哈夫曼編碼解碼 C++實現

編程語言 · 發表 2017-05-14

錯誤 urn using 過程 簡單 cin n) struct ren

哈夫曼編碼是一個通過哈夫曼樹進行的一種編碼，一般情況下，以字符：‘0’與‘1’表示。編碼的實現過程很簡單，只要實現哈夫曼樹，通過遍歷哈夫曼樹，這裏我們從每一個葉子結點開

tcp和udp之間區別的

前端基本知識：JS的原始鏈的理解

始向上遍歷，如果該結點為父節點的左孩子，則在字符串後面追加“0”，如果為其右孩子，則在字符串後追加“1”。結束條件為沒有父節點。然後將字符串倒過來存入結點中。

C++實現代碼如下：

```
  1 #include<iostream>
  2 #include<string>
  3 using namespace std;
  4
  5 struct Node
  6 {
  7     double weight;
  8     string ch;
  9     string code;
 10     int lchild, rchild, parent;
 11 };
 12
 13 void Select(Node huffTree[], int *a, int *b, int n)//找權值最小的兩個a和b
 14 {
 15     int i;
 16     double weight = 0; //找最小的數
 17     for (i = 0; i <n; i++)
 18     {
 19         if
```

python基礎一 ------Python 的編碼

使用openstack的虛擬機模版註意事項

ListView優化總結（二）--Android

wncryt病毒大爆發

Sublime Text3學習參考集

continue 的用法

【java設計模式】代理模式

AJAX+PHP實現三級聯動

```
(huffTree[i].parent
!= -1)      //判斷節
點是否已經選過
 20
continue;
 21          else
 22          {
 23              if
(weight == 0)
 24              {
 25
weight =
huffTree[i].weight;
 26
*a = i;
 27              }
 28
else
 29              {
 30
if
(huffTree[i].weight
< weight)
 31
{
 32
weight =
huffTree[i].weight;
 33
*a = i;
 34
}
 35              }
 36          }
 37      }
 38      weight = 0;
//找第二小的數
 39      for (i = 0;
i < n; i++)
 40      {
 41          if
(huffTree[i].parent
!= -1 || (i ==
*a))//排除已選過的數
 42
continue;
 43          else
 44          {
 45              if
```

```
(weight == 0)
 46              {
 47
weight =
huffTree[i].weight;
 48
*b = i;
 49              }
 50
else
 51              {
 52
if
(huffTree[i].weight
< weight)
 53
{
 54
weight =
huffTree[i].weight;
 55
*b = i;
 56
}
 57              }
 58          }
 59      }
 60      int temp;
 61      if
(huffTree[*a].lchil
d <
huffTree[*b].lchild
)  //小的數放左邊
 62      {
 63          temp =
*a;
 64          *a =
*b;
 65          *b =
temp;
 66      }
 67 }
 68
 69 void
Huff_Tree(Node
huffTree[], int
w[], string ch[],
int n)
 70 {
```

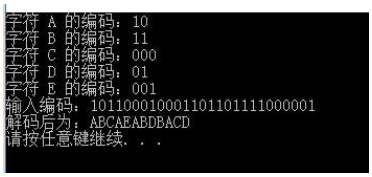
```
 71      for (int i
= 0; i < 2 * n - 1;
i++) //初始過程
 72      {
 73
huffTree[i].parent
= -1;
 74
huffTree[i].lchild
= -1;
 75
huffTree[i].rchild
= -1;
 76
huffTree[i].code =
"";
 77      }
 78      for (int i
= 0; i < n; i++)
 79      {
 80
huffTree[i].weight
= w[i];
 81
huffTree[i].ch =
ch[i];
 82      }
 83      for (int k
= n; k < 2 * n - 1;
k++)
 84      {
 85          int i1
= 0;
 86          int i2
= 0;
 87
Select(huffTree,
&i1, &i2, k); //將
i1，i2節點合成節點k
 88
huffTree[i1].parent
= k;
 89
huffTree[i2].parent
= k;
 90
huffTree[k].weight
=
huffTree[i1].weight
```

```
+
huffTree[i2].weight
;
 91
huffTree[k].lchild
= i1;
 92
huffTree[k].rchild
= i2;
 93      }
 94 }
 95
 96 void
Huff_Code(Node
huffTree[], int n)
 97 {
 98      int i, j,
k;
 99      string s =
"";
100      for (i = 0;
i < n; i++)
101      {
102          s = "";
103          j = i;
104          while
(huffTree[j].parent
!= -1) //從葉子往上找
到根節點
105          {
106              k =
huffTree[j].parent;
107              if
(j ==
huffTree[k].lchild)
//如果是根的左孩子，則
記為0
108              {
109
s = s + "0";
110              }
111
else
112              {
113
s = s + "1";
114              }
115              j =
huffTree[j].parent;
```

```
116         }
117         cout << "字符 " << huffTree[i].ch << " 的編碼：";
118         for (int l = s.size() - 1; l >= 0; l--)
119         {
120 
cout << s[l];
121 
huffTree[i].code += s[l]; //保存編碼
122         }
123         cout << endl;
124     }
125 }
126 
127 string Huff_Decode(Node huffTree[], int n,string s)
128 {
129     cout << "解碼後為：";
130     string temp = "",str="";//保存解碼後的字符串
131     for (int i = 0; i < s.size(); i++)
132     {
133         temp = temp + s[i];
134         for (int j = 0; j < n; j++)
135         {
136             if (temp == huffTree[j].code)
137             {
138 
str=str+ huffTree[j].ch;
139 
temp = "";
```

```
140
break;
141             }
142
else if (i ==
s.size()-1&&j==n-
1&&temp!="")//全部遍
歷後沒有
143             {
144
str= "解碼錯誤！";
145             }
146         }
147     }
148     return str;
149 }
150
151 int main()
152 {
153     //編碼過程
154     const int
n=5;
155     Node
huffTree[2 * n];
156     string
str[] = { "A", "B",
"C", "D", "E"};
157     int w[] = {
30, 30, 5, 20, 15
};
158
Huff_Tree(huffTree,
w, str, n);
159
Huff_Code(huffTree,
n);
160     //解碼過程
161     string s;
162     cout << "輸
入編碼：";
163     cin >> s;
164     cout <<
Huff_Decode(huffTre
e, n, s)<< endl;;
165
system("pause");
166     return 0;
167 }
```

運行結果如下：



哈夫曼編碼解碼 C++實現

標籤：

您可能也會喜歡...

轉載：哈夫曼樹的構造和哈夫曼編碼（C++代碼實現）

哈夫曼編碼--貪心策略

哈夫曼編碼解碼 C++實現

【視頻編解碼·學習筆記】7. 熵編碼 算法：基礎知識 & 哈夫曼編碼

樹-哈夫曼編碼

SDUT 3345 數據結

構實驗之二叉樹六：哈夫曼編碼	bzoj 4198 [Noi 2015] 荷馬史詩 —— 哈夫曼編碼(k叉哈夫曼樹)
HDU 1053 Entropy（哈夫曼編碼 貪心+優先隊列)	5.2哈夫曼樹——哈夫曼樹與哈夫曼編碼
【BZOJ 4198】[Noi2015]荷馬史詩 哈夫曼編碼	哈夫曼編碼大全
	哈夫曼編碼
	哈夫曼編碼 (Huffman coding) 的那些事,(編碼技術介紹和程序實現)
霍夫曼編碼	通過編碼來實現軟件界面

首頁

站长统计