Digitální učební materiál

Číslo projektu:	**CZ.1.07/1.5.00/34.0548**
Název školy:	**Gymnázium, Trutnov, Jiráskovo náměstí 325**
Název materiálu:	**VY_32_INOVACE_153_IVT**
Autor:	**Ing. Pavel Bezděk**
Tematický okruh:	**Algoritmy**
Datum tvorby:	**srpen 2013**
Ročník:	**4. ročník a oktáva**
Anotace:	**Algoritmus XIII. – Třídění dat V. - Merge Sort**
Metodický pokyn:	**Při výuce nutno postupovat individuálně.**

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Autor	**Ing. Pavel Bezděk**		
Vytvořeno dne	**25. 8. 2013**		
Odpilotováno dne	**3. 3. 2014**	**ve třídě**	**8.Y**
Vzdělávací oblast	**Informatika a informační a komunikační technologie**		
Vzdělávací obor	**Informatika a výpočetní technika**		
Tematický okruh	**Algoritmus**		
Téma	**Algoritmus XIII. - Třídění dat V. – Merge Sort**		
Klíčová slova	**Algoritmus, Merge Sort, rekurze**		

Třídění dat

Merge Sort

MergeSort

Řadicí algoritmus, jehož průměrná i nejhorší možná asymptotická časová složitost je ($O(N \log N)$).

Z těchto důvodů je Merge Sort implementován v mnoha programovacích jazycích jako implicitní třídící algoritmus. Algoritmus je velmi dobrým příkladem programátorské metody rozděl a panuj (latinsky divide et impera).

Merge sort pracuje na bázi slévání již seřazených částí pole za pomoci dodatečného pole velikosti **n**. Algoritmus využívá **řazení pomocí slučování** (merging). **Slučování** je proces, kdy **ze dvou či více seřazených sekvencí** vzniká **jedna seřazená sekvence**.

Merge sort byl vynalezen v roce 1945 *Johnem von Neumannem*.

Paměťová složitost: O(N)

algoritmus potřebuje druhé pomocné pole, navíc je ale třeba paměť na realizaci rekurzivních volání (zásobník)

Časová složitost: … **O(N.logN)**

procházíme 1 úsek délky N, 2 úseky délky N/2, 4 úseky délky N/4, …,

celkem hloubka rekurze logN a na každé hladině rekurze práce N

(součet délek K úseků, každý dlouhý N/K prvků) … **O(N.logN)**

Popis Merge Sort – rozděl a panuj

Rozdělit: Jestliže má sekvence **S** více než dva prvky (pokud má 0 nebo jeden prvek, je již seřazená), odeber všechny prvky z **S** a vlož je do dvou sekvencí $\mathbf{S_1}$ a $\mathbf{S_2}$, kde každá bude obsahovat polovinu prvků sekvence **S.**

$\mathbf{S_1}$ obsahuje první polovinu prvků posloupnosti **S**

$\mathbf{S_2}$ obsahuje zbývající prvky posloupnosti **S.**

Rekurze: Rekurzivně setřiď sekvence $\mathbf{S_1}$ a $\mathbf{S_2}$. (Rekurzivně se sekvence stále zmenšují až dvouprvkové sekvence (tříprvková sekvence se dělí na jednu dvouprvkovou, třetí prvek zůstává součástí předchozí rekurze).

Panovat: Vrať elementy do sekvence **S**, sloučením (sléváním) setříděných sekvencí $\mathbf{S_1}$ a $\mathbf{S_2}$ do jediné setříděné sekvence.

