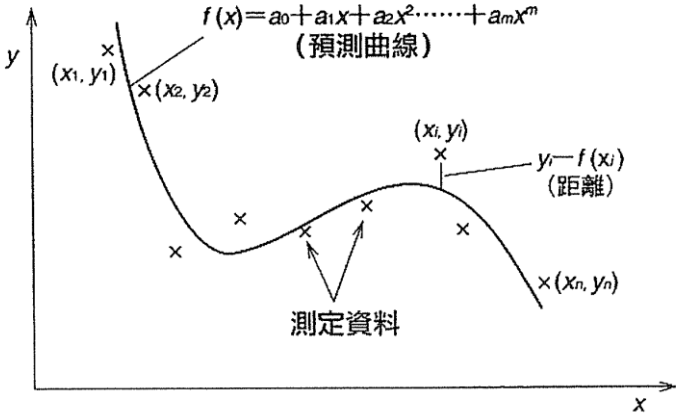
Advanced Class Lab05 04/21

1. Least Square Algorithm

The method of least squares is a standard approach to the approximate solution of overdetermined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation.



Take following figure as example, there are n training data, $(x_i, y_i)(i = 1,2,\cdots,n)$**.**

We need to decide coefficients $a_0 \sim a_m$ **to make the square sum of distance**

$\left[\sum_{i=1}^{n}(y_i - f(x_i))^2\right]$ **between approximate equation**

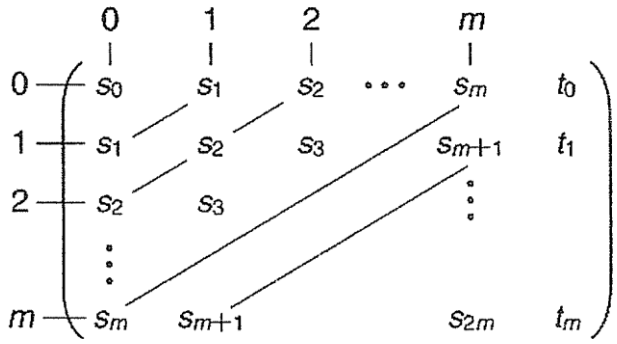
$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$ **and training data be the least. We can obtain the coefficients by solving following simultaneous equations of which m is suitable order of this curve.**

$$\begin{cases} s_0 a_0 + s_1 a_1 + s_2 a_2 + s_3 a_3 \cdots + s_m a_m = t_0 \\ s_1 a_0 + s_2 a_1 + s_3 a_2 + s_4 a_3 \cdots + s_{m+1} a_m = t_1 \\ s_2 a_0 + s_3 a_1 + s_4 a_2 + s_5 a_3 \cdots + s_{m+2} a_m = t_2 \\ \vdots \\ s_m a_0 + s_{m+1} a_1 + s_{m+2} a_2 + \cdots + s_{2m} a_m = t_m \end{cases}$$

but

$$\begin{cases} s_0 = \sum_{j=1}^{n} x_j^0 \\ s_1 = \sum_{j=1}^{n} x_j^1 \\ \vdots \\ s_{2m} = \sum_{j=1}^{n} x_j^{2m} \end{cases} \quad \textbf{and} \quad \begin{cases} t_0 = \sum_{j=1}^{n} y_j x_j^0 \\ t_1 = \sum_{j=1}^{n} y_j x_j^1 \\ \vdots \\ t_m = \sum_{j=1}^{n} y_j x_j^m \end{cases}$$

You can solve it by Gauss-Jordan method, so you will need following coefficient matrix.



If the training data sets are $(-3,5),(-2,-2),(-1,-3),(0,-1),(1,1),(2,4),(3,5)$**, please printout other data by least square algorithm when** $x = -3.0,-2.5,-2.0,-1.5,\cdots,1.5,2.0,2.5,3.0$.

