INSERTIONSORT

```
a(0) := -∞
for i := 2 to n do
    x := a(i)
    j := j-1
    while x < a(j) do
        a(j+1) := a(j)
        j := j-1
    enddo
    a(j+1) := x
endddo
```

třídí se posloupnost $a(1), \ldots, a(n)$

každý prvek počínaje $a(2)$ a konče $a(n)$ se postupně posunuje doleva na svou správnou pozici

po i-tém kroku je část $a(1), \ldots, a(i)$ setříděná

Složitost

porovnání: nejlepší případ $n-1$ $O(n)$

nejhorší $\sum_{i=2}^{n} i = \frac{1}{2}n(n+1)-1$ $O(n^2)$

průměrný ?

výměny (lépe "moves"):

nejlepší případ $2(n-1)$ $O(n)$

(výměna $a(i)$ za $a(i)$)

nejhorší $\sum_{i=2}^{n} (2+i-1) = n-1+\frac{1}{2}n(n+1)-1$

$O(n^2)$

průměrný ?

složitost $= O(n + I(n))$

$I(n)$... počet inverzí ve vstupní permutaci

$0 \le I(n) \le \sum_{i=1}^{n} (n-i) = \sum_{i=0}^{n-1} i = \frac{1}{2}n(n-1) = \binom{n}{2}$

Očekávaný počet inverzí

mám permutaci prvků $1, 2, \dots, n-1$

přidám do ní prvek n

může být vložen na n různých pozic

počet inverzí se může zvýšit o $0, 1, \dots, n-1$

se stejnou pravděpodobností

platí $I(n) = I(n-1) + X_n$

X_n ... náhodná veličina, hodnoty $0, 1, \dots, n-1$

$$P(X_n = k) = \frac{1}{n} \quad \forall k = 0, 1, \dots, n-1$$

$$I(n) = X_n + I(n-1) = X_n + X_{n-1} + I(n-2) = \dots$$

$$= \sum_{j=1}^{n} X_j$$

střední hodnota $E(I(n)) = \sum_{j=1}^{n} EX_j$

$$EX_j = \sum_{k=0}^{j-1} k \cdot \frac{1}{j} = \frac{1}{j} \sum_{k=0}^{j-1} k = \frac{1}{j} \cdot \frac{j(j-1)}{2} = \frac{j-1}{2}$$

$$E(I(n)) = \sum_{j=1}^{n} \frac{j-1}{2} = \frac{1}{2} \cdot \frac{n(n-1)}{2} = \frac{1}{2} \binom{n}{2}$$

očekávaná složitost: $O(n^2)$

Insertionsort je adaptivní na předtříděné posloupnosti (s malým počtem inverzí)

BINÁRNÍ INSERTIONSORT

vylepšuje počet porovnání (nikoli výměn)

Idea: pozice pro prvek a(i) se hledá binárním vyhledáváním v setříděném poli a(1), ..., a(i-1)

Složitost

porovnání $\sum_{i=1}^{n} \lceil \log i \rceil \leq n \log n$ $O(n \log n)$

v nejhorším i v nejlepším případě

výměny $O(n^2)$

Pro nesetříděnou posloupnost je to zlepšení;
pro setříděnou naopak zhoršení!

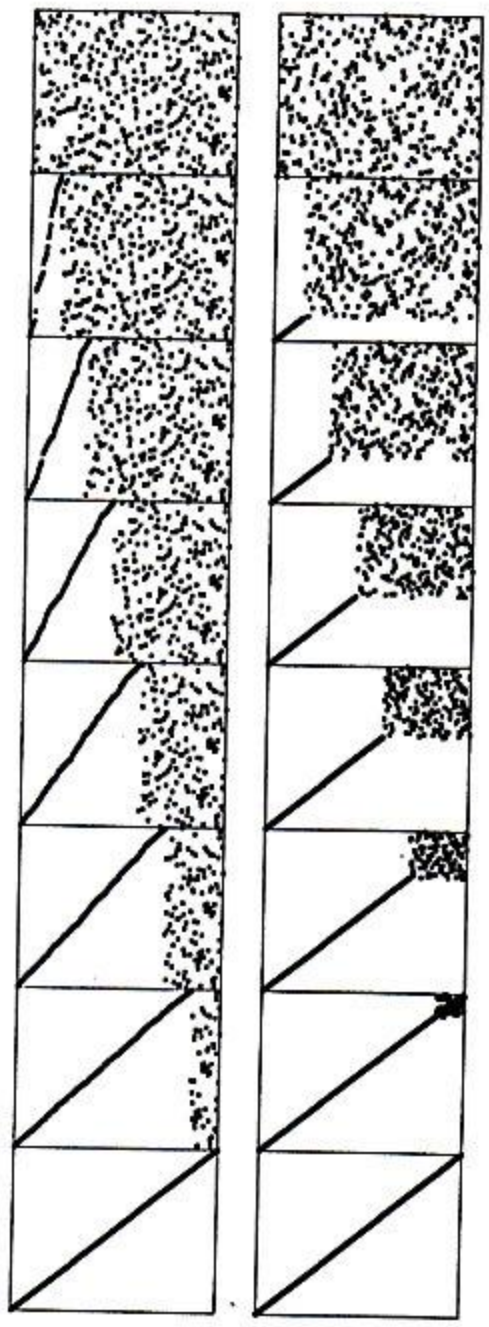


Diagram 6.5 Dynamické charakteristiky vkládacího třídění a výběrového třídění

Tyto snímky z průběhu vkládacího třídění (vlevo) a výběrového třídění (vpravo) nad náhodným uspořádáním ilustrují, jak která metoda při třídění postupuje. Tříděné pole reprezentujeme v grafech `i` *proti* `a[i]` *pro všechna* `i`*. Před tříděním je graf rovnoměrně nahodilý a po třídění se z něj stane úhlopříčná čára vycházející z levého dolního rohu do pravého horního.Vkládací třídění se nikdy z aktuální pozice nedívá před sebe a výběrové třídění se nikdy nedívá za sebe.*

SHELLSORT (D. Shell 1959)

přírůstková' sekvence : $1 = h_1, h_2, \ldots, h_t < n$

```
for k := t downto 1 do
    h := h(k)
    for i := 1 to h do
        Insertionsort(a(i), a(i+h), ..., a(i + ⌊(n-i+1)/h⌋ · h))
    enddo
enddo
```

Idea :