Merge Sort Pascal

```
program Mergesort_rekurzivni; {Tridici algoritmus Mergesort - rekurzivni verze}
uses Crt; {Unita pro praci s obrazovkou}
 const MaxN = 100; {maximalni pocet tridenych cisel (dle potreby)}
type  Pole = array[1..MaxN] of integer;      {tridena cisla}
 var P: Pole;     {ulozeni tridenych udaju}
     Q: Pole;     {Q je pomocne odkladaci pole pro realizaci slevani}
     N: 1..MaxN;  {pocet prvku v poli P}
     I: integer;
procedure Mergesort (var P,Q:Pole; Zac,Kon:integer);{Zacatek deklarace procedury }
{setridi v poli P usek od indexu Zac do indexu Kon}
{Q je pomocne odkladaci pole pro realizaci slevani}
var Stred:integer;    {index prostredku trideneho useku}
    i,j,k: integer;   {pomocné indexy pro slevani}
    begin    Stred:=(Zac+Kon) div 2;
      {konec leveho useku pri rozdeleni useku <Zac,Kon>}
    if Zac < Stred then Mergesort(P,Q,Zac,Stred);      {utridit levy usek}
    if Stred+1 < Kon then Mergesort(P,Q,Stred+1,Kon);  {utridit pravy usek}
```

{Máme proceduru MergeSort1(P,Q,1,4), volá procedura MergeSort1 proceduru MergeSort11(P,Q,1,2), která slévá 1. a 2. prvek a pak skončí procedura MergeSort11, a procedura MergeSort1 volá MergeSort12(P,Q,3,4), která slévá 3. a 4. prvek a pak skončí procedura MergeSort12 a pokračuje dále procedura MergeSort1 sléváním 1.,2. a 3.,4. prvku a po slévání skončí procedura MergeSort1}

{Máme proceduru MergeSort1(P,Q,1,3), volá procedura MergeSort1 proceduru MergeSort11(P,Q,1,2) , která slévá 1. a 2. prvek a pak skončí procedura MergeSort11, a procedura MergeSort1 již nevolá MergeSort12(P,Q,3), ale pokračuje procedura MergeSort1 sléváním 1.,2. a 3. prvku a po slévání skončí procedura MergeSort1}

```
{slevani  obou useku:}
 i:=Zac;{levy usek}   j:=Stred+1;{pravy usek}  k:=Zac;{vysledek}
 while (i<=Stred) and (j<=Kon) do
 {slevani setridenych useku do pomocneho pole Q}
      begin if P[i]<=P[j] then
                  begin Q[k]:=P[i]; i:=i+1 end
                    else
                  begin Q[k]:=P[j]; j:=j+1 end;
           k:=k+1
      end;
```

{Může nastat případ, že již i>Stred a j<=Kon, a my jsme ještě neprobrali min. 1 prvek zprava, ale nelze dále pokračovat ve while cyklu nebo již i<=Stred a j>Kon, neprobrali jsme min. 1 prvek zleva, ale nelze již pokračovat ve while cyklu. }

```
    while i<=Stred do   {dokopirovani zbytku leveho useku}
          begin Q[k]:=P[i]; i:=i+1; k:=k+1 end;
 while j<=Kon do     {dokopirovani zbytku praveho useku}
          begin   Q[k]:=P[j]; j:=j+1; k:=k+1 end;
 {zbyva prenest setrideny usek <Zac,Kon> zpet z Q do P:}
 for k:=Zac to Kon do  P[k]:=Q[k]
 end;  {konec deklarace procedury Mergesort}
```

```
 begin  {hlavni program}
  clrscr; {Vymazani obrazovky}
  write('Pocet tridenych cisel: ');
  readln(N);
  writeln;
  writeln('Zadani posloupnosti tridenych cisel:');
  for I:=1 to N do read(P[I]);

  Mergesort(P,Q,1,N);
  writeln;

  writeln('Setridena cisla:');   writeln;
  for I:=1 to N do write(P[I]:5);

  repeat until keypressed;
end.
```

