Олександр Рудик

Пошук бінарного коду (дерева) з мінімальною вагою

(Інформатика та інформаційні технології в навчальних закладах, 2008, № 6, с. 98–102)

Тут і далі A — множина всіх символів, за допомогою яких зберігають інформацію про речення, записане певною мовою (латиниця, кирилиця, множина ієрогліфів, {0,1} тощо).

Означення 1. *Запровадимо такі поняття й позначення.*

1. *Кодом однозначного декодування називають множину C скінчених послідовностей символів, при якій довільний рядок тексту можна взаємно однозначно задати конкатенацією (послідовним записом) елементів C.*
2. *Код називають бінарним (двійковим), якщо множина символів, за допомогою яких записано код, має два елементи.*
3. *Код називають префіксним, якщо жоден елемент коду не є початком (підпослідовністю без вилучення, починаючи з першого елемента послідовності) іншого елемента коду.*
4. *Кодом шляху в бінарному дереві називають бінарний код шляху від кореня (вершини, що не є кінцем дуги) до листка (вершини, що не є початком дуги графа).*
5. *Нехай кожна літера a_j абетки A має додатну вагу f_j, а відповідний елемент двійкового коду має довжину l_j. Тоді вагою коду називають суму:*

$$f_1 l_1 + f_2 l_2 + f_3 l_3 + \cdots .$$

Зауваження 1.

1. *Довільний елемент бінарного коду можна подати послідовністю нулів та одиниць.*
2. *Код шляху в бінарному дереві має доволі прозору геометричну інтерпретацію: один символ вказує на рух праворуч, інший — на рух ліворуч при переході до нащадка при зображенні бінарного дерева на площині.*
3. *Традиційно за вагу літери вибирають відносну частоту її появи.*
4. *Існує взаємно однозначна відповідність між двійковими префіксними кодами і бінарними деревами. При приписуванні листкам дерев певної ваги вагою дерева називають вагу відповідного коду (не плутати з випадком приписування ваги ребру).*

Подамо алгоритм побудови бінарного дерева (префіксного коду) з мінімальною вагою за відомими вагами листків (літер).

Алгоритм Хаффмана (Haffman's algorithm)

1. Впорядковуємо ваги (відносні частоти) у порядку неспадання.
2. Вилучаємо з масиву ваг (частот) найменші елементи f_j та f_k, замість них долучаємо ($f_j + f_k$).
3. Створюємо бінарне дерево з одним коренем і його двома безпосередніми нащадками (символами або раніше побудованими бінарними деревами), що відповідають вилученим вагам (частотам) f_j та f_k :
   - дузі від кореня до одного з нащадків ставимо у відповідність 0;
   - дузі від кореня до іншого нащадка ставимо у відповідність 1;
   - долученій вазі (частоті) ($f_j + f_k$) ставимо у відповідність корінь побудованого бінарного дерева.
4. Кроки 1–3 здійснюємо доти, поки у масиві невилучених ваг (частот) не залишиться один елемент.
5. Формуємо код Хаффмана, утворюючи послідовність з нулів і одиниць, поставлених у відповідність дугам маршруту (шляху) з кореня останнього побудованого дерева до заданого елемента (код шляху).

Зауваження 2. *Для однозначності алгоритму Хаффмана на кроці 3 потрібно задати правило відповідності різним дугам нуля чи одиниці. Наприклад, з використанням алфавітного порядку для символу чи послідовності символів, що є листками побудованих дерев.*

Лема 1. *Для довільної скінченої множини листків (символів) і довільного розподілу ваг (частот) серед цих символів існує бінарне дерево мінімальної ваги з цими листками (символами).*

Доведення. Існує скінчена кількість бінарних дерев з даними листками. Всі можливі ваги таких дерев утворюють скінчену множину. Скінчена множина дійсних чисел має найбільший та найменший елементи.

Лема 2. *Для довільної скінченої множини листків і довільного розподілу ваг (частот) серед цих листків у бінарному дереві з найменшою вагою на максимальному рівні листки присутні парами.*

Доведення (від супротивного). Припустимо, що на максимальному рівні бінарного дерева з найменшою вагою є лише один листок. Утворимо нове дерево, вилучивши дугу, спрямовану в цей листок і поставивши його вагу (частоту) у відповідність початку цієї дуги. Отримаємо дерево з меншою вагою, що суперечить припущенню про мінімальну вагу початкового дерева.