Insertionsort vyměňuje sousední' prvky -
v 1 kroku odstraní' 1 inverzi

Shellsort vyměňuje prvky od sebe více vzdálené'-
v 1 kroku odstraní' více inverzí'

$h_1 = 1$ je nutná' podmínka

Příklad: třídím 44, 55, 12, 42, 94, 18, 6, 67

přírůstkova' sekvence: 1, 2, 4

1. fáze: 4-sort

44, 55, 12, 42, 94, 18, 6, 67

44, 18, 6, 42, 94, 55, 12, 67

2. fáze: 2-sort

44, 18, 6, 42, 94, 55, 12, 67

6, 18, 12, 42, 44, 55, 94, 67

3. fáze: Insertionsort

6, 12, 18, 42, 44, 55, 67, 94

SHELL SORT

```
for k := t downto 1 do
    h := h(k)
    for i := h+1 to n do
        j := i - h
        v := a(i)
        while v > a(j) and j > 0 do
            a(j+h) := a(j)
            j := j - h
        endddo
        a(j+h) := v
    endddo
endddo
```

Odstraňuje 1 vnitřní cyklus - všechny Insertion sorty se dělají najednou

PŘÍRŮSTKOVÉ SEKVENCE

Shell (1959): $1, 2, 2^2, \ldots, 2^t$

nebo $h_t = \lfloor \frac{n}{2} \rfloor$, $h_k = \lfloor \frac{h_{k+1}}{2} \rfloor$, $h_1 = 1$

složitost: nejhorší $\Theta(n^2)$

očekávaná $\Theta(n^{3/2})$

Hibbard (1963): $1, 3, 7, \ldots, 2^k - 1$

složitost: nejhorší $\Theta(n^{3/2})$

očekávaná ? neznámá

Knuth: $h_k = \frac{1}{2}(3^k - 1)$

Papernov-Stasevich: $h_k = 2^k + 1$

složitost: nejhorší $\Theta(n^{3/2})$

očekávaná ? neznámá

Pratt (1971): $1, 2, 3, \ldots, 2^i 3^j$

konstrukce: $2^i 3^j$ pro $i,j = 0, 1, 2, \ldots$

$1, 2, 3, 4, 6, 9, 8, 12, 18, 27, 16, \ldots$

složitost: nejhorší' = nejlepší' = očekávaná' =
$= O(n \log^2 n)$

Sedgewick (1982-6):

$1, 8, 23, \ldots, 4^{j+1} + 2 \cdot 2^j + 1, \ldots$

$1, 5, 65, \ldots, 2 \cdot 4^j - 9 \cdot 2^j + 9, \ldots$

složitost: nejhorší' $\Theta(n^{4/3})$

očekávaná' ? **neznámá'**

Neznámé': Existuje přírůstková' sekvence,
pro kterou je složitost $O(n \log n)$?
(alespoň v průměrném případě)

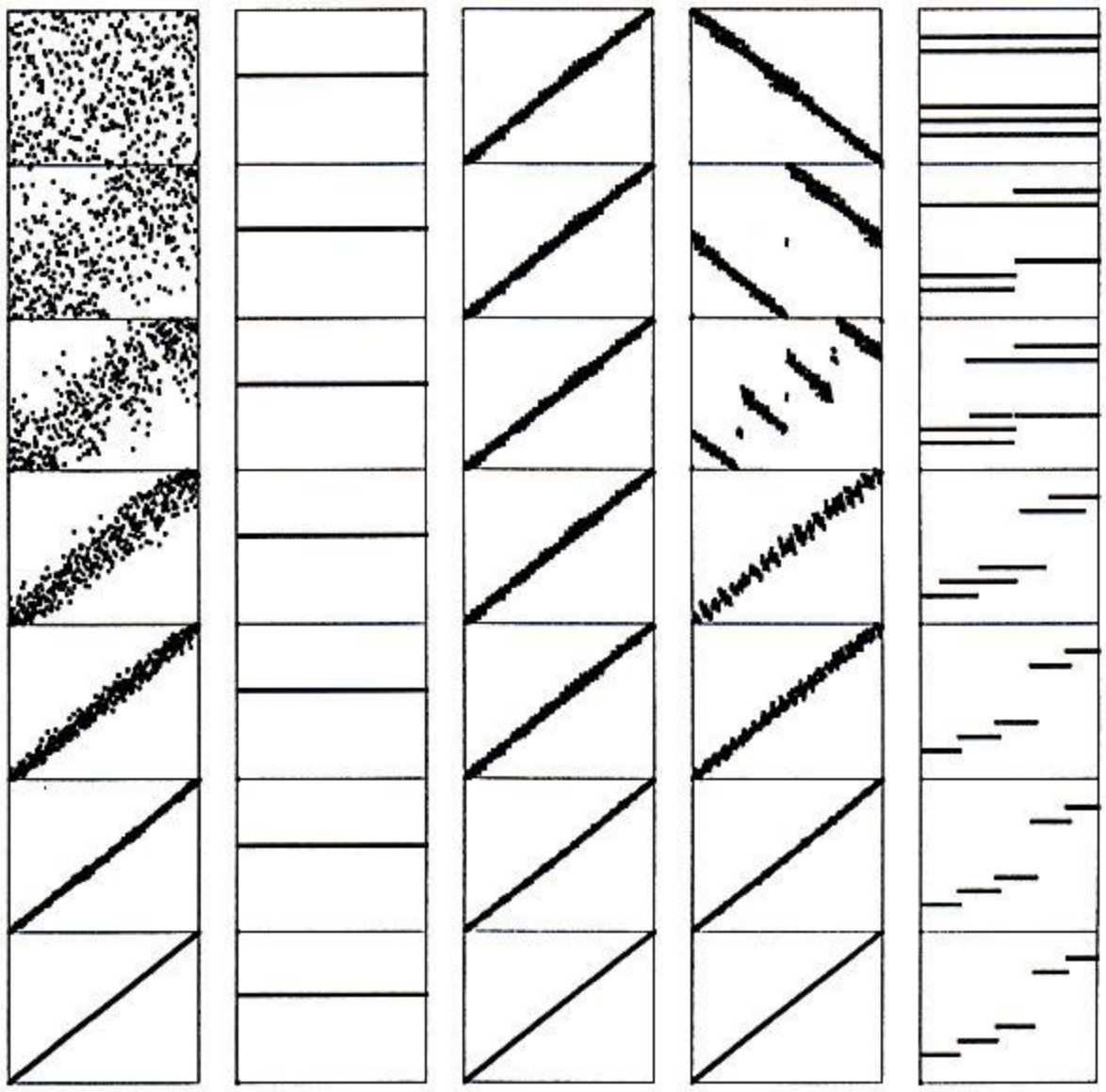


Diagram 6.13 Dynamické charakteristiky shellsortu pro různé typy souborů

Tyto diagramy ukazují shellsort s přírůstky 209 109 41 19 5 1 *při práci se soubory, které jsou nahodilé, gausiánské, téměř uspořádané, reverzně uspořádané a nahodile uspořádané s 10 různými hodnotami klíčů (zleva doprava nahoře). Doba běhu každého průchodu závisí na tom, jak dobře je soubor na začátku průchodu uspořádán. Po několika průchodech jsou tyto soubory podobně setříděny, takže doba běhu není až tak závislá na uspořádání vstupu.*

