Data Structures and Algorithms (8)

Instructor: Ming Zhang
Textbook Authors : Ming Zhang , Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

https://courses.edx.org/courses/PekingX/04830050x/2T2014/

Overview

8.2 Insertion Sort

- 8.2.1 Direct insertion Sort
- 8.2.2 Shell Sort

Animation of Insertion Sort

Algorithm of Insertion Sort

12	34	45	78	34’

```
template <class Record>
void ImprovedInsertSort (Record Array[], int n){
//Array[] is the unsorted sequence, n is its length
    Record TempRecord;      // temporary variable
    for (int i=1; i<n; i++){    // insert the ith record in turn
    TempRecord = Array[i];
        //find the correct position for i from i
        int j = i-1;
        //move the records bigger than or equal to i backwards
        while ((j>=0) && (TempRecord < Array[j])){
            Array[j+1] = Array[j];
            j = j - 1;
        }
        //now the one after j is the correct position of i, fill it in
        Array[j+1] = TempRecord;
    }
}
```

Algorithm Analysis

- Stable
- Space Cost : $\Theta(1)$
- Time Cost :
  - Best Case : n-1 comparisons , 2(n-1) movements , $\Theta(n)$
  - Worst Case : $\Theta(n^2)$
    - Comparisons: $\sum_{i=1}^{n-1} i = n(n-1)/2 = \Theta(n^2)$
    - Movements: $\sum_{i=1}^{n-1} (i+2) = (n-1)(n+4)/2 = \Theta(n^2)$
  - Average : $\Theta(n^2)$

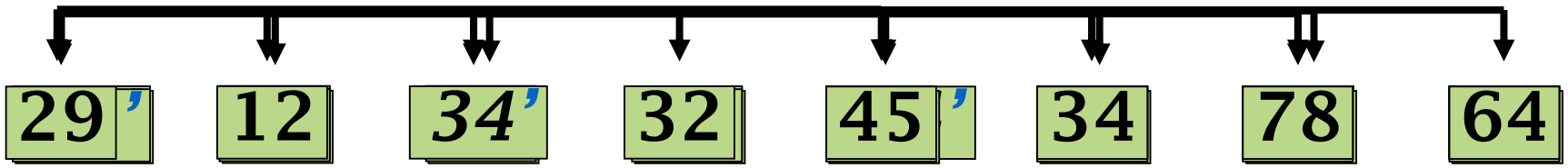
8.2.2 Shell Sort

- Two features of direct insertion sort :
  - In the best case (sequence is in order) , time cost is $\Theta(n)$
  - For short sequence , direct insertion sort is effective
- Shell Sort takes full advantage of these two features of direct insertion sort.

Algorithm of Shell Sort

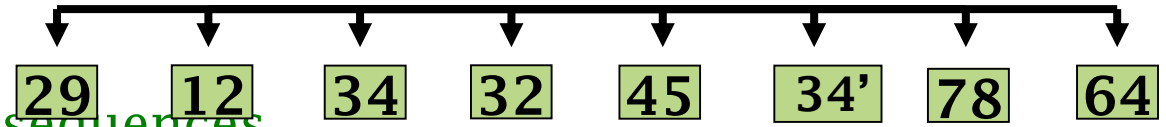
- Transform the sequence into small sequences, do insertion sort in these small sequences.
- Increase the scale of small sequences gradually and reduce the number of small sequences to make the sequence in a more ordered state.
- At last, do the direct insertion sort for the whole sequence for rounding off to complete.

The Animation of Shell Sort



Shell Sort with Increment Decreasing by being divided by 2

```
template <class Record>
void ShellSort(Record Array[], int n) {
// Shell Sort, Array[] is the unsorted array, n is its length
    int i, delta;
    // The increment delta decreases by being divided by 2
    for (delta = n/2; delta>0; delta /= 2)
        for (i = 0; i < delta; i++)
        // do the insertion sort for the delta subsequences
        //"&" passes the address of Array[i], the length of array
is n-i
            ModInsSort(&Array[i], n-i, delta);
    // If the increment sequence can not guarantee the last
    // delta is 1, the following insertion for rounding off can be
    // used: ModInsSort(Array, n, 1);
}
```



Insertion Sort Modified for Increment

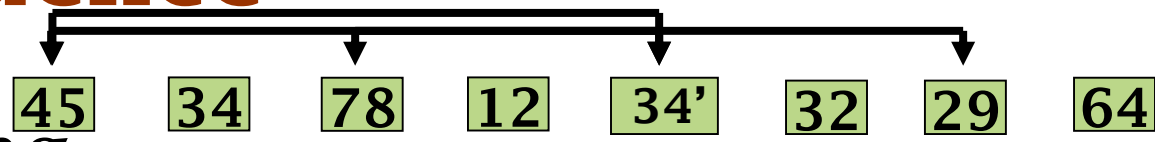
```
template <class Record> // delta means current increment
void ModInsSort(Record Array[], int n, int delta) {
    int i, j;
    // For the ith record of subsequence, find appropriate position
    for (i = delta; i < n; i += delta)
            // j find the reversed pair forward in step of delta
            for (j = i; j >= delta; j -= delta)  {
                if (Array[j] < Array[j-delta]) // reversed pair
                        swap(Array, j, j-delta);// exchange
                else break;
            }
}
```

Algorithm Analysis

- Unstable
- Space Cost : $\Theta(1)$
- Time Cost
  - Increment decreased by being divided by 2, $\Theta(n^2)$
- Choose appropriate increment sequence
  - Make the time approximate to $\Theta(n)$

8.2.2 Shell Sort

Choose Increment Sequence for Shell Sort



- Increment decreased by beding divided by 2
  - Efficiency is still $\Theta(n^2)$
- Problem : Increments chosen are not coprime
  - Subsequences with interval 2^{k-1} , are all made up of the subsequences with interval of 2^k
  - These subsequences are all in order in the last sort, which makes the efficiency low.

Hibbard Increment Sequence

- Hibbard Increment Sequence
  - $\{2^k - 1 , 2^{k-1} - 1 , \ldots , 7 , 3 , 1\}$
- Shell(3) and Shell Sort with Hibbard Increment Sequence can reach the efficiency of $\Theta(n^{3/2})$.
- Select other increment sequences can reduce the time cost much further.

The Best Cost of Shell

- a series of integer in the form of 2^p3^q :
  - 1, 2, 3, 4, 6, 8, 9, 12
- $\Theta(n\ (\log_2 n)^2)$

8.2.2 Shell Sort

Thinking

- 1. Variation of insertion sort
  - Exchange as soon as reversed pairs is found
  - Find the position for insertion, use binary search

- 2. What is the increment of Shell Sort used for ? Which one is better, the sequence with increment 2 or increment 3 ? Why ?

- 3. Can other methods be used for the sort of subsequence in each round of Shell sort ?

Data Structures and Algorithms

Thanks

The National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)