Sortowanie Shella Shell Sort

W latach 50-tych ubiegłego wieku informatyk **Donald Shell** zauważył, iż algorytm sortowania przez wstawianie pracuje bardzo efektywnie w przypadku gdy zbiór jest w dużym stopniu **uporządkowany** (sprawdź wyniki naszych badań czasów sortowania dla tego algorytmu). Z kolei algorytm ten pracuje nieefektywnie w zbiorach **nieuporządkowanych**, ponieważ elementy są przesuwane w każdym obiegu o jedną pozycję przy wstawianiu elementu wybranego na listę uporządkowaną.

Pomysł Shella polegał na tym, iż sortowany zbiór dzielimy na podzbiory, których elementy są odległe od siebie w sortowanym zbiorze o pewien odstęp *h*. Każdy z tych podzbiorów sortujemy algorytmem przez wstawianie. Następnie odstęp zmniejszamy. Powoduje to powstanie nowych podzbiorów (będzie ich już mniej). Sortowanie powtarzamy i znów zmniejszamy odstęp, aż osiągnie on wartość 1. Wtedy sortujemy już normalnie za pomocą Instertion Sort. Jednakże z uwagi na wcześniejsze obiegi sortujące mamy ułatwione zadanie, ponieważ zbiór został w dużym stopniu uporządkowany. Dzięki początkowym dużym odstępom elementy były przesuwane w zbiorze bardziej efektywnie - na duże odległości. W wyniku otrzymujemy najlepszy pod względem szybkości czasu wykonania algorytm sortujący w klasie $O(n^2)$. Algorytm ten nosi również nazwę algorytmu **sortowania przez wstawianie z malejącym odstępem** (ang. Diminishing Increment Sort).

Efektywność algorytmu **sortowania metodą Shella** zależy w dużym stopniu od ciągu przyjętych odstępów. Pierwotnie Shell proponował pierwszy odstęp równy połowie liczby elementów w sortowanym zbiorze. Kolejne odstępy otrzymujemy dzieląc odstęp przez 2 (dzielenie całkowitoliczbowe).

Posortujemy metodą Shella zbiór ośmiu liczb: { 4 2 9 5 6 3 8 1 } w porządku rosnącym. Zbiór posiada osiem elementów, zatem przyjmiemy na wstępie odstęp h równy 4. Taki odstęp podzieli zbiór na 4 podzbiory, których elementy będą elementami zbioru wejściowego odległymi od siebie o 4 pozycje. Każdy z otrzymanych podzbiorów sortujemy algorytmem sortowania przez wstawianie. Ponieważ zbiory te są dwuelementowe, to sortowanie pędzie polegało na porównaniu pierwszego elementu podzbioru z elementem drugim i ewentualną zamianę ich miejsc, jeśli będą w niewłaściwym porządku.

	Podział, h=4	Sortowanie	Wynik
	4 2 9 5 6 3 8 1 - *Zbiór wejściowy*		
1	4 6	4 6	4 6
2	2 3	2 3	2 3
3	9 8	8 9	8 9
4	5 1	1 5	1 5
		Zbiór wyjściowy -	4 2 8 1 6 3 9 5

Zmniejszamy odstęp h o połowę, więc $h = 2$. Zbiór podstawowy zostanie podzielony na dwa podzbiory. dy z tych podzbiorów sortujemy przez wstawianie:

	Podział, h=2	Sortowanie	Wynik
	4 2 8 1 6 3 9 5 - *Zbiór wejściowy*		
1	4 8 6 9	4 6 8 9	4 6 8 9
2	2 1 3 5	1 2 3 5	1 2 3 5
		Zbiór wyjściowy -	4 1 6 2 8 3 9 5

Zmniejszamy odstęp h o połowę, $h = 1$. Taki odstęp nie dzieli zbioru wejściowego na podzbiory, więc teraz będzie sortowany przez wstawianie cały zbiór. Jednak algorytm sortujący ma ułatwioną pracę, ponieważ dzięki poprzednim dwóm obiegom zbiór został częściowo uporządkowany - elementy małe zbliżyły się do początku zbioru, a elementy duże do końca.

	Podział, h=1	Sortowanie	Wynik
	4 1 6 2 8 3 9 5 - *Zbiór wejściowy*		
1	4 1 6 2 8 3 9 5	1 2 3 4 5 6 8 9	1 2 3 4 5 6 8 9
		Zbiór wyjściowy -	1 2 3 4 5 6 8 9

Dobór optymalnych odstępów

prof. Donald Knuth

Kluczowym elementem wpływającym na efektywność **sortowania metodą Shella** jest właściwy dobór **ciągu odstępów**. Okazuje się, iż ciąg zaproponowany przez twórcę algorytmu jest jednym z najgorszych, ponieważ w kolejnych podzbiorach uczestniczą wielokrotnie te same elementy (możesz to prosto sprawdzić w podanym powyżej przykładzie).

Dotąd problem optymalnych odstępów w algorytmie sortowania metodą Shella nie został rozwiązany matematycznie, ponieważ w ogólnym przypadku jest niezwykle trudny. Wielu badaczy proponowało na wybór tych odstępów różne ciągi liczbowe otrzymując lepsze lub gorsze rezultaty.