Shell

přírůstky : $1, 2, \dots, 2^k, \dots, 2^t < n$

Horní odhad: $O(n^2)$

k-tá fáze (h_k-sort) – h_k posloupností

každá má délku $\frac{n}{h_k}$

čas na 1 posloupnost: $O\left(\frac{n}{h_k}\right)^2$

– " – 1 fázi: $h_k\left(\frac{n}{h_k}\right)^2 = \frac{n^2}{h_k}$

– " – všechny fáze: $\sum_{k=1}^{t} \frac{n^2}{h_k} = n^2 \sum_{k=1}^{t} \frac{1}{h_k} =$

$$= n^2 \sum_{k=1}^{t} \left(\frac{1}{2}\right)^k = n^2 \frac{1-\left(\frac{1}{2}\right)^t}{1-\frac{1}{2}} \leq 2n^2$$

Dolní odhad: $\Omega(n^2)$

vstupní permutace: $\frac{n}{2}$ největších prvků na sudých pozicích

$\frac{n}{2}$ nejmenších na lichých pozicích

před poslední fází jsou sudé i liché pozice setříděné

i-tý nejmenší prvek pro $i \leq \frac{n}{2}$ je na pozici $2i-1$

k zatřídění potřebuje $i-1$ posunů

čas na zatřídění lichých prvků: $\sum_{i=1}^{n/2} (i-1) =$

$$= \frac{1}{2}\left(\frac{n}{2}+1\right)\frac{n}{2} - \frac{n}{2} \sim n^2$$

Hibbard

přírůstky: $1, 3, 7, \dots, 2^k-1, \dots < n$

$h_{k+1} = 2h_k + 1 \Rightarrow$ sousední úseky nemají žádné společné dělitele

Horní odhad: $O(n^{3/2})$

h_k-sort: složitost $O(\frac{n^2}{h_k})$ ... použijeme pro $h_k \geq \sqrt{n}$

pro $h_k < \sqrt{n}$:

Platí: před h_k-sortem je posloupnost již $\Rightarrow$

h_{k+1}-setříděná a h_{k+2} setříděná

$a(i) \leq a(i+p)$ kde p je lineární kombinace h_{k+1} a h_{k+2} (s nezápornými celočíselnými koeficienty)

Platí: Protože h_{k+1} a h_{k+2} nemají společné dělitele, je možné všechna čísla $\geq (h_{k+1}-1)(h_{k+2}-1)$ vyjádřit jako lineární kombinaci h_{k+1} a h_{k+2}. (Frobeniův problém)

$(h_{k+1}-1)(h_{k+2}-1) = 8h_k^2 + 4h_k$

To určí hranici, do jaké hloubky až může jít každý vnitřní Insertionsort.

h_k-sort: zatřiďujeme prvek na pozici i

je větší než všechny prvky na pozicích

$i-(8h_k^2+4h_k),\ i-(8h_k^2+4h_k)-1,\ \dots,\ 1$

$\Rightarrow$ v celé posloupnosti je nejvýše $8h_k^2+4h_k$

prvků větších stojících před ním

v jeho podposloupnosti $\frac{8h_k^2+4h_k}{h_k} = 8h_k+4$

celkem n prvků, každý se posune maximálně

o $8h_k+4$ pozic

složitost h_k-sortu: $n(8h_k+4) = O(nh_k)$

<u>Celkem</u>: $h_k = \sqrt{n}$ pro $k = \frac{t}{2}$

$$\sum_{k=1}^{\frac{t}{2}} nh_k + \sum_{k=\frac{t}{2}+1}^{t} \frac{n^2}{h_k} = n\sum_{k=1}^{\frac{t}{2}} h_k + n^2 \sum_{k=\frac{t}{2}+1}^{t} \frac{1}{h_k}$$

1. člen: $h_k = 2^k - 1 \le 2^k$

$$n\sum_{k=1}^{\frac{t}{2}} h_k \le n\sum_{k=1}^{\frac{t}{2}} 2^k = n\cdot 2\cdot \frac{2^{\frac{t}{2}}-1}{2-1} = 2n\left(2^{\frac{t}{2}}-1\right) =$$

$$= 2n\,h_{\frac{t}{2}} \le 2n\sqrt{n} = O(n^{3/2})$$

2. člen: symetricky, $O(n^{3/2})$

<u>Dolní odhad</u>: $\Omega(n^{3/2})$ konstrukcí speciálního případu

Pratt

přírůstky: $1, 2, 3, \ldots, 2^i 3^j, \ldots < n$

počítají se postupně pro $i+j = 0, 1, \ldots$

$1, 2, 3, 2^2, 2\cdot 3, 3^2, 2^3, 2^2\cdot 3, 2\cdot 3^2, 3^3, 2^4, \ldots$

tj. $1, 2, 3, 4, 6, 9, 8, 12, 18, 27, 16, \ldots$ není monotonní (nevadí)

Důležité: při průchodu odkazu každému p

předcházejí 2p a 3p $\Rightarrow$

při každém p-sortu je vstupní posloupnost již

2p-setříděná a 3p-setříděná $\Rightarrow$

podposloupnost vybraná s krokem p je již

2-setříděná a 3-setříděná

Každé $r > 1$ se dá vyjádřit jako lin. kombinace 2 a 3

$\Rightarrow$ každý vnitřní Insertionsort jde jen do hloubky 1

Složitost: $O($délka přírůstkové sekvence $\times\, n) =$

$= O(n \log_2 n \log_3 n) = O(n (\log n)^2)$

Teoreticky nejrychlejší známé verze Shellsortu,

prakticky nedává nejlepší výsledky.