The result is as follows.

x	y
-3.0	5.0
-2.5	0.3
-2.0	-2.1
-1.5	-2.9
-1.0	-2.8
-0.5	-2.2
0.0	-1.3
0.5	-0.1
1.0	1.2
1.5	2.6
2.0	3.9
2.5	4.9
3.0	5.0

You should implement a function with prototype:
double *LeastSquare(double *x, double *y, int n, int m);
which receives training data array x and y, the size of array n, and the least square order m, and returns the (m+1) coefficients computed by least square method.
You are NOT allowed to modify any data in x and y while computing the Least Square.

2. Shaker Sort Algorithm

Bubble sort repeats comparison no matter when series is sorted in order or not, so it is inefficient. When it sweeps from beginning until exchanging location, i and i+1, the i+1~n-1 elements should be sorted in order, so you do not need to sweep them. As a result, we can store the final exchanging location into a variable "shift", and set it as sweep range in next sweeping. But if there are smaller elements in end part, the final exchange will be done in end part, the effect of "shift" will be not so obvious. So we can do sweep in two directions, from beginning to end and from end to beginning with interaction, and the process is so called shaker sort. You can refer to following figure to understand the detail of shaker sort.

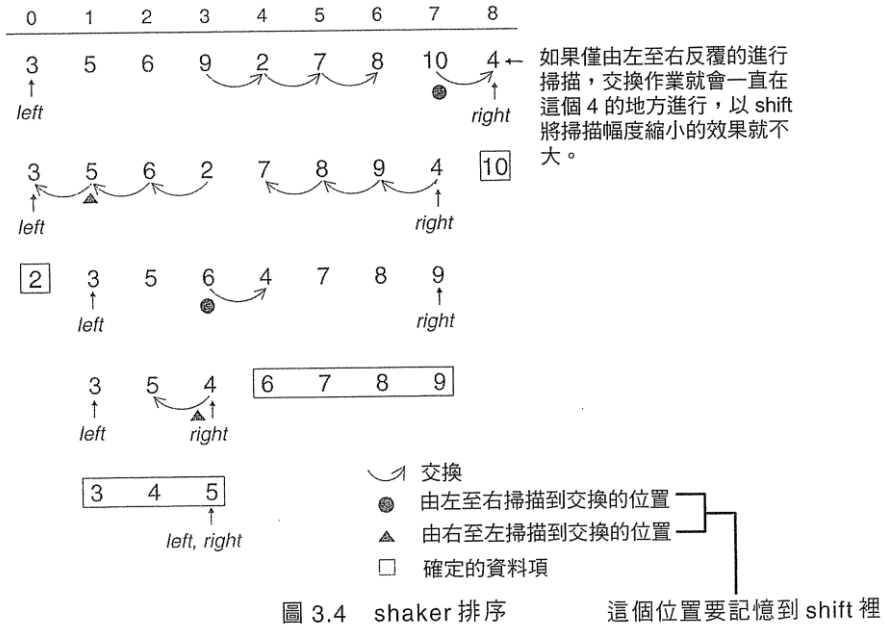


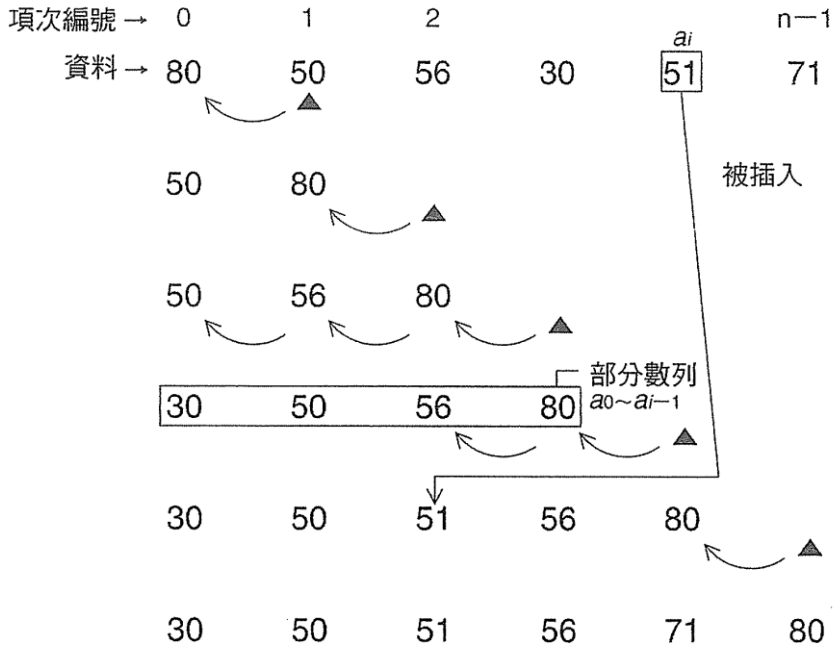
圖 3.4 shaker 排序

Please modify bubble sort to implement shaker sort algorithm and test your program.

3. Insertion Sort

