Algorytmy wyszukujące - wyszukiwanie interpolacyjne

Algorytm

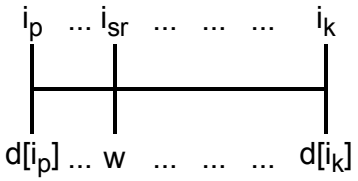
Wyszukiwanie interpolacyjne jest ulepszeniem przedstawionego w poprzednim rozdziale algorytmu wyszukiwania binarnego. Nadaje się ono dla zbiorów uporządkowanych o liniowym rozkładzie elementów. W algorytmie wyszukiwania binarnego partycję zbioru dzieliliśmy zawsze w połowie i tam poszukiwaliśmy danego elementu. Tymczasem możemy skorzystać z własności zbiorów uporządkowanych liniowo i wyliczyć prawdopodobną pozycję elementu wg wzoru:

$d[\,]$ - przeszukiwany zbiór
w - poszukiwana wartość
i_p - indeks pierwszego elementu partycji
i_k - indeks ostatniego elementu partycji
i_{sr} - wyznaczona, przypuszczalna pozycja

$$i_{sr} = i_p + \left[\frac{(w - d[i_p]) \times (i_k - i_p)}{d[i_k] - d[i_p]} \right]$$

We wzorze stosujemy dzielenie całkowitoliczbowe (pozycja nie może przecież być ułamkowa !).

Powyższy wzór wyprowadzamy z prostej proporcji. Jeśli zbiór posiada liniowy rozkład elementów, to odległość wartości poszukiwanej w od $d[i_p]$ jest w przybliżeniu proporcjonalna do liczby elementów pomiędzy i_p a i_{sr}:



Skoro tak, to zachodzi przybliżona proporcja:

$$\frac{i_{sr} - i_p}{i_k - i_p} \approx \frac{w - d[i_p]}{d[i_k] - d[i_p]}$$, mnożymy obustronnie przez $i_k - i_p$

$$i_{sr} - i_p \approx \frac{(w - d[i_p]) \times (i_k - i_p)}{d[i_k] - d[i_p]}$$, dodajemy obustronnie i_p

$$i_{sr} = i_p + \left[\frac{(w - d[i_p]) \times (i_k - i_p)}{d[i_k] - d[i_p]} \right]$$, zaokrąglamy do wartości całkowitej i otrzymujemy wzór końcowy

Aby zrozumieć zaletę tego sposobu wyznaczania pozycji, przyjrzyjmy się poniższemu przykładowi, gdzie wyliczyliśmy pozycję wg wzoru z poprzedniego rozdziału oraz wg wzoru podanego powyżej. Poszukiwany element zaznaczono kolorem czerwonym:

nr pozycji	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
elementy	10	11	13	13	**15**	16	18	19	20	22	22	23	25	27	27	28	29	30	32
binarnie i_{sr} =										**X**									
interpolacyjnie i_{sr} =					**X**														

Binarne wyznaczenie pozycji:

$$i_{sr} = \left[\frac{i_p + i_k}{2} \right] = \left[\frac{1 + 19}{2} \right] = \left[\frac{20}{2} \right] = 10$$

Interpolacyjne wyznaczenie pozycji:

$$i_{sr} = i_p + \left[\frac{(w - d[i_p]) \times (i_k - i_p)}{d[i_k] - d[i_p]} \right] = 1 + \left[\frac{(15 - 10) \times (19 - 1)}{32 - 10} \right] = 1 + \left[\frac{5 \times 18}{22} \right] = 1 + \left[\frac{90}{22} \right] = 1 + 4 = 5$$

Metoda interpolacyjna wyznacza pozycję zwykle dużo bliżej poszukiwanego elementu niż metoda binarna (w przykładzie trafiamy za pierwszym razem).
Zauważ, iż w metodzie binarnej nie korzystamy zupełnie z wartości elementów na krańcach dzielonej partycji, czyli działamy niejako na ślepo. Natomiast metoda interpolacyjna wyznacza położenie w zależności od wartości elementów zbioru oraz wartości poszukiwanego elementu. Działa zatem celowo dostosowując się do zastanych w zbiorze warunków. Stąd jej dużo większa efektywność.
Po wyznaczeniu położenia i_{sr} pozostała część algorytmu jest bardzo podobna do wyszukiwania binarnego.
Wyszukiwanie interpolacyjne posiada klasę czasowej złożoności obliczeniowej równą O(log log n), zatem wyszukuje znacząco szybciej w zbiorach o liniowym rozkładzie elementów niż wyszukiwanie binarne o klasie O(log n).