Na przykład profesor Donald Knuth (autor "Sztuki Programowania Komputerów" - "The Art of Computer Programming") zbadał bardzo dokładnie algorytm sortowania metodą Shella i doszedł do wniosku, iż dobry ciąg odstępów dla n elementowego zbioru można wyznaczyć następująco:

Przyjmujemy $h_1 = 1$

obliczamy $h_s = 3h_{s-1} + 1$ aż do momentu gdy $h_s \geq n$

Ostatecznie $h = [h_s : 9]$

Obliczmy pierwszy odstęp dla zbioru o $n = 200$ elementach:

$h_1 = 1$

$h_2 = 3h_1 + 1 = 3 + 1 = 4$ - mniejsze od 200, kontynuujemy

$h_3 = 3h_2 + 1 = 12 + 1 = 13$ - kontynuujemy

$h_4 = 3h_3 + 1 = 39 + 1 = 40$ - kontynuujemy

$h_5 = 3h_4 + 1 = 120 + 1 = 121$ - kontynuujemy

$h_6 = 3h_5 + 1 = 363 + 1 = 364$ - stop, ponieważ jest większe od 200

$h = [h_6 : 9] = [364 : 9] = [40\ ^4/_9] = 40$ (zwróć uwagę, iż jest to zawsze element wcześniejszy o dwie pozycje, czyli h_4)

Kolejne odstępy obliczamy dzieląc całkowitoliczbowo bieżący odstęp przez 3. Taki właśnie sposób wyliczania odstępów przyjmiemy w naszym algorytmi

Specyfikacja problemu

Dane wejściowe

n - liczba elementów w sortowanym zbiorze, n Î N

d[ ] - zbiór n-elementowy, który będzie sortowany. Elementy zbioru mają indeksy od 1 do n.

Dane wyjściowe

d[ ] - posortowany zbiór n-elementowy. Elementy zbioru mają indeksy od 1 do n.

Zmienne pomocnicze

i - indeks elementu listy uporządkowanej, i Î N

j - zmienna sterująca pętli, j Î N

h - odstęp pomiędzy kolejnymi elementami podzbiorów, h Î N

x - zawiera wybrany ze zbioru element

Lista kroków

K01: h ← 1

K02: Powtarzaj h ← 3h + 1, aż h ≥ n

K03: h ← h **div** 9

K04: Jeśli h = 0, to h ← 1

K05: Dopóki h > 0: wykonuj kroki K06...K10

K06: Dla j = n - h, n - h - 1, ..., 1: wykonuj kroki 7...9

K07: x ← d[j]; i ← j + h

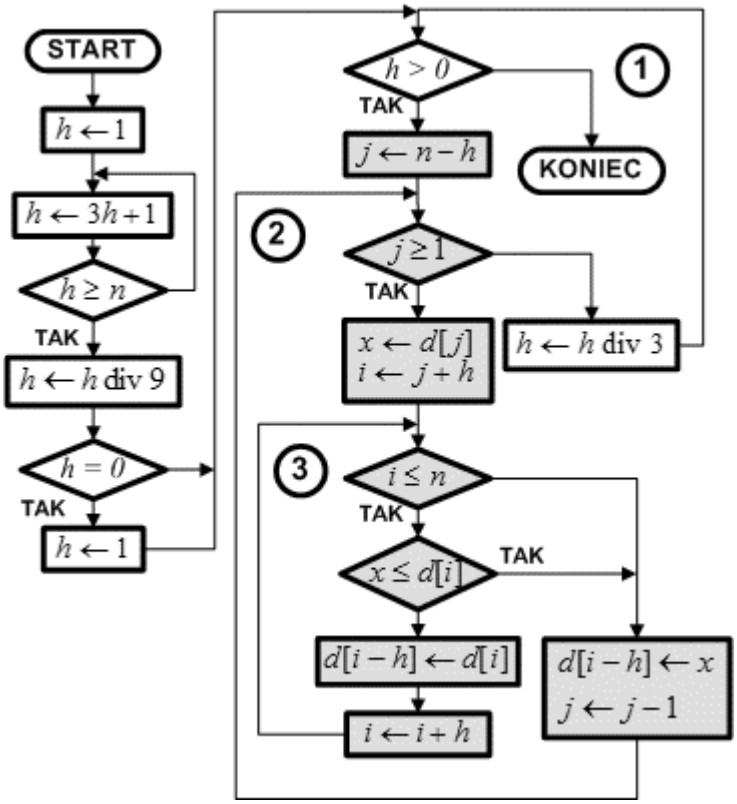
K08: Dopóki (i ≤ n) i (x > d[i]): wykonuj d[i - h] ← d[i]; i ← i + h

K09: d[i - h] ← x

K10: h ← h **div** 3

K11: Zakończ algorytm

Schemat blokowy



Algorytm sortowania metodą Shella jest ulepszonym algorytmem sortowania przez wstawianie. Aby się o tym przekonać, wystarczy spojrzeć na schemat blokowy. Kolorem szarym zaznaczyliśmy na nim bloki, które dokładnie odpowiadają algorytmowi sortowania przez wstawianie. Jedyną modyfikacją jest wprowadzenie odstępu *h* zamiast liczby 1.

Na początku algorytmu wyznaczamy wartość początkowego odstępu *h*. Wykorzystujemy tu sugestie prof. Donalda Knutha.

Po wyznaczeniu *h* rozpoczynamy pętlę warunkową nr 1. Pętla ta jest wykonywana dotąd, aż odstęp *h* przyjmie wartość 0. Wtedy kończymy algorytm, zbiór będzie posortowany.

Wewnątrz pętli nr 1 umieszczony jest opisany wcześniej algorytm sortowania przez wstawianie, który dokonuje sortowania elementów poszczególnych podzbiorów wyznaczonych przez odstęp *h*. Po zakończeniu sortowania podzbiorów odstęp *h* jest zmniejszany i następuje powrót na początek pętli warunkowej nr 1.

Uwaga techniczna:

Zwróć uwagę, iż każdy obieg pętli nr 2 sortuje przemiennie jeden element z kolejnych podzbiorów. Najpierw będą to elementy przedostatnie w kolejnych podzbiorach wyznaczonych odstępem h, później wcześniejsze i wcześniejsze. Takie podejście znacząco upraszcza algorytm sortowania.