Modifikace:

přírůstky $2^i 5^j$

$2^i 7^j$

atd. – libovolná dvojice nesoudělných čísel

Složitost stále $O(n \log^2 n)$

Vnitřní Insertionsort jde jen do konstantní hloubky

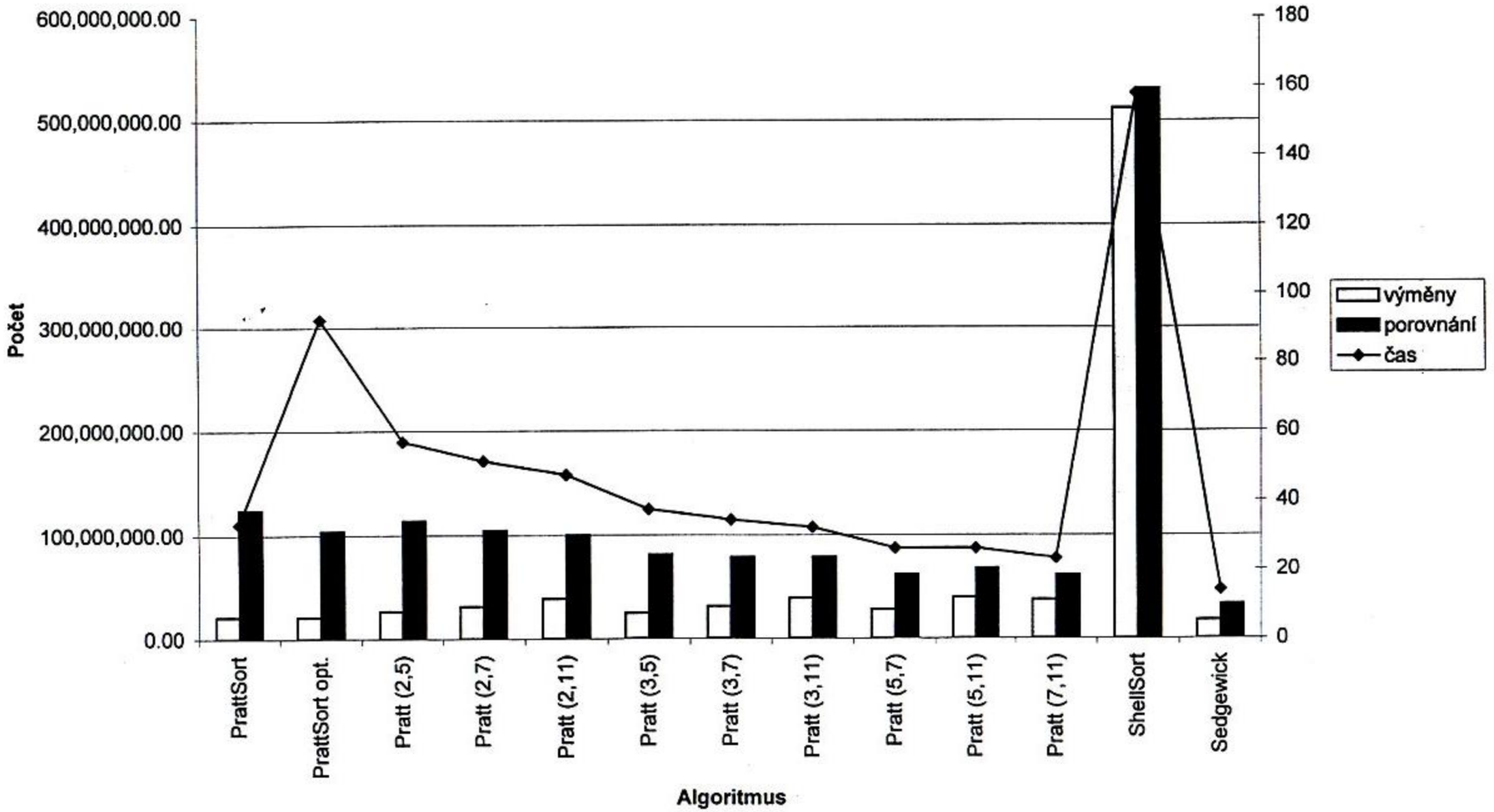
Liu (1997)

používá binární Insertionsort

počet porovnání: $O(n \log n)$

T. Hubička - diplomová práce (1999)

Porovnání výsledků měření - délka 1000000



DALŠÍ PŘÍRŮSTKOVÉ SEKVENCE

Nerovnoměrné varianty klasických přírůstků

Shell - rovnoměrná: $1, 2, \ldots, 2^t < n$

nerovnoměrná: $h_t = \lfloor \frac{n}{2} \rfloor$, $h_k = \lfloor \frac{1}{2} h_{k+1} \rfloor$, $h_1 = 1$

(analogicky Hibbard, Knuth, ...)

Dobosiewicz (1980): $h_t = \lfloor \frac{n}{2} \rfloor$, $h_k = \lfloor \frac{3}{4} h_{k+1} \rfloor$, $h_1 = 1$

původně navržena pro vylepšení Bubblesortu

(analogicky $h_k = \lfloor \frac{2}{3} h_{k+1} \rfloor$ apod.)

