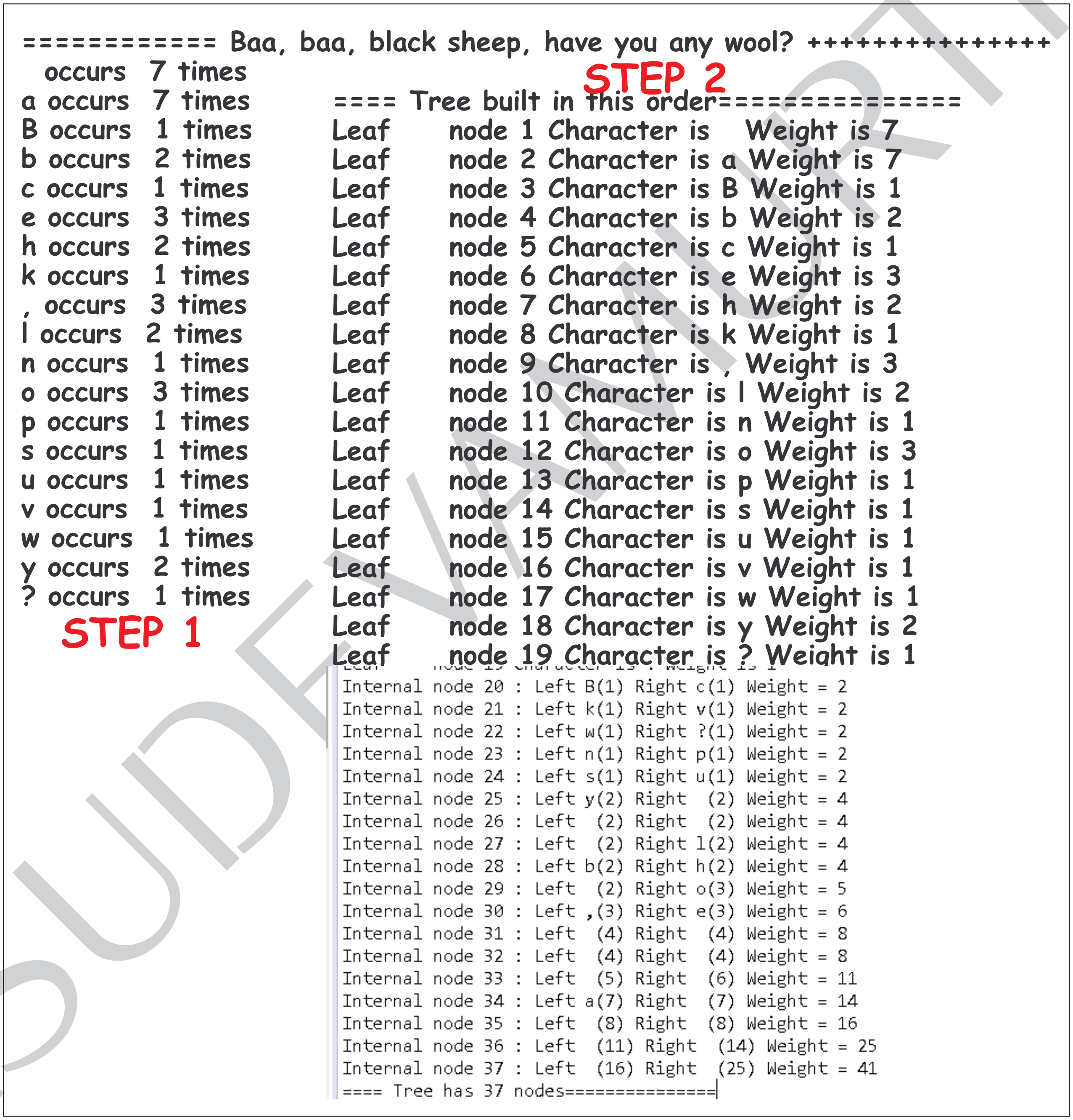


Figure 16.3: Minimum channels required to broadcast seven programs

16.5 Problem set

Problem 16.5.1. Write a class called **Hauffman** as explained in figures below.

```java
/**
 * File Name: HauffmanTest.java
 * Test Hauffman encode and decode algorithms
 *
 * @author Jagadeesh Vasudevamurthy
 * @year 2016
 */

public class HauffmanTest{
  private static final IntUtil u = new IntUtil();

  public static void test1(String s, boolean show, String dotfilename) {
    Hauffman h = new Hauffman(s,show,dotfilename);
    String d = h.decode();
    String f = h.encode();
    u.myassert(s.equals(f)) ;
    double sl = s.length() * 7 ;
    double dl = d.length();
    System.out.println("Original string cost = " + sl) ;
    System.out.println("Decoded  string cost = " + dl) ;
    double r = ((dl - sl)/sl) * 100 ;
    System.out.println("% reduction = " + (-r)) ;
  }

  public static void testbed() {
    boolean show = true ;
    test1("a",show,"C:\\work\\java\\fig\\1.dot");
    test1("aba",show,"C:\\work\\java\\fig\\2.dot");
    test1("aaabbgggghhhhaaagggaaaaa_+@#",show,"C:\\work\\java\\fig\\3.dot");
    test1("A quick brown fox jumps over the lazy dog",show,"C:\\work\\java\\fig\\4.dot");
    test1("Pack my box with five dozen liquor jugs",show,"C:\\work\\java\\fig\\5.dot");
    test1("Long years ago we made a tryst with destiny, and now the time comes when we shall
redeem our pledge, not wholly or in full measure, but very substantially.At the stroke of the
midnight hour, when the world sleeps, India will awake to life and freedom. A moment comes,
which comes but rarely in history, when we step out from the old to the new, when an age ends,
and when the soul of a nation, long suppressed, finds utterance.",show,"C:\\work\\java\\fig\
\6.dot");
    test1("Baa, baa, black sheep, have you any wool?",show,"C:\\work\\java\\fig\\7.dot") ;

    if (show) {
      System.out.println("===============  Done with Test1 ==================") ;
    }
  }

  public static void main(String[] args) {
    System.out.println("HauffmanTest.java");
    testbed() ;
    System.out.println(" All Hauffman Test passed. You are great. You should get an award");
  }

}
```