Specyfikacja problemu

Dane wejściowe

n - ilość elementów w przeszukiwanym zbiorze. $n \in \mathbf{N}$
d[ ] - posortowany rosnąco zbiór danych. Indeksy elementów rozpoczynają się od 1. Dodatkowo żądamy, aby zbiór posiadał liniowy rozkład elementów.
w - wartość poszukiwanego elementu. Typ elementu taki sam jak typ elementów zbioru.

Dane wyjściowe

p - pozycja elementu w zbiorze, $p \in \mathbf{C}$, $1 \leq p \leq n$ - element znaleziony, $p = 0$ - element nie znaleziony

Zmienne pomocnicze

i_p - zawiera indeks pierwszego elementu w partycji. $i_p \in \mathbf{C}$

i_k - zawiera indeks ostatniego elementu w partycji, $i_k \in \mathbf{C}$

i_{sr} - zawiera indeks środkowego elementu, $i_{sr} \in \mathbf{C}$

Lista kroków

krok 1: $i_p \leftarrow 1$; $i_k \leftarrow n$; $p \leftarrow 0$

krok 2: Dopóki $d[i_p] \leq w \leq d[i_k]$: wykonuj kroki 3...5

krok 3: $i_{sr} = i_p + \left[\frac{(w - d[i_p]) \times (i_k - i_p)}{d[i_k] - d[i_p]} \right]$

krok 4: Jeśli $w = d[i_{sr}]$, to $p \leftarrow i_{sr}$ i zakończ algorytm

krok 5: Jeśli $w > d[i_{sr}]$, to $i_p \leftarrow i_{sr} + 1$. Inaczej $i_k \leftarrow i_{sr} - 1$

krok 6: Zakończ algorytm

Schemat blokowy

Algorytm wyszukiwania interpolacyjnego jest bardzo podobny do algorytmu wyszukiwania binarnego. Różnice zaznaczyliśmy szarym kolorem symboli blokowych.

Inny jest warunek kontynuacji pętli wyszukującej - wykonuje się ona dotąd, dopóki poszukiwana wartość wpada w zakres partycji od $d[i_p]$ do $d[i_k]$.

Druga różnica występuje przy wyznaczaniu pozycji i_{sr}. W wyszukiwaniu binarnym pozycja ta znajdowała się w środku partycji zbioru. W wyszukiwaniu interpolacyjnym staramy się ustalić położenie poszukiwanego elementu na podstawie jego wartości oraz wartości krańców partycji. Zakładamy oczywiście, iż elementy są w miarę równomiernie rozłożone w obrębie partycji.

Po wyznaczeniu pozycji i_{sr} reszta algorytmu jest identyczna jak przy wyszukiwaniu binarnym - sprawdzamy, czy poszukiwany element znajduje się na wyliczonej pozycji. Jeśli tak, zwracamy w zmiennej p tę pozycję i kończymy algorytm. W

przeciwnym razie za nową partycję wyznaczamy lewą lub prawą część partycji względem i_{sr} w zależności od relacji poszukiwanego elementu z elementem zbioru leżącym na pozycji i_{sr} i kontynuujemy pętlę warunkową.
Jeśli pętla warunkowa zakończy się w sposób naturalny, to poszukiwanego elementu w zbiorze nie ma. W takim przypadku pozycja p ma wartość 0 i tę wartość zwracamy jako wynik negatywny przy zakończeniu algorytmu.

Programy

Poniższe, przykładowe programy są praktyczną realizacją omawianego w tym rozdziale algorytmu. Zapewne można je napisać bardziej efektywnie. To już twoje zadanie. Dokładny opis stosowanych środowisk programowania znajdziesz we wstępie. Programy przed opublikowaniem w serwisie edukacyjnym zostały dokładnie przetestowane. Jeśli jednak znajdziesz jakąś usterkę (co zawsze może się zdarzyć), to prześlij o niej informację do autora. Pozwoli to ulepszyć nasze artykuły. Będziemy Ci za to wdzięczni.

