Shell sort

Shell sort: En raskere algoritme

- Aka. inkrementell sortering og “gap sort” *
- Sorterer f.eks. hvert 100. element innbyrdes med instikksortering, deretter hvert 50. element, hvert 25. element, hvert 12. element etc., og tilslutt alle elementer i siste gjennomgang
- “Grovsorteringen” går raskt for store “gaps”
- Hele sorteringen går mye raskere fordi vi hele tiden bruker innstikksortering på en array som etterhvert vil være “nesten sortert”

*: Først publisert av Donald Shell, 1959

Shell sort: Eksempel

Shell sort med “gap”-sekvens 5, 3 og 1:

Gap 5:	81 94 11 96 12 35 17 95 28 58 41 75 15
“5-sort”:	35 17 11 28 12 41 75 15 96 58 81 94 95
Gap 3:	35 17 11 28 12 41 75 15 96 58 81 94 95
“3-sort”:	28 12 11 35 15 41 58 17 94 75 81 96 95
Gap 1:	28 12 11 35 15 41 58 17 94 75 81 96 95
Ferdig:	11 12 15 17 28 35 41 58 75 81 94 95 96

En annen måte å se på Shell sort

- Begynner med å dele opp arrayen i mange små lister, som sorteres hver for seg
- Etter hvert deler vi opp i færre og færre lister som stadig blir lengre
- Antall elementer som står feil avtar etterhvert som listene blir lengre, lange lister vil være “nesten sortert”
- Instikksortering blir derfor effektivt å bruke fordi det etterhvert er svært lite swapping
- Se eksempel på neste side med n = 16, der vi bruker gap-sekvensen: 8, 4 , 2, 1

23 42 18 96 77 30 11 87 54 29 74 16 56 48 79 33

23 42 18 96 77 30 11 87 | 23 29 18 16 56 30 11 33
54 29 74 16 56 48 79 33 | 54 42 74 96 77 48 79 87

23 29 18 16 56 30 11 33 54 42 74 96 77 48 79 87

23 29 18 16 | 23 29 11 16
56 30 11 33 | 54 30 18 33
54 42 74 96 | 56 42 74 87
77 48 79 87 | 77 48 79 96

23 29 11 16 54 30 18 33 56 42 74 87 77 48 79 96

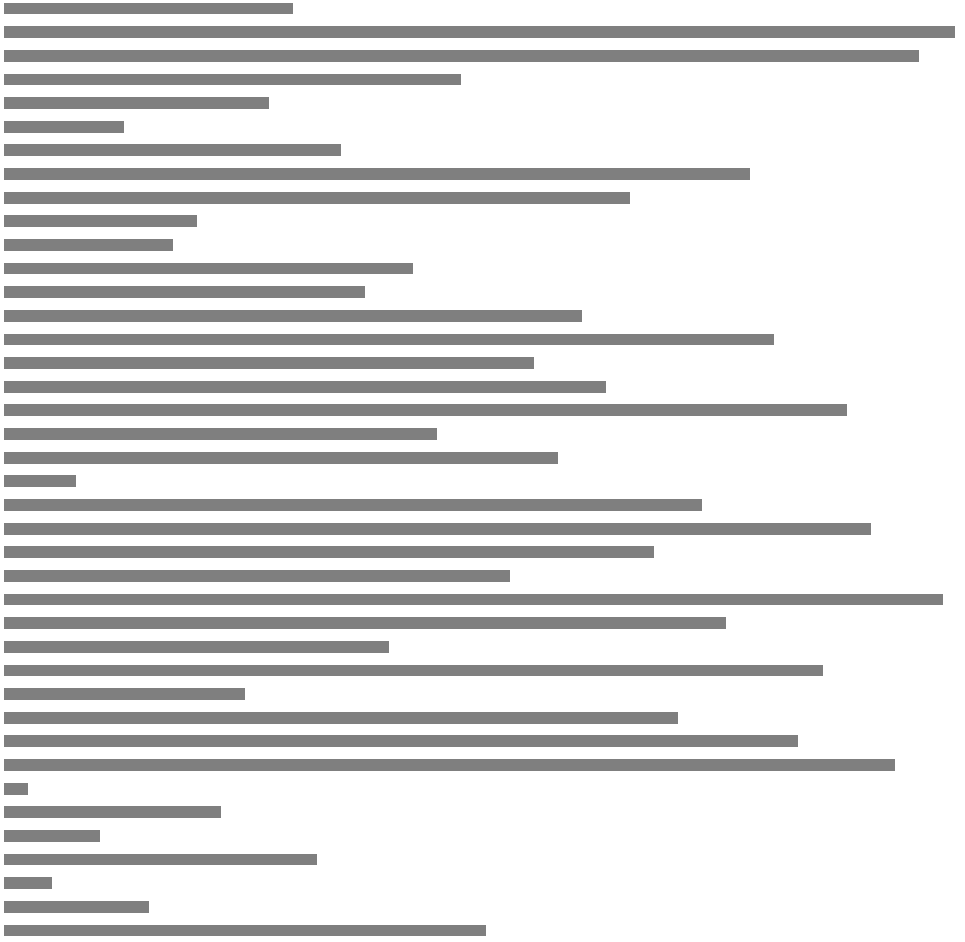
23 29 | 11 16
11 16 | 18 29
54 30 | 23 30
18 33 | 54 33
56 42 | 56 42
74 87 | 74 48
77 48 | 77 87
79 96 | 79 96

11 16 18 29 <u>23</u> 30 54 <u>33</u> 56 <u>42</u> 74 <u>48</u> 77 87 <u>79</u> 96

Implementering av Shell sort

- Shell sort programmeres med tre løkker, programkoden blir forbausende(?) enkel
- Ytre løkke går gjennom hele gap-sekvensen, f.eks. $n/2$, $n/4$, $n/8$, $n/16$, …, 1
- De to indre løkkene gjør innstikksortering av hver av de mindre listene som fremkommer for hvert "gap"
- Se Java-koden

Shell sort: Animasjon



Effektivitet av Shell sort

- Kan bevises at gjennomsnittlig arbeidsmenge for standard Shell sort er ca. $O(n^{3/2}) = O\left(n\sqrt{n}\right)$
- Er *mye* raskere enn $O(n^2)$-metoder for store n
- Effektiviteten avhenger av valget av "gap"-sekvens:
  - Shells opprinnelige forslag: $n/2$, $n/4$, $n/8$,..., 2, 1
  - Bedre: Rund av hvert "gap" til nærmeste oddetall
  - Erfaring viser at en enda bedre sekvens er:

    $n/2$, $(n/2) / (2.2)$, $(n/2) / (2.2^2)$, $(n/2) / (2.2^3)$, ..., 1
- Se testprogram

Andre sekvensielle sorteringer

- Finnes mange andre forbedringer og varianter av sekvensielle algoritmer i tillegg til Shell sort, f.eks:
  - Comb sort
  - Cocktail shaker sort
  - Odd-even sort
- Alle disse er polynomiske, og vil derfor være mindre effektive for store verdier av n enn "smarte" logaritmiske metoder som er $O(n \log(n))$