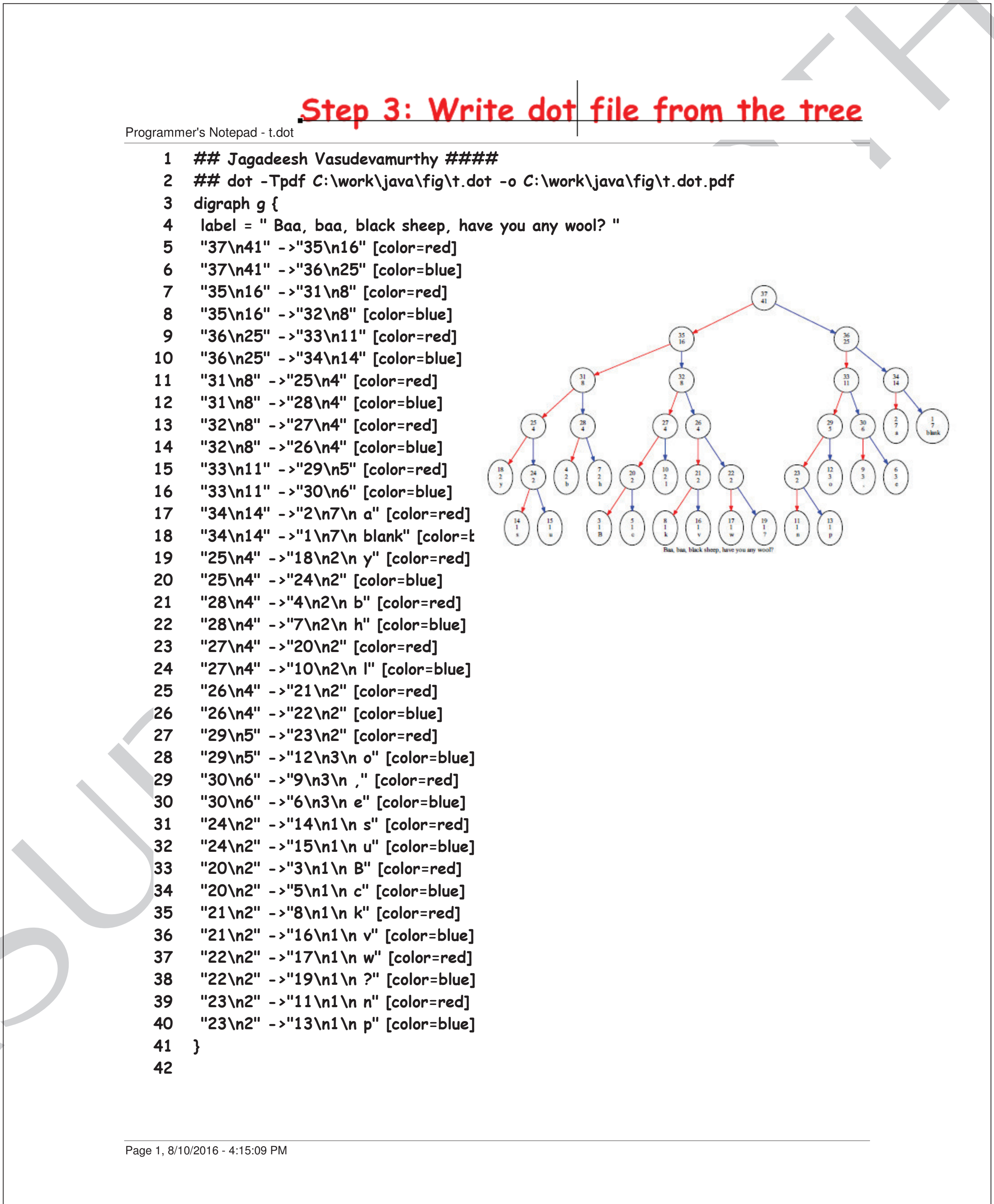


STEP 2

STEP 1

```
============ Baa, baa, black sheep, have you any wool? +++++++++++++++
  occurs  7 times
a occurs  7 times        ==== Tree built in this order===============
B occurs  1 times        Leaf    node 1 Character is   Weight is 7
b occurs  2 times        Leaf    node 2 Character is a Weight is 7
c occurs  1 times        Leaf    node 3 Character is B Weight is 1
e occurs  3 times        Leaf    node 4 Character is b Weight is 2
h occurs  2 times        Leaf    node 5 Character is c Weight is 1
k occurs  1 times        Leaf    node 6 Character is e Weight is 3
, occurs  3 times        Leaf    node 7 Character is h Weight is 2
l occurs  2 times        Leaf    node 8 Character is k Weight is 1
n occurs  1 times        Leaf    node 9 Character is , Weight is 3
o occurs  3 times        Leaf    node 10 Character is l Weight is 2
p occurs  1 times        Leaf    node 11 Character is n Weight is 1
s occurs  1 times        Leaf    node 12 Character is o Weight is 3
u occurs  1 times        Leaf    node 13 Character is p Weight is 1
v occurs  1 times        Leaf    node 14 Character is s Weight is 1
w occurs  1 times        Leaf    node 15 Character is u Weight is 1
y occurs  2 times        Leaf    node 16 Character is v Weight is 1
? occurs  1 times        Leaf    node 17 Character is w Weight is 1
                         Leaf    node 18 Character is y Weight is 2
                         Leaf    node 19 Character is ? Weight is 1
                         Internal node 20 : Left B(1) Right c(1) Weight = 2
                         Internal node 21 : Left k(1) Right v(1) Weight = 2
                         Internal node 22 : Left w(1) Right ?(1) Weight = 2
                         Internal node 23 : Left n(1) Right p(1) Weight = 2
                         Internal node 24 : Left s(1) Right u(1) Weight = 2
                         Internal node 25 : Left y(2) Right  (2) Weight = 4
                         Internal node 26 : Left  (2) Right  (2) Weight = 4
                         Internal node 27 : Left  (2) Right l(2) Weight = 4
                         Internal node 28 : Left b(2) Right h(2) Weight = 4
                         Internal node 29 : Left  (2) Right o(3) Weight = 5
                         Internal node 30 : Left ,(3) Right e(3) Weight = 6
                         Internal node 31 : Left  (4) Right  (4) Weight = 8
                         Internal node 32 : Left  (4) Right  (4) Weight = 8
                         Internal node 33 : Left  (5) Right  (6) Weight = 11
                         Internal node 34 : Left a(7) Right  (7) Weight = 14
                         Internal node 35 : Left  (8) Right  (8) Weight = 16
                         Internal node 36 : Left  (11) Right  (14) Weight = 25
                         Internal node 37 : Left  (16) Right  (25) Weight = 41
                         ==== Tree has 37 nodes==============
```