Лема 3. *Для довільної скінченої множини листків і довільного розподілу ваг (частот) серед цих листків у бінарному дереві з найменшою вагою два символи з мінімальними частотами розташовані на максимальному рівні, тобто з максимальною довжиною коду шляху.*

Доведення (від супротивного). Припустимо, що у бінарному дереві з найменшою вагою є листки з вагами (частотами) f_j та f_k, що відповідно мають довжини l_j та l_k кодів шляху, при яких $f_j > f_k$, $l_j > l_k$. Вклад цих листків у вагу дерева складає ($f_j \; l_j + f_k \; l_k$). Після того, як ми поміняємо їх місцями, їхній вклад складе ($f_j \; l_k + f_k \; l_j$). Приріст складе:

$$(f_j \; l_k + f_k \; l_j) - (f_j \; l_j + f_k \; l_k) = - (f_j - f_k) \; (\; l_j - l_k \;) < 0,$$

що суперечить твердженню про мінімальність початкового дерева. Отже, хибним є припущення про існування листка з мінімальною вагою не на максимальному рівні дерева з найменшою вагою.

Лема 4. *Для довільної скінченої множини листків і довільного розподілу ваг (частот) серед цих листків існує бінарне дерево з найменшою вагою, у якому два листки з найменшими вагами (частотами) мають одного безпосереднього предка.*

Доведення. У бінарному дереві з найменшою вагою:

- згідно з лемою 2 на максимальному рівні розташовано щонайменше два листки;
- згідно з лемою 3 два листки з найменшою вагою (частотою) розташовані на максимальному рівні.

Припустимо, що два листки з найменшою вагою (частотою) не мають спільного безпосереднього предка. Тоді заміною місць листків з максимального рівня, що не змінює ваги дерева, отримаємо потрібне дерево.

Теорема 1. *Для довільної скінченої множини листків і довільного розподілу ваг (частот) серед цих листків дерево, побудоване згідно з алгоритмом Хаффмана, є бінарним деревом з найменшою вагою.*

Доведення (методом математичної індукції за кількістю листків n).

1. **База індукції.** Очевидно, що твердження теореми справджується для $n = 2$.
2. **Припущення індукції.** Припустимо, що твердження теореми справджується для натурального n.
3. **Крок індукції.** Нехай T_{n+1} мінімальне дерево з $(n + 1)$ листком, у якому два листки з найменшими вагами (частотами) f_j та f_k мають одного безпосереднього предка. Утворимо дерево T_n з T_{n+1}, вилучивши вказані два листки, а їхньому безпосередньому предку надавши вагу (частоту) ($f_j + f_k$). Дерево T_n має n листків. Для цих n листків і заданого

розподілу ваг (частот) згідно з припущенням індукції існує дерево з мінімальною вагою, побудоване згідно з алгоритмом Хаффмана. Позначимо його через T_n'. Утворимо дерево T'_{n+1} з T_n', здійснивши перетворення, оберне до застосованого при побудові T_{n+1}. Інакше кажучи, замінимо листок з вагою (частотою) $(f_j + f_k)$ на вершину, що має два прямі нащадки з вагами (частотами) f_j та f_k. Зауважимо: T'_{n+1} побудоване згідно з алгоритмом Хаффмана для $(n + 1)$-го листка. З такої системи висловлювань:

- при переході від дерева T_n до дерева T_{n+1} вага дерева зростає на $(f_j + f_k)$;
- при переході від дерева T_n' до дерева T'_{n+1} вага дерева зростає на (f_j+f_k);
- вага T_n не менша за вагу дерева T'_n, що є найменшою з усіх можливих для вказаних n листків;
- вага T'_{n+1} не менша за вагу дерева T_{n+1}, що є найменшою з усіх можливих для вказаних $(n + 1)$-го листка

випливає, що ваги дерев T'_{n+1} і T_{n+1} збігаються. Отже, вага дерева T'_{n+1}, побудованого згідно з алгоритмом Хаффмана, є найменшою з усіх можливих для вказаних $(n + 1)$-го листка.

Згідно з п'ятою аксіомою Пеано (чи принципом математичної індукції) робимо висновок про істинність твердження теореми для довільної кількості листків.