Závisí na délce tříděné posloupnosti –

Shell pro $n = 10$: rovnoměrná 1, 2, 4, 8

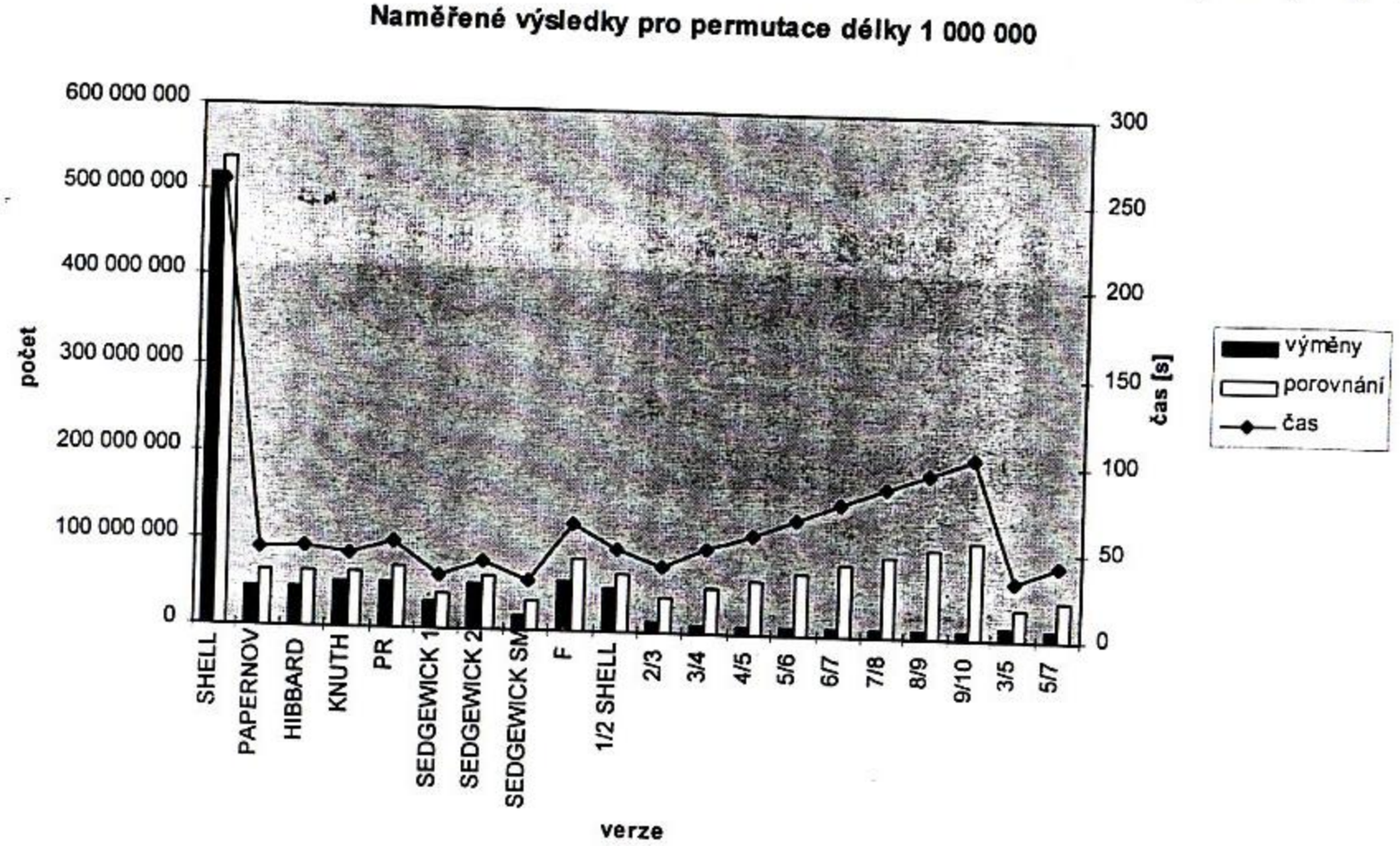
nerovnoměrná 1, 2, 5

Špatně se analyzují

T. Nádvorníková - dipl práce (1997)

Graf 19 - Porovnání měření pro různé verze - permutace délky 1 000 000

Graf zobrazuje průměr naměřených hodnot. V grafu vidíme poměr počtu výměn prvků vůči počtu porovnání prvků a času. U verze Sedgewick 2 bylo vysoké procento prvků po porovnání vyměněno, naopak verze 8/9 a 9/10 prováděly zbytečně moc nevyužitých porovnání (tj. porovnání, po kterých nenásledovala výměna). Naměřený čas koresponduje s počtem porovnání.



Přírůstkové sekvence s konstantním počtem členů

Weiss (1996):

3 přírůstky: $1, \lceil n^{1/3} \rceil, \lceil n^{1/3} \rceil + 1$

složitost $O(n^{5/3})$

4 přírůstky: $1, X, Y, Z$ kde $X = 4^i + 2^i + 1$

$Y = X \cdot 2^i + 1$

$Z = Y + 2^i$

$i = \lceil \frac{\log n}{7} \rceil$

složitost $O(n^{11/7})$

5 přírůstků: $1, A, B, C, D$ kde $A = 4^i + 1$

$B = 64^i$

$C = B \cdot 2^i + 1$

$D = C + 8^i$

$i = \lceil \frac{\log n}{15} \rceil$

složitost $O(n^{23/15})$

Pro $c \geq 6$, $c = 2m^2 - m$ existuje sekvence c přírůstků takových, že složitost Shellsorty je $O(n^{1+1/m})$

OČEKÁVANÁ SLOŽITOST SHELLSORTU

Teoretických výsledků je málo

Knuth

Shellova varianta $1, 2, \ldots, 2^t$ - složitost $O(n^{3/2})$

2-fázový Shellsort s přírůstky $1, h$:

$$\underbrace{\frac{n^2}{4h}}_{\text{1. fáze}} + \underbrace{\frac{1}{8}\sqrt{\pi n^3 h}}_{\text{2. fáze}} + O(n)$$

optimální volba $h = \left(\frac{16}{\pi} n\right)^{1/3}$

optimální očekávaný čas $O(n^{5/3})$

Knuth a Janson (1997)

3-fázový Shellsort s přírůstky $1, g, h$

optimální volba $h = \Theta(n^{7/15})$, $g = \Theta(h^{1/5})$

očekávaný čas $O(n^{23/15})$

Problém - výpočet očekávaného počtu inverzí v předtříděné permutaci (která už není náhodná)

Jiang, Li, Vitányi (1999)

Očekávaná složitost p-stupňového Shellsorty pro libovolnou přírůstkovou sekvenci je $\Omega(pn^{1+1/p})$ pro všechna $p \leq \log n$.

Speciálně:

$p=1$: $\Omega(n^2)$ (Insertionsort)

$p=2$: $\Omega(n^{3/2})$ (Knuth : $O(n^{5/3})$)

$p=3$: $\Omega(n^{4/3})$ (Knuth, Janson : $O(n^{23/15})$)

$p=\log n$: $\Omega(n \log n)$ (obecný dolní odhad)

Důsledek:

Přírůstková sekvence, pro kterou by Shellsort mohl mít očekávanou složitost $O(n \log n)$, by musela mít délku $O(\log n)$.