Figure 16.4: Step 1 and Step 2 of the Hauffman algorithm



Step 3: Write dot file from the tree

Programmer's Notepad - t.dot

```
## Jagadeesh Vasudevamurthy ####
## dot -Tpdf C:\work\java\fig\t.dot -o C:\work\java\fig\t.dot.pdf
digraph g {
 label = " Baa, baa, black sheep, have you any wool? "
 "37\n41" ->"35\n16" [color=red]
 "37\n41" ->"36\n25" [color=blue]
 "35\n16" ->"31\n8" [color=red]
 "35\n16" ->"32\n8" [color=blue]
 "36\n25" ->"33\n11" [color=red]
 "36\n25" ->"34\n14" [color=blue]
 "31\n8" ->"25\n4" [color=red]
 "31\n8" ->"28\n4" [color=blue]
 "32\n8" ->"27\n4" [color=red]
 "32\n8" ->"26\n4" [color=blue]
 "33\n11" ->"29\n5" [color=red]
 "33\n11" ->"30\n6" [color=blue]
 "34\n14" ->"2\n7\n a" [color=red]
 "34\n14" ->"1\n7\n blank" [color=b
 "25\n4" ->"18\n2\n y" [color=red]
 "25\n4" ->"24\n2" [color=blue]
 "28\n4" ->"4\n2\n b" [color=red]
 "28\n4" ->"7\n2\n h" [color=blue]
 "27\n4" ->"20\n2" [color=red]
 "27\n4" ->"10\n2\n l" [color=blue]
 "26\n4" ->"21\n2" [color=red]
 "26\n4" ->"22\n2" [color=blue]
 "29\n5" ->"23\n2" [color=red]
 "29\n5" ->"12\n3\n o" [color=blue]
 "30\n6" ->"9\n3\n ," [color=red]
 "30\n6" ->"6\n3\n e" [color=blue]
 "24\n2" ->"14\n1\n s" [color=red]
 "24\n2" ->"15\n1\n u" [color=blue]
 "20\n2" ->"3\n1\n B" [color=red]
 "20\n2" ->"5\n1\n c" [color=blue]
 "21\n2" ->"8\n1\n k" [color=red]
 "21\n2" ->"16\n1\n v" [color=blue]
 "22\n2" ->"17\n1\n w" [color=red]
 "22\n2" ->"19\n1\n ?" [color=blue]
 "23\n2" ->"11\n1\n n" [color=red]
 "23\n2" ->"13\n1\n p" [color=blue]
}
```