Merge Sort C++

```
#include <iostream> //Merge Sort
using namespace std;
const int POCET=10;
int merge(int p[],int q[],int n);
int main()
{    int pocet=POCET;                    // vstupni parametr - pocet prvku
             int pole[POCET];              // vstupní pole pro trideni
             int pole2[POCET];         // pro mergesort
             int k;
             cout<<"  Zadej prvky pole ke trideni: "<<endl<<endl;
             for (k=0; k<pocet; k++)     {cout<<"   pole["<<k<<"]= "; cin>>pole[k];}
             cout<<endl<<endl;
             merge(pole,pole2,pocet);
             cout<<"  Setridene pole (mergesort): "<<endl;
             for (k=0; k<pocet; k++) {cout<<"  "<<pole2[k];}
   cout<<endl;
             return 0;
}
```

```
int merge(int p[], int q[], int n)
// trideni slucovanim - mergesort, algoritmus se lisi od  standardniho postupu
// v tom, ze nerozdeluji tridene pole  na useky stejne velikosti, nybrz delitkem  je
// inverze mezi  sousednimi prvky. Algoritmus se  diky tomu chova prirozeneji.
{
int * uk;                  // tady pamatovat ukazatel na pole p
int * pom;                 // pro vymenu ukazatelu
int i,j,k,m,l1,r1,l2,r2;   uk=p;    zac:m=0;
            while(m<n)
            { l1=m;r1=l1+1;                    // urcim meze 1.serazeneho useku
              while ((r1<n)&&(p[r1]>p[r1-1])) r1++;
                                                 // a urcim, není-li to cele pole
                        if ((r1==n) &&  (l1==0))
                        {   if (pom==uk)                 // je to pole p?
                                    {    for(k=0;k<n;k++) q[k]=p[k];  }
                                    return 0;          // hotovo
                         }
```

```
if (r1==n)          {for (i=l1;i<r1;i++)  {q[k]=p[i]; k++;}     m=i+1;    }
                    else
                    {           l2=r1; r1--; r2=l2+1;
                                while ((r2<n) && (p[r2]>p[r2-1])) r2++; // 2. usek
                                r2--;  i=l1; k=l1; j=l2;
                                while ((i<=r1) && (j<=r2))
                                {  if (p[i] < p[j])
                                            { q[k]=p[i]; k++; i++; }
                                   else     { if (p[j] < p[i])
                                                {       q[k]=p[j]; j++; k++;       }
                                            }
                                }
```

```
if (i>r1)
                                {
                                        while (j<=r2)
                                        { q[k]=p[j]; j++;  k++; }
                                }
        else
                                if (j>r2)
                                {   while(i<=r1)       { q[k]=p[i]; i++; k++; }
                                }
                        m=r2+1;
                       }
              }
              pom=p;
              p=q;
              q=pom;
              goto zac;
}
```

Dělení

Jak algoritmus rozdělí jeden seznam na dva seznamy. Dělicí část Merge Sortu má na svém vstupu celé pole. Toto rozdělí na dvě stejně velké části, když je N sudé, (N liché S1= (N div 2)+1; S2=(N div 2)), tyto dvě části pak dále rekurzivně dělí. V okamžiku, kdy rekurze rozdělí seznamy na dva dvouprvkové, dělení se zastaví. Popř. tříprvkový seznam se dělí na jeden dvouprvkový, třetí prvek zůstává součástí předchozí rekurze. Dvouprvkový seznam se setřídí již jednoduše. Není-li setříděn, prohodí se prvky.

Slévání

Předpokládejme, že máme dva seznamy ($\mathbf{S_1}$, $\mathbf{S_2}$) seřazené ve vzestupném pořadí. Tyto seznamy můžeme setřídit do jediného seznamu (**S**) následující procedurou. V každém kroku vždy porovnejme první prvky obou seznamů (tj. nejmenší prvky obou seznamů) a vyberme ten menší. Tento nakopírujme na začátek pomocného seznamu **S'** a odstraňme jej ze seznamu původního. Tento postup opakujme tak dlouho, dokud se jeden ze seznamů nevyprázdní. Potom překopírujme zbytek druhého seznamu do pomocného seznamu **S'** a překopírujeme prvky z pomocného seznamu **S'** do seznamu **S** .