Experimentální výsledky

Peterson, Russel (1971)

Hibbard: $1{,}22\, n^{1{,}26}$

$0{,}29\, n \ln^2 n - 1{,}26\, n \ln n$

Knuth: $1{,}66\, n^{1{,}25}$

$0{,}33\, n \ln^2 n - 1{,}26\, n \ln n$

Papernov - Stasevich: $1{,}09\, n^{1{,}27}$

$0{,}3\, n \ln^2 n - 1{,}35\, n \ln n$

Weiss (1991)

Hibbard, Knuth: $\Theta(n^{5/4})$

Sedgewick: $\Theta(n^{7/6})$

Hypotéza: pro přírůstkové sekvence délky $\Theta(\log n)$ platí, že nejhorší případ $\Theta(n^k)$ implikuje průměrný případ $\Theta(n^{(k+1)/2})$

M. A. Weiss

30

Short Note

Empirical study of the expected running time of Shellsort

We present the results of a large empirical study of the running time of Shellsort. A previous study reported by Knuth for several increment sequences gave running times between $\Theta(N^{1.25})$ and $\Theta(N^{1.28})$ or $\Theta(N\log^2 N)$, but these forms are not very accurate especially for large permutations.

Our results give a running time of $\Theta(N^{5/4})$ for all of the increment sequences suggested by Knuth and $\Theta(N^{7/6})$ for an increment sequence suggested by Sedgewick. Our fits are accurate to within 1 % for $250 \leqslant N < 100000$ and about 0.1 % for larger N. Much of the error in the fits appears to be related to the error in the measured data. These results are significant because they suggest that $O(\log N)$ increment sequences with $\Theta(N^6)$ worst-case running times have $\Theta(N^{(i+1)/2})$ average-case running times and that there is no increment sequence for which Shellsort is $O(N \log N)$, even on average.

Received March 1990

1. Introduction

Shellsort is a simple sorting algorithm proposed by Shell in 1959.[13] For nearly sorted or mid-sized files (a few thousand elements), Shellsort performs as well as or better than any other known algorithm, including quicksort. Furthermore, it is an in-place sorting algorithm requiring little extra space and is easy to code.

Shellsort uses a sequence of integers $h_t, h_{t-1}, \ldots, h_1$ and works by performing insertion sort on subfiles consisting of elements h_t apart. We call this an h_t-sort. If the input in array a is to be sorted in nondecreasing order, then after an h_t-sort, $a[x] \geqslant a[x-h_t]$ for all $h_t < x \leqslant N$. Typically the increment sequences are 'almost' geometric sequences with $h_k = O(\alpha^k)$ for some α, stopping with h_t being the largest integer in this sequence less than N.

The three most popular increment sequences are Shell's ($h_t = \lfloor h_{t+1}/2 \rfloor$), with $h_1 = \lfloor N/2 \rfloor$. Hibbard's ($h_t = 2^t - 1$) and Knuth's $h_t = \frac{1}{2}(3^t - 1)$).[8] Sedgewick[11] gave a fourth increment sequence which is somewhat more complicated, but seems to perform well in practice. The running times of various sorting algorithms on a SUN 3/60 are given in Table 1, below.

Table 1 shows that Shellsort is faster than heapsort and competitive with quicksort. Quicksort, however, is complicated to implement, and can take quadratic time on sorted input if median-of-three partitioning is not used (see Ref. 12 for a description of all these algorithms). It is thus surprising to find that many books which discuss insertion sort, heapsort, mergesort and quicksort fail to mention Shellsort.[1,3,5,6,9,16] This is probably due to the lack of average-case results for the running time of Shellsort. The only accessible empirical study of Shellsort appears in Knuth,[8] but is rather incomplete and inconclusive. Since the study is twenty years old, it does not cover Sedgewick's increment sequence and essentially extends only to $N = 60000$.

When implemented with Shell's increment sequence, Shellsort can take quadratic time if N is a power of 2, and will take $\Theta(N^{\frac{3}{2}})$ time on average. For example, under the same conditions as in Table 1, if $N = 1048576$, the average time to sort is roughly 22 minutes. Thus this increment sequence, although easy to implement, is a poor choice for Shellsort.

Pratt[10] has shown that for increment sequences of the form $1, \ldots, h_k = c_1\alpha^k + c_2, \ldots, \alpha$ an integer the worst-case running time is $\Theta(N^{\frac{3}{2}})$ (if consecutive increments are relatively prime). This covers many of the increment sequences commonly used, including Hibbard's and Knuth's. Knuth[8] conjectured that the average running time for Shellsort using Hibbard's increment sequence was either $\Theta(N^{1.26})$ or $\Theta(N\log^2 N)$. In addition, the running time using Knuth's sequence was conjectured to be either $\Theta(N^{1.25})$ or $\Theta(N\log^2 N)$. The exponential form was deemed more likely because the coefficients changed less when large files were sorted, and because it was more accurate for smaller files. The problem with the fit is that the running time should grow at the same rate for both increment sequences. Indeed, Knuth gives fits of $\Theta(N^{1.27})$ and $\Theta(N^{1.28})$ for two other (rarely used) increment sequences that should also grow at the same rate. Furthermore, examination of the fits show that this functional form is inaccurate for large values (as well as

Table 1. Running times (in seconds) of various sorting algorithms

	Shellsort					Quicksort	
N	Shell's	Hibbard's	Knuth's	Sedgewick's	Heapsort	Standard	Optimised*
100	0.0024	0.0022	0.0021	0.0017	0.0042	0.0028	0.0024
1000	0.0354	0.0344	0.0306	0.0293	0.0557	0.0315	0.0259
10000	0.5890	0.5563	0.5000	0.4300	0.7165	0.3677	0.3153
100000	9.230	8.408	8.041	5.730	8.859	4.230	3.588
1000000	141.6	130.3	135.4	71.2	104.7	47.1	41.3

* Recursive with median-of-three partitioning and a cutoff of 10.