The concept of basic insertion is to suppose the elements $a_0 \sim a_{i-1}$ **(**$i \le n-1$**) are ordered series, and to search which location the target element should be inserted. To decide which location the target element** a_i **should be inserted, start comparison from** a_{i-1} **with repeating exchange operation and stop if the element is smaller than** a_{i-1}**.**

Take following figure as example, to decide which location 51 should be placed, firstly compare 51 with 80, and exchange 80 and 51. And then compere 51 with 56, so we have to exchange 56 and 51. Next step is to compere 51 with 50, and we will find that 51 is bigger than 50, so 51 should be placed at this location, and this step is end.



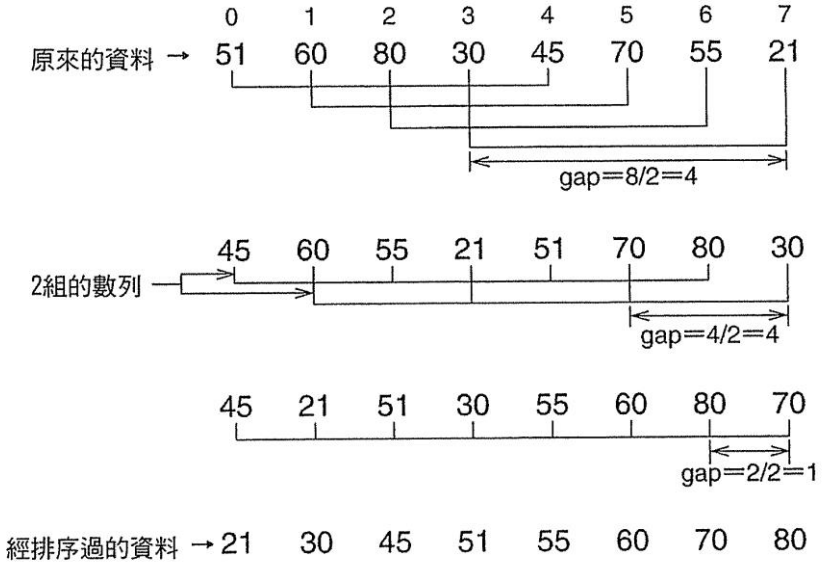
4. Shell Sort

Shell sort, developed by Donald L. Shell, is a non-stable in-place sort. Shell sort improves on the efficiency of insertion sort by quickly shifting values to their destination. Average sort time is $O(n^{1.25})$, while worst-case time is $O(n^{1.5})$.

The shell sort algorithm divides all numbers into several series by fixed gap and do shell sort in each part. It is doing sorting as well as being convergent to whole series from part series, and the sorting operation ends when gap is set as 1. In other words, to decrease counts of comparison and exchange, smaller elements are roughly sorted in the front of series and larger elements are roughly sorted in the back of series before gap becomes 1.