Ve skutečnosti slévá merge sort vždy dvě sousední části pole. Drží si dva ukazatele, každý na první prvek seznamu a po každém porovnání přesune jeden z prvků do pomocného pole a posune příslušný ukazatel o jedno místo (čímž se dostane na nový nejnižší prvek příslušného seznamu). Poté, co zkopíruje všechny prvky obou seznamů do pomocného pole, tak celé původní pole přepíše seřazeným seznamem (pomocným polem).

Řazení

Merge sort se tedy začne navracet z rekurze a při každém návratu sleje dva seznamy pomocí výše zmíněné procedury *slévání*. Algoritmus má jistotu, že buď slévá triviálně seřazené prvky nebo seznamy, které již byly slity. V okamžiku, kdy se merge sort plně navrátí z rekurze, tak terminuje (končí). Pole je seřazeno od nejnižší hodnoty po nejvyšší.

Merge Sort lze s výhodou aplikovat na sekvenčních médiích s nižší přístupovou dobou, typicky například u dříve používaných magneto-páskových systémů. Dnes tape drive na serverech (servery pravidelně zálohující na Tape Drive).

Tape drive (streamer) - pásková mechanika pro ukládání dat zařízení, která čte a zapisuje data na magnetickou pásku. Obvykle se používá pro archivaci nebo zálohy dat. Pásková média obecně mají příznivé náklady a dlouhou archivní stabilitu. Pásková jednotka poskytuje sekvenční přístup k ukládání, na rozdíl od disku, který poskytuje libovolný přístup k datům. Na disku lze přesunout data na libovolné místo na disku během několika milisekund, ale pásková jednotka musí fyzicky odmotat pásku a číst nějakou jednu konkrétní část dat, což nám neumožňuje dosahovat vysoké rychlosti přenosu dat. Páskové mechaniky jsou pomalejší, ale např. pro zálohovaní dat jsou vhodné (zálohujeme velké soubory dat, které v případě potřeby kopírujeme zpět na disk (máme pásková zařízení s kapacitou vyšší než 1TB)). Pro sekvenční přístup k datům, zde lze dobře použít Merge Sort.