Program generuje 100 pseudolosowych liczb z zakresu od 0 do 99. Następnie losowana jest wartość od 0 do 99 i poszukiwana w zbiorze za pomocą wyszukiwania interpolacyjnego. Znaleziony element jest odpowiednio oznaczany. Dodatkowo program zlicza i wypisuje ilość wykonań pętli warunkowej w algorytmie.

Efekt uruchomienia programu

```
Wyszukiwanie interpolacyjne
---------------------------
(C)2006  mgr Jerzy Wałaszek

  0   1   2   2   3   5   6   8   8   9  10  11  11  13  14  15  15  15  15  16
 20  21  25  25  25  26  26  26  28  29  29  30  30  31  31  32  33  33  33  34
 35  35  35  38  40  41  42  42 (48) 49  49  49  50  52  53  54  55  56  58  58
 58  60  61  63  64  65  65  65  65  66  67  68  68  68  69  71  74  74  78  78
 80  81  81  81  81  81  82  83  84  85  85  86  86  89  92  92  96  98  98  99

Ilość prób = 1

Element w = 48 jest na pozycji nr 49

KONIEC. Naciśnij dowolny klawisz...
```

Program w Borland Delphi

```
// Wyszukiwanie interpolacyjne
//----------------------------
// (C)2006  mgr Jerzy Wałaszek
//       I LO w Tarnowie
//----------------------------

program intersearch;

{$APPTYPE CONSOLE}

const
  N = 100;  // Liczba elementów w zbiorze
  M =  99;  // Maksymalna wartość elementu

var
  d                 : array[1..N] of integer;
  w,i,ip,ik,isr,p,l : integer;

begin
  writeln('Wyszukiwanie interpolacyjne');
  writeln('----------------------------');
  writeln('(C)2006  mgr Jerzy Walaszek');
  writeln;

// Generujemy zbiór pseudolosowy

  randomize;
  for i := 1 to N do d[i] := random(M + 1);

// Zbiór sortujemy rosnąco

  for i := N - 1 downto 1 do
  begin
    w := d[i]; ip := i + 1;
    while (ip <= N) and (w > d[ip]) do
    begin
      d[ip - 1] := d[ip]; inc(ip);
    end;
    d[ip - 1] := w;
  end;

// Generujemy poszukiwany element

  w := random(M + 1);
```

```
// Szukamy elementu w

  ip := 1; ik := N; p := 0; l := 0;
  while (d[ip] <= w) and (w <= d[ik]) do
  begin
    isr := ip + (w - d[ip]) * (ik - ip) div (d[ik] - d[ip]);
    inc(l);
    if w = d[isr] then
    begin
      p := isr; break;
    end
    else if w > d[isr] then
      ip := isr + 1
    else
      ik := isr - 1;
  end;

// Prezentujemy wyniki

  for i := 1 to N do
    if i = p then write('(',d[i]:2,')') else write(d[i]:3,' ');
  writeln;
  writeln('Ilosc prob = ',l);
  writeln;
  write('Element w = ',w);
  if p > 0 then
    writeln(' jest na pozycji nr ',p)
  else
    writeln(' w zbiorze nie wystepuje.');
  writeln;

// Gotowe

  writeln('Nacisnij klawisz Enter...');
  readln;
end.
```


```
// Wyszukiwanie interpolacyjne
//----------------------------
```

```
// (C)2006  mgr Jerzy Wałaszek
//       I LO w Tarnowie
//----------------------------

#include <iostream>
#include <iomanip>

using namespace std;

const int N = 100;  // Liczba elementów w zbiorze
const int M =  99;  // Maksymalna wartość elementu

int main(void)
{
  int d[N + 1],w,i,ip,ik,isr,p,l;

  cout << "Wyszukiwanie interpolacyjne\n"
          "---------------------------\n"
          "(C)2006  mgr Jerzy Walaszek\n\n";

// Generujemy zbiór pseudolosowy

  srand((unsigned)time(NULL));
  for(i = 1;i <= N; i++) d[i] = rand() % (M + 1);

// Zbiór sortujemy rosnąco

  for(i = N - 1; i >= 1; i--)
  {
    w = d[i]; ip = i + 1;
    while((ip <= N) && (w > d[ip]))
    {
      d[ip - 1] = d[ip]; ip++;
    }
    d[ip - 1] = w;
  }

// Generujemy poszukiwany element

  w = rand() % (M + 1);

// Szukamy elementu w

  ip = 1; ik = N; p = l = 0;
```

```
  while((d[ip] <= w) && (w <= d[ik]))
  {
    isr = ip + (w - d[ip]) * (ik - ip) / (d[ik] - d[ip]);
    l++;
    if(w == d[isr])
    {
      p = isr; break;
    }
    else if(w > d[isr]) ip = isr + 1; else ik = isr - 1;
  }

// Prezentujemy wyniki

  for(i = 1; i <= N; i++)
    if(i == p)
      cout << "(" << setw(2) << d[i] << ")";
    else
      cout << setw(3) << d[i] << " ";
  cout << "\nIlosc prob = " << l << "\n\nElement w = " << w;
  if(p)
    cout << " jest na pozycji nr " << p << endl;
  else
    cout << " w zbiorze nie wystepuje.\n";
  cout << endl;

// Gotowe

  system("pause"); return 0;
}
```