Table 2

N	Observed exchanges	Our fit	$\Theta(N^{1.25})$ fit	$\Theta(N\log^2 N)$ fit
100	347.6	360.2	400.7	−245.9
500	2950.0	2944.1	3044.4	291.1
1000	7200.8	7139.0	7291.0	2514.6
5000	55831.4	55871.1	55397.2	42232.7
10000	134658.1	134966.1	132673.9	116237.5
50000	1038169.2	1038814.1	1008062.8	1022314.1
100000	2500788.7	2497784.2	2414267.4	2486839.0
500000	19120679.6	19113501.4	18343726.9	18290748.7
1000000	45848881.9	45873300.0	43932444.6	42248509.9
12000000	1058423789.1	1052351112.6	1005897526.8	787733188.2

Table 3

Permutation sizes	Total number of sorts	Machines used
$10i$, $10 \leqslant i \leqslant 99$	16000	16 SUN 3 workstations
$100i$, $10 \leqslant i \leqslant 99$	16000	16 SUN 3 workstations
$1000i$, $10 \leqslant i \leqslant 45$	4000	16 SUN 3 workstations
$100000i$, $5 \leqslant i \leqslant 10$	2000	5 SUN 3 workstations with $\geqslant 8$ Mb
12000000	100	1 VAX 785 with 128 Mb

Table 4

	Hibbard's		Knuth's		Sedgewick's	
N	Average	S.D.	Average	S.D.	Average	S.D.
100	347.6	26.3	432.1	34.3	461.8	37.9
1000	7200.8	361.8	8901.3	446.8	7725.2	192.0
10000	134658.1	6485.6	164960.3	7218.7	108811.7	943.0
100000	2500788.7	116348.3	2963566.6	1331956.0	1409404.5	5281.4
1000000	45848881.9	2.045907.5	53477706.7	2306321.8	17421118.3	36068.8

BUBBLESORT

```
for i := 2 to n do
    for j := n downto i do
        if a(j-1) > a(j) then  x := a(j-1)
                               a(j-1) := a(j)
                               a(j) := x
        endif
    enddo
enddo
```

Průchod zprava doleva, menší prvky se posunují vlevo.
V každém průchodu se minimální člen úseku dostane na svou správnou pozici:
(ex. symetrická varianta - průchod zleva doprava)

Porovnání: $\frac{1}{2} n(n-1) = O(n^2)$ vždy

Výměny: minimálně 0

maximálně $\frac{3}{2} n(n-1) = O(n^2)$

průměrně $\frac{3}{4} n(n-1) = O(n^2)$

V této verzi není adaptivní na předtříděné posloupnosti.

Vylepšení:

1) v každém průchodu počítat výměny –
   když je počet výměn 0, posloupnost je setříděná

2) v každém průchodu zaznamenat pozici, kde došlo
   k poslední výměně – to bude dolní mez pro
   následující průchod

3) **SHAKERSORT**
   střídá průchody zprava doleva a opačně

Bubblesort je nesymetrický – příklad:

8, 9, 10, 2 stačí 1 průchod zprava doleva

10, 2, 8, 9 3 průchody zprava doleva
             1 průchod zleva doprava

Shakersort : 10, 2, 8, 9

← 2, 10, 8, 9

→ 2, 8, 9, 10 2 průchody

Některé prvky se mohou zbytečně posouvat tam a zpět:

10, 2, 8, 9, 1

← 1, 10, 2, 8, 9

→ 1, 2, 8, 9, 10

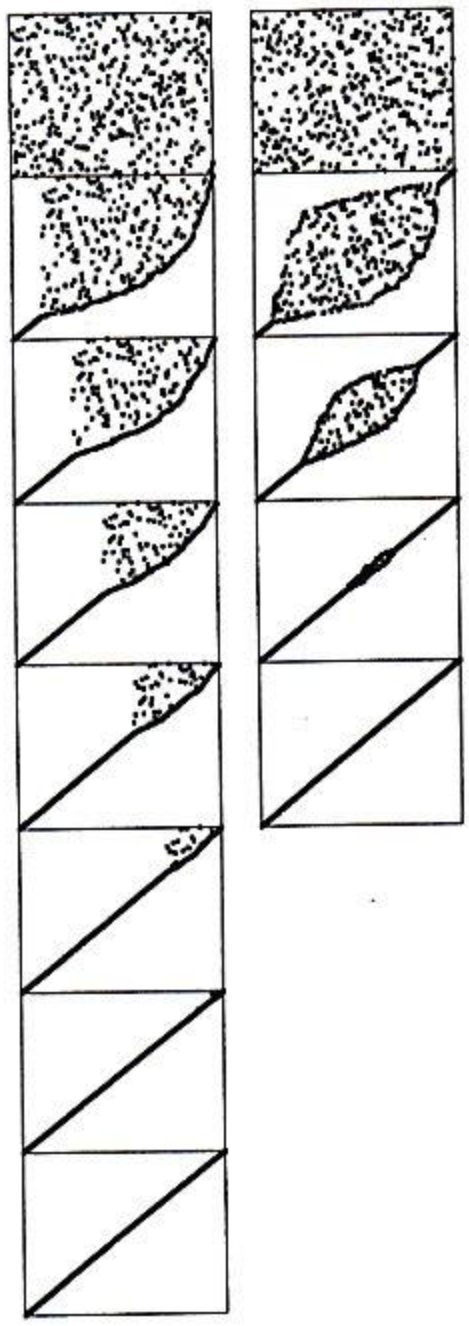


Diagram 6.7 Dynamické charakteristiky dvou bublinkových třídění

Standardní bublinkové třídění (vlevo) pracuje způsobem podobným výběrovému třídění v tom, že v každém průchodu přenese jeden prvek na konečnou pozici a současně, symetrickým způsobem, přinese také určité uspořádání pro druhou část pole. Změnou procházení pole střídáním od začátku do konce a od konce k začátku vznikne verze bublinkového třídění zvaná střásací třídění (shaker sort) (vpravo), která skončí třídění mnohem rychleji (viz cvičení 6.30).

n(
vs
tří
V
tic
oa
vý

Vl
ná
v r

Jal
ná
tut
zer
riti
ny
že
tak
říč

Vl
rov
při

Při
ope
jak
pro
dol
třet
dán
a z
nejl
stru

ně
děn
jež

Vylepšení 1), 2), 3) - složitost stále $O(n^2)$

4) Dobosiewicz (1980)

princip Shellsorty

místo interního Insertionsorty 1 průchod Bublesorty

přírůstky: $h_t = \lfloor \frac{1}{2} n \rfloor$, $h_k = \lfloor \frac{3}{4} h_{k+1} \rfloor$, $h_1 = 1$

Pozor! $h_1 = 1$ nezaručuje, že posloupnost bude setříděna

dotřídit Bublesortem

5) Incerpi, Sedgewick (1987)

princip Shellsorty s libovolnou přírůstkovou sekvencí

místo interního Insertionsorty 1 průchod Shakersorty

dotřídit Insertionsortem

Složitost Brejová (2001):

Existuje přírůstková sekvence, pro kterou 5) má nejhorší případ $O(n^{3k} \log^3 n)$.