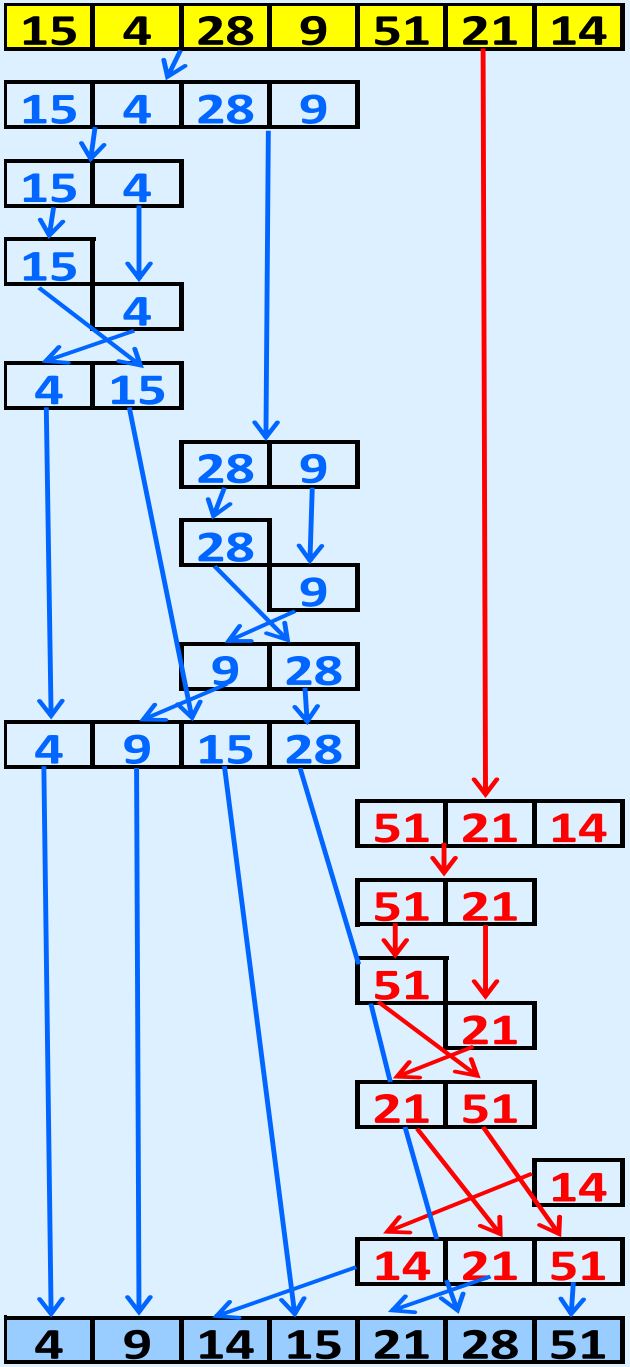

Schéma třídění MergeSort

Je načten nesetříděný seznam. Pro snazší vysvětlení si zjednodušíme parametry volání procedury. Nejprve zavoláme proceduru MergeSort(15,4,28,9,51,21,14), ta volá MergeSort1(15,4,28,9). MergeSort1 volá MergeSort11(15,4), MergeSort11 setřídí sléváním 15 a 4 na seznam 4,15 a MergeSort11 skončí. MergeSort1 pokračuje dále a volá MergeSort12(28,9). MergeSort12 setřídí sléváním 28 a 9 na seznam 9,28 a MergeSort12 končí. MergeSort1 pokračuje dále a setřídí sléváním seznamy 4,15 a 9,28 na seznam 4,9,15,28 a MergeSort1 končí, ale MergeSort ještě stále běží.

MergeSort volá MergeSort2(51,21,14). MergeSort2 volá MergeSort21(51,21), MergeSort21 setřídí sléváním 51 a 21 na seznam 21,51 a MergeSort21 skončí. MergeSort2 pokračuje dále, ale nevolá další proceduru, protože s jedním prvkem se procedura nevolá. MergeSort2 setřídí sléváním 21,51 a 14 na seznam 14,21,51 a MergeSort2 končí. MergeSort pokračuje dále a setřídí sléváním seznamy 4,9,15,28 a 14,21,51 na seznam 4,9,14,15,21,28,51 a MergeSort končí. Vytiskne se setříděný seznam a program skončil. {Ve skutečnosti je procedura MergeSort volána s polem, ve kterém je uložen tříděný seznam, s pomocným polem pro slévání a s počátečním a koncovým indexem tříděného pole.}

Doporučená videa

Merge Sort:

Merge Sort – taneční soubor:
http://www.youtube.com/watch?v=XaqR3G_NVoo

Vizualizace:
http://www.youtube.com/watch?v=A215cCriLy4
http://www.youtube.com/watch?v=_r0gV2hQYf0
http://www.youtube.com/watch?v=63IMRspwdQ8
http://www.youtube.com/watch?v=IR7DFDej6l4

Videa od Unacademy v angličtině:
https://www.facebook.com/unacademy

http://www.youtube.com/watch?v=zkdwpdHLuII
http://www.youtube.com/watch?v=mYywuhRzNPI

Použité zdroje

WIRTH, Niklaus. *Algoritmy a štruktúry údajov*. 2.vyd. Bratislava: Alfa, 1989, 481 s. ISBN 80-050-0153-3.

BÖHM, Martin. *Programátorská kuchařka: Recepty z programátorské kuchařky* [online]. Praha: KSP MFF UK Praha, 2011/2012 [cit. 2013-08-25]. KSP, Korespondenční seminář z programování: Programátorské kuchařky, 24. ročník KSP. Dostupné z: http://ksp.mff.cuni.cz/tasks/24/cook1.html