Page 1, 8/10/2016 - 4:15:09 PM

Figure 16.5: Step 3 of the Hauffman algorithm

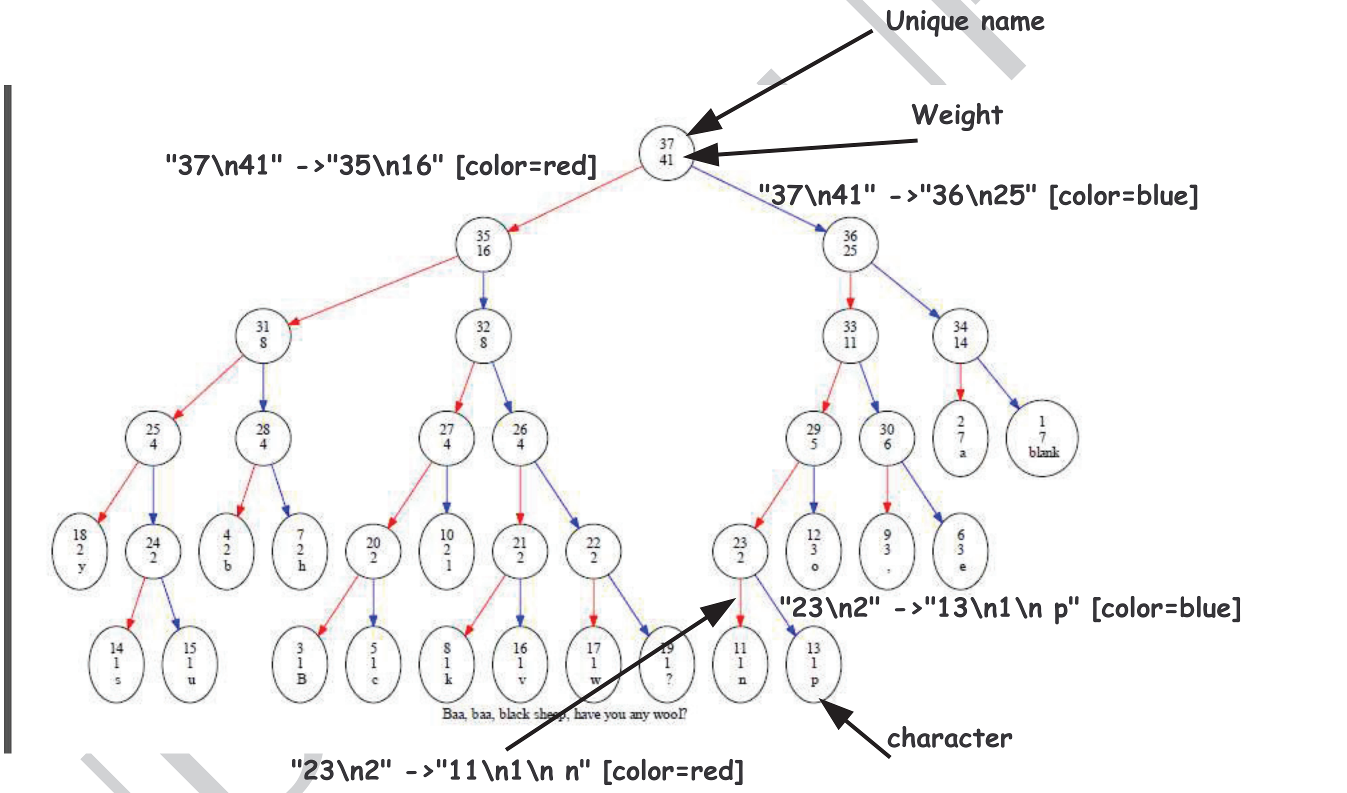
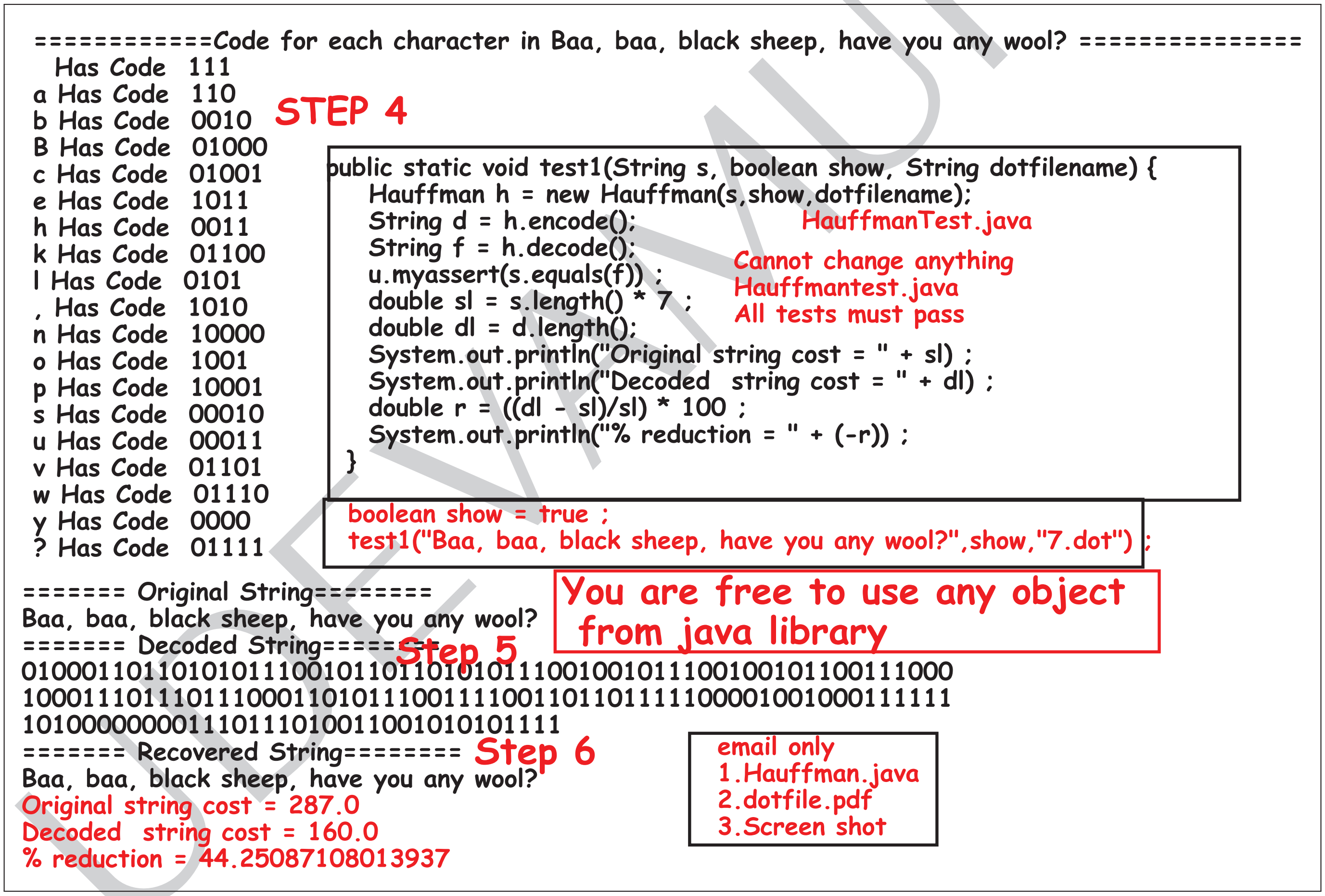