На завершення викладу подамо приклад програми мовою Turbo Pascal 7.0, що зчитує з вхідного файлу haffman.dat ваги (частоти) листків і, не перевіряючи коректність даних, записує у вихідний файл haffman.res коди Хаффмана для зчитаних ваг (частот) у тому порядку, як ваги (частоти) було записано у вхідному файлі. Після побудови відповідного бінарного дерева на етапі визначення кодів листків подана програма реалізує *алгоритм обходу листків бінарного дерева*, у якому кожна наступна серед нерозглянутих вершин має найдовшу послідовність нулів на початку коду шляху (див. текст програми з коментарями).

```
{$N+}
const nd=1000; {Максимальна кількість листків}
      ns=20;   {Максимальна довжина коду}
type pointer=^vertex;
     vertex = record i: word;   {Тип вершини}
                 l,r,p: pointer end;
     data= array[0..nd] of record
            i: word;    {Номер листка}
            f: extended;{Вага (частота)}
            p: pointer  {Вказівник} end;
    binar= array[1..nd] of string[ns];
var v,vl,vr: pointer;
    a: ^data;   s: string;   o: text;
    b: ^binar;       j,j1,k,na: word;

PROCEDURE SWAP(j,k: word);              BEGIN
```

```
               a^[0]:=a^[j];
                      a^[j]:=a^[k];
                             a^[k]:=a^[0] END;

PROCEDURE QUICK (left,right: word);
                var l,r: word;          BEGIN
l:=left+random(right-left+1);
swap(l,left);   l:=left;   r:=right;
REPEAT
while(a^[r].f>=a^[0].f) and (l < r) do dec(r);
a^[l]:=a^[r];  a^[r]:=a^[0];
while(a^[l].f<=a^[0].f) and (l < r) do inc(l);
a^[r]:=a^[l];  a^[l]:=a^[0];
UNTIL l=r;     a^[l]:=a^[0];
if left < l-1  then quick(left,l-1);
if l+1 < right then quick(l+1,right)      END;

                                        BEGIN
assign(o,'HAFFMAN.DAT');   {Зчитування даних}
reset (o);   new(a);       na:=0;
REPEAT inc(na);  a^[na].i:=na;
          read(o,a^[na].f);
                a^[na].p:=nil
 UNTIL seekeoln(o);   close(o);

randomize;  quick(1,na);  {Впорядкування ваг}

                  {Побудова бінарного дерева}
for j:=2 to na do                       BEGIN
       j1:=j-1;    new(v);     v^.p:=nil;

                              {листок + листок}
if (a^[j1].i>0) and (a^[j].i>0) then begin
  new(vl);     vl^.i:=a^[j1].i;   v^.l:=vl;
  new(vr);     vr^.i:=a^[j].i;    v^.r:=vr;
  vl^.l:=nil;  vl^.r:=nil;  vl^.p:=v;
  vr^.l:=nil;  vr^.r:=nil;  vr^.p:=v end else

                              {листок + дерево}
if (a^[j1].i>0) and (a^[j].i=0) then begin
  new(vl);   vl^.i:=a^[j1].i;  vl^.l:=nil;
  v^.l:=vl;   v^.r:=a^[j].p;   vl^.r:=nil;
  vl^.p:=v;   a^[j].p^.p:=v          end else

                              {дерево + листок}
if (a^[j1].i=0) and (a^[j].i>0) then begin
  new(vr); a^[j1].p^.p:=v;  vr^.i:=a^[j].i;
  v^.l:=a^[j1].p;  vr^.l:=nil; vr^.p:=v;
  v^.r:=vr;        vr^.r:=nil        end else

                              {дерево + дерево}
if (a^[j1].i=0) and (a^[j].i=0) then begin
  a^[j].p^.p:=v;     v^.r:=a^[j].p;
```

```
 a^[j1].p^.p:=v;     v^.l:=a^[j1].p  end;

a^[j].i:=0;
a^[j].f:=a^[j].f+a^[j1].f;
a^[j].p:=v;                k:=j;
if k < na then repeat if a^[k].f>a^[k+1].f
                      then SWAP(k,k+1);
                      inc(k)
               until  na=k               END;
dispose(a);
new(b);            {Обчислення коду Хаффмана}
s:='';        {за створеним бінарним деревом}
REPEAT
              {Максимальний рух ліворуч вниз}
  while v^.l<>nil  do  begin
     v:=v^.l;   s:=s+'0' end;

  if    v^.r= nil then begin {Листок ліворуч}
    b^[v^.i]:=s;
    v:=v^.p;
    delete(s,length(s),1);
    dispose(v^.l);
    v^.l:=nil           end
  else                                  begin
{Рух праворуч вниз до першого повороту ліворуч
                       або до листка праворуч}
    repeat v:=v^.r;  s:=s+'1'
    until  (v^.l<>nil) or
           (v^.l= nil) and (v^.r=nil);

          {Якщо досягнуто листок праворуч...}
    if (v^.l=nil) and (v^.r=nil)   then begin
      b^[v^.i]:=s;
      repeat v:=v^.p;
        case s[length(s)] of '0':       begin
                dispose(v^.l);  v^.l:=nil end;
                             '1':       begin
                dispose(v^.r);  v^.r:=nil end
        end;
        delete(s,length(s),1);
      until (v^.p =nil) or
            (v^.l<>nil) or (v^.r<>nil)end end
UNTIL       (v^.l =nil)and (v^.r =nil);

      {Запис відповіді та звільнення пам'яті}
assign(o,'HAFFMAN.RES');           rewrite(o);
for j:=1 to na do writeln(o,b^[j]);  close(o);
dispose(v);  dispose(b)                  END.
```