4) s přírůstkovou sekvencí délky p má průměrnou složitost $\Omega(n^2/2^p)$

5) s přírůstkovou sekvencí délky p má průměrnou složitost $\Omega(n^2/4^p)$

Praktické výsledky ve srovnání se Shellsortem jsou horší.

SELECTIONSORT

```
for i := 1 to n-1 do
    a(k) := min (a(i), ..., a(n))
    vyměň a(i) s a(k)
enddo
```

```
for i := 1 to n-1 do
    k := i, x := a(i)
    for j := i+1 to n do
        if a(j) < x then k := j, x := a(j) endif
    enddo
    a(k) := a(i)
    a(i) := x
enddo
```

Složitost:

Porovnání: $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = O(n^2)$ vždy

Výměny: $(n-1)$(3 + počet dosazení do x ve vnitřním for-cyklu)

nejlepší případ: $(n-1)(3+0) = O(n)$

nejhorší: $3(n-1)$ + počet porovnání = $O(n^2)$

průměrný: $3(n-1) + Y$

Y = průměrný počet přepsání x ve vnitřním for-cyklu

$$Y = \sum_{i=1}^{n} Y_i$$

Y_i = počet přepsání x při hledání minima v posloupnosti délky i

vstup: náhodná permutace čísel $\{1, \dots, n\}$

$P(a(1) = m) = \frac{1}{n}$ $\forall\, m = 1, \dots, n$

$a(1) = 1 \Rightarrow Y_n = 0$

$a(1) > 1 \Rightarrow$ přepsání x nastane jen pro členy rovné $1, 2, \dots, m-1$

protože členy s hodnotou $> m$ to neovlivní; tak je z posloupnosti rovnou vyloučíme

platí $Y_n = \sum_{m=2}^{n} (1 + Y_{m-1}) \frac{1}{n}$

řešení rekurze:

$$n Y_n = \sum_{m=2}^{n} (1 + Y_{m-1}) = n - 1 + \sum_{m=1}^{n-1} Y_m$$

pro $n+1$:

$$(n+1) Y_{n+1} = n + \sum_{m=1}^{n} Y_m$$

odečteme:

$$(n+1) Y_{n+1} - n Y_n = 1 + Y_n$$

$$Y_{n+1} = \frac{1}{n+1} + Y_n$$

počáteční podmínka: $Y_1 = 0$

řešení: $Y_n = \sum_{m=2}^{n} \frac{1}{m} = H_n - 1 \sim \log n - 1$

$$Y = \sum_{i=1}^{n} Y_i \sim \sum_{i=1}^{n} \log i \leq n \log n$$

Výměny - očekávaný případ: $O(n \log n)$

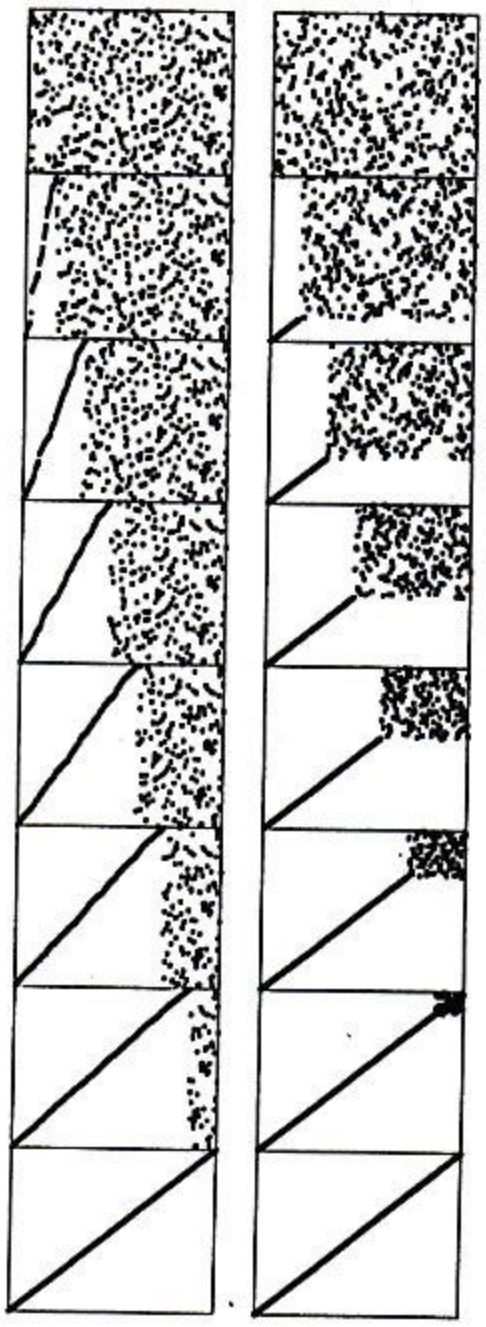


Diagram 6.5 Dynamické charakteristiky vkládacího třídění a výběrového třídění

Tyto snímky z průběhu vkládacího třídění (vlevo) a výběrového třídění (vpravo) nad náhodným uspořádáním ilustrují, jak která metoda při třídění postupuje. Tříděné pole reprezentujeme v grafech `i` *proti* `a[i]` *pro všechna* `i`*. Před tříděním je graf rovnoměrně nahodilý a po třídění se z něj stane úhlopříčná čára vycházející z levého dolního rohu do pravého horního. Vkládací třídění se nikdy z aktuální pozice nedívá před sebe a výběrové třídění se nikdy nedívá za sebe.*

Vylepšení

posloupnost délky n se rozdělí na $\sqrt{n}$ úseků po $\sqrt{n}$ prvcích

v každém úseku se najde minimum:

$\sqrt{n}\,(\sqrt{n}-1) = O(n)$ porovnání

vybere se minimum z minimálních:

$\sqrt{n}-1$ porovnání

v úseku, ze kterého se vzalo minimum, se vybere nové:

$\sqrt{n}-2$ porovnání

vybere se minimum z minimálních:

$\sqrt{n}-2$ porovnání

atd.

Celkem: $O(n\sqrt{n}) = O(n^{3/2})$ porovnání

Friend (1956)

Zobecnění: $\sqrt[3]{n}$ skupin

v každé $\sqrt[3]{n}$ skupinek po $\sqrt[3]{n}$ prvcích

složitost $O(n\sqrt[3]{n}) = O(n^{4/3})$

atd.