Unique name
Weight
"37\n41" ->"35\n16" [color=red]
"37\n41" ->"36\n25" [color=blue]
37
41
35
16
36
25
31
8
32
8
33
11
34
14
25
4
28
4
27
4
26
4
29
5
30
6
2
7
a
1
7
blank
18
2
y
24
2
4
2
b
7
2
h
20
2
10
2
1
21
2
22
2
23
2
12
3
o
9
3
,
6
3
e
"23\n2" ->"13\n1\n p" [color=blue]
14
1
s
15
1
u
3
1
B
5
1
c
8
1
k
16
1
v
17
1
w
19
1
?
11
1
n
13
1
p
Baa, baa, black sheep, have you any wool?
character
"23\n2" ->"11\n1\n n" [color=red]

Figure 16.6: Binary tree created by Hauffman algorithm



```
============Code for each character in Baa, baa, black sheep, have you any wool? ===============
   Has Code  111
 a Has Code  110
 b Has Code  0010
 B Has Code  01000
 c Has Code  01001
 e Has Code  1011
 h Has Code  0011
 k Has Code  01100
 l Has Code  0101
 , Has Code  1010
 n Has Code  10000
 o Has Code  1001
 p Has Code  10001
 s Has Code  00010
 u Has Code  00011
 v Has Code  01101
 w Has Code  01110
 y Has Code  0000
 ? Has Code  01111
```

STEP 4

```
public static void test1(String s, boolean show, String dotfilename) {
    Hauffman h = new Hauffman(s,show,dotfilename);
    String d = h.encode();
    String f = h.decode();
    u.myassert(s.equals(f)) ;
    double sl = s.length() * 7 ;
    double dl = d.length();
    System.out.println("Original string cost = " + sl) ;
    System.out.println("Decoded  string cost = " + dl) ;
    double r = ((dl - sl)/sl) * 100 ;
    System.out.println("% reduction = " + (-r)) ;
 }
```

HauffmanTest.java

Cannot change anything
Hauffmantest.java
All tests must pass

```
boolean show = true ;
test1("Baa, baa, black sheep, have you any wool?",show,"7.dot") ;
```

You are free to use any object
from java library

```
======= Original String========
Baa, baa, black sheep, have you any wool?
======= Decoded String========
01000110110101011100101101101010111001001011100100101100111000
10001110111011100011010111001111001101101111100001001000111111
101000000001110111010011001010101111
======= Recovered String========
Baa, baa, black sheep, have you any wool?
Original string cost = 287.0
Decoded  string cost = 160.0
% reduction = 44.25087108013937
```

Step 5

Step 6

email only
1.Hauffman.java
2.dotfile.pdf
3.Screen shot

Figure 16.7: Step 4 and Step 5 of the Hauffman algorithm