Program w Microsoft Visual Basic

```
' Wyszukiwanie interpolacyjne
'-----------------------------
' (C)2006  mgr Jerzy Wałaszek
'       I LO w Tarnowie
'-----------------------------

Module Module1

  Sub Main()
    Const N = 100  ' Liczba elementów w zbiorze
```

```
Const M = 99  ' Maksymalna wartość elementu

class="i">Dim d(N), w, i, ip, ik, isr, p, l As Integer

Console.WriteLine("Wyszukiwanie interpolacyjne")
Console.WriteLine("---------------------------")
Console.WriteLine("(C)2006  mgr Jerzy Wałaszek")
Console.WriteLine()

' Generujemy zbiór pseudolosowy

Randomize()
For i = 1 To N : d(i) = Int(Rnd(1) * (M + 1)) : Next

' Zbiór sortujemy rosnąco

For i = N - 1 To 1 Step -1
  w = d(i) : ip = i + 1
  While (ip <= N) AndAlso (w > d(ip))
    d(ip - 1) = d(ip) : ip += 1
  End While
  d(ip - 1) = w
Next

' Generujemy poszukiwany element

w = Int(Rnd(1) * (M + 1))

' Szukamy elementu w

ip = 1 : ik = N : p = 0 : l = 0
While (d(ip) <= w) AndAlso (w <= d(ik))
  isr = ip + (w - d(ip)) * (ik - ip) \ (d(ik) - d(ip))
  l += 1
  If w = d(isr) Then
    p = isr : Exit While
  ElseIf w > d(isr) Then
    ip = isr + 1
  Else
    ik = isr - 1
  End If
End While

' Prezentujemy wyniki
```

```
    For i = 1 To N
      If i = p Then
        Console.Write("({0,2})", d(i))
      Else
        Console.Write(" {0,2} ", d(i))
      End If
    Next
    Console.WriteLine()
    Console.WriteLine("Ilość prób = {0}", l)
    Console.WriteLine()
    Console.Write("Element w = {0}", w)
    If p > 0 Then
      Console.WriteLine(" jest na pozycji nr {0}", p)
    Else
      Console.WriteLine(" w zbiorze nie występuje.")
    End If

    ' Gotowe

    Console.WriteLine()
    Console.WriteLine("KONIEC. Naciśnij dowolny klawisz...")
    Console.ReadLine()

  End Sub

End Module
```


```
# -*- coding: cp1250 -*-
# Wyszukiwanie interpolacyjne
#----------------------------
# (C)2006  mgr Jerzy Wałaszek
#       I LO w Tarnowie
#----------------------------

import random

N = 100  # Liczba elementów w zbiorze
M =  99  # Maksymalna wartość elementu
```

```
d = []

print "Wyszukiwanie interpolacyjne"
print "---------------------------"
print "(C)2006  mgr Jerzy Walaszek"
print

# Generujemy zbiór pseudolosowy

for i in range(N): d.append(random.randint(0, M))

# Zbiór sortujemy rosnąco

d.sort()

# Generujemy poszukiwany element

w = random.randint(0, M)

# Szukamy elementu w

ip, ik, p, l = 0, N - 1, -1, 0
while (d[ip] <= w) and (w <= d[ik]):
  isr = ip + (w - d[ip]) * (ik - ip) // (d[ik] - d[ip])
  l += 1
  if w == d[isr]:
    p = isr
    break
  elif w > d[isr]:
    ip = isr + 1
  else:
    ik = isr - 1

# Prezentujemy wyniki

for i in range(N):