There is method to decide the gap value, but we just use the gap with half part of original series. Take following figure as example, suppose there are 8 data, and we set the initial gap as 8/2=4. Furthermore, sequentially apply shell sort on four divisions of series, $(51,45),(60,70),\cdots$. Next step is to set gap as 4/2=2, and apply shell sort on two divisions of series, $(45,55,51,80),(60,21,70,30)$. Finally,

set gap as 2/2=1, and apply shell sort on the only one division series, $(45,21,\cdots)$**. Because gap is 1, the series includes all numbers, the sort operation ends.**

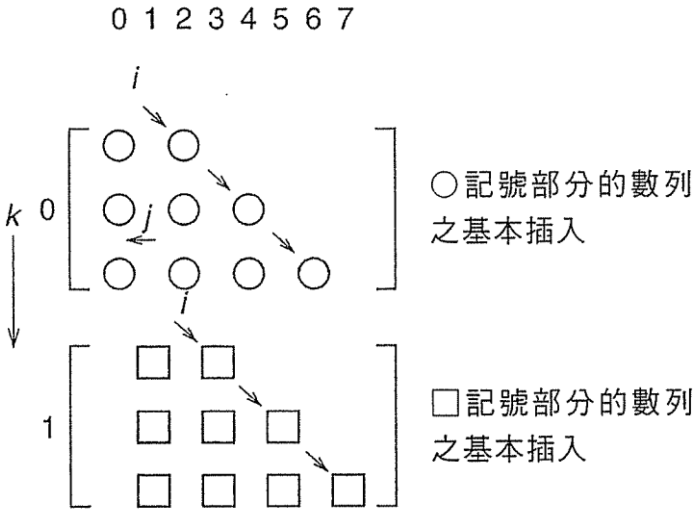


In above figure, when gap is 2, the numbers are divided into two divisions.

0 1 2 3 4 5 6 7

○□○□○□○□

Divide these numbers into two series, ○○○○ and □□□□, and apply basic insertion algorithm on them separately, the result is as follows.



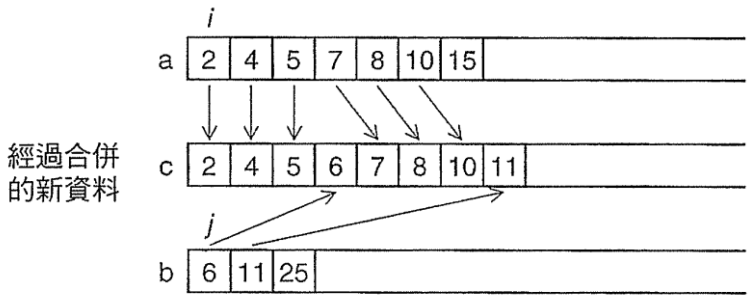
The overall algorithm is as follows.

a) Set initial gap value as N/2.

b) Repeat following steps until gap is 1. Apply basic insertion on each "gap" numbers $(a_j, a_{j+gap}, a_{j+2gap}, \cdots)$**. (There are "gap" numbers in each division, so the repeating counts is smaller than "gap".)**

5. Merging

Merge is to combine two or more sorted data series into one sorted data series.



Suppose a and b are two sorted data series, and c are new data storage which we want to put new combined data here. Refer to above figure, and the merging algorithm is as follows.

a) Repeat following step until end of a or b. Compare a_i or b_j, copy the smaller one to c_p, and then move number i or j of smaller data to next location.

b) Copy all data of a or b between current location and end location to c.

Please implement above merge algorithm and verify your program.

6. Pattern Matching

Suppose there is a text shown as follows, and the key is the string which we want to find in text. Please implement a function char *search(char *text, char *key), which can return the address of found string. If it finds no target, it will return NULL. In other words, the function of "search" is the same as C-String function "strstr()". You just can use C-String function "strlen()" and "strncmp()" to complete this program.