  if i == p:
    print "(%2d)" % d[i],
  else:
    print " %2d " % d[i],
print
print
print "Ilosc prob =", l
print
```

```
 print "Element w =", w,
 if p >= 0:
   print "jest na pozycji nr", p + 1
 else:
   print "w zbiorze nie wystepuje."
 print

 # Gotowe

 raw_input("Nacisnij klawisz Enter...")
```


```
<html>
  <head>
  </head>
  <body>
    <div align="center">
      <form style="BORDER-RIGHT: #ff9933 1px outset; PADDING-RIGHT: 4px;
                   BORDER-TOP: #ff9933 1px outset; PADDING-LEFT: 4px;
                   PADDING-BOTTOM: 1px; BORDER-LEFT: #ff9933 1px outset;
                   PADDING-TOP: 1px; BORDER-BOTTOM: #ff9933 1px outset;
                   BACKGROUND-COLOR: #ffcc66" name="frmInterSearch">
        <h4 id="out_t0" style="text-align: center">Wyszukiwanie interpolacyjne</h4>
        <p style="TEXT-ALIGN: center">
          (C)2006 mgr Jerzy Wałaszek<br>
          I LO w Tarnowie
        </p>
        <hr>
        <p style="TEXT-ALIGN: center">
          <input type="button" value="Szukaj" name="B1" onclick="main();">
        </p>
        <p style="TEXT-ALIGN: center" id="t_out">...</p>
      </form>

<script language=javascript>

// Wyszukiwanie interpolacyjne
//----------------------------
// (C)2006 mgr Jerzy Wałaszek
// I LO w Tarnowie
//----------------------------

function main()
{
  var N = 100; // Liczba elementów w zbiorze
  var M =  99; // Maksymalna wartość elementu
  var d = new Array(N + 1);
```

```
  var w,i,ip,ik,isr,p,l,t;

// Generujemy zbiór pseudolosowy

  for(i = 1; i <= N; i++) d[i] = Math.floor(Math.random() * (M + 1));

// Zbiór sortujemy rosnąco

  for(i = N - 1; i >= 1; i--)
  {
    w = d[i]; ip = i + 1;
    while((ip <= N) && (w > d[ip]))
    {
      d[ip - 1] = d[ip]; ip++;
    }
    d[ip - 1] = w;
  }

// Generujemy poszukiwany element

  w = Math.floor(Math.random() * (M + 1));

// Szukamy elementu w

  ip = 1; ik = N; p = l = 0;
  while((d[ip] <= w) && (w <= d[ik]))
  {
    isr = ip + Math.floor((w - d[ip]) * (ik - ip) / (d[ik] - d[ip]));
    l++;
    if(w == d[isr])
    {
      p = isr; break;
    }
    else if(w > d[isr]) ip = isr + 1; else ik = isr - 1;
  }

// Prezentujemy wyniki

  t = "";
  for(i = 1; i <= N; i++)
  if(i == p)
    t += " <b><font color=red>(" + d[i] + ")</font></b>";
  else
    t += "  " + d[i] + " ";
  t += "<br><br>Ilość prób = " + l + "<br><br>Element w = " + w;
  if(p > 0)
    t += " jest na pozycji nr " + p;
  else
    t += " w zbiorze nie wystepuje";

// Gotowe
```

```
  document.getElementById("t_out").innerHTML = t;
}

</script>
    </div>
  </body>
</html>
```


Źródło: mgr Jerzy Wałaszek