अध्याय 3

सॉर्टिंग(Sorting)

सॉर्टिंग (Sorting) एक विशेष स्वरूप में डेटा को व्यवस्थित करने को संदर्भित करता है। सॉर्टिंग एल्गोरिथ्म डेटा को एक विशेष क्रम में व्यवस्थित करने का तरीका निर्दिष्ट करती है। सबसे आम क्रम संख्यात्मक या वर्णानुक्रम हैं। यदि डेटा एक क्रमबद्ध तरीके से संग्रहित किया गया है तो सॉर्टिंग का सर्वाधिक महत्व डाटा सर्च को आसान बनाने में है। सॉर्टिंग डेटा को ओर अधिक पठनीय प्रारूप में प्रदर्शित करने के लिए भी प्रयोग कि जाती है। वास्तविक जीवन में सॉर्टिंग के कुछ उदाहरण हैं:

टेलीफोन निर्देशिका–

टेलीफोन निर्देशिका, लोगों के टेलीफोन नंबरो को उनके नाम के अनुसार क्रमबद्ध करके संग्रहीत करती है ताकी नामों को आसानी से सर्च किया जा सकता है।

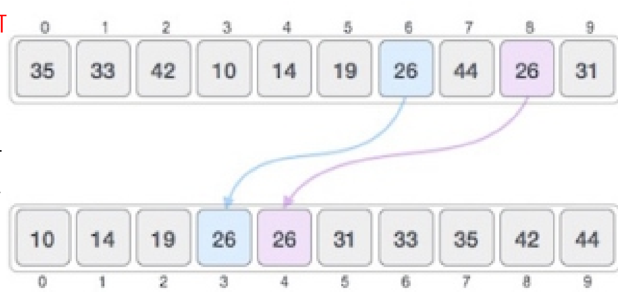
शब्दकोश– शब्दकोश मे शब्द वर्णमाला के क्रम मे संग्रहीत किये जाते है इसलिए किसी भी शब्द को सर्च करना आसान हो जाता है।

इन–प्लेस सॉर्टिंग और नाट–इन–प्लेस सॉर्टिंग(In-place Sorting and Not-in-place Sorting):

सॉर्टिंग एल्गोरिथ्म को तुलना और कुछ डेटा तत्वों के अस्थायी भंडारण के लिए कुछ अतिरिक्त स्थान की आवश्यकता हो सकती है। इन–प्लेस सॉर्टिंग एल्गोरिथ्म को किसी भी अतिरिक्त जगह की आवश्यकता नहीं होती है और इसलिए इन्हे सॉर्टिंग इन–प्लेस कहा जाता है, उदाहरण के लिए, ऐरे के भीतर ही सॉर्टिंग। बबल सॉर्ट इन–प्लेस सॉर्टिंग का एक उदाहरण है।

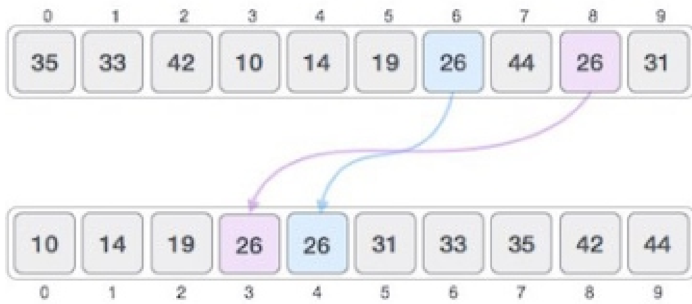
हालांकि, कुछ सॉर्टिंग एल्गोरिथ्म में, प्रोग्राम को स्पेस की आवश्यकता है जोकि तत्वो, जिन्हे सॉर्ट करना है के बराबर या उनसे अधिक हो सकती है और इसलिए इन्हे नाट–इन–प्लेस सॉर्टिंग कहा जाता है। मर्ज–सॉर्ट नाट–इन–प्लेस सॉर्टिंग का एक उदाहरण है।

स्टेबल और अनस्टेबल सॉर्टिंग (Stable and Unstable Sorting):



सॉर्टिंग एल्गोरिथ्म, तत्वो को सॉर्ट करने के बाद, एक जैसे तत्वो के क्रम जिसमे वो प्रकट होते है को परिवर्तित नहीं करती है उनको स्टेबल सॉर्टिंग कहा जाता है।

सॉर्टिंग एल्गोरिथ्म, तत्वो को सॉर्ट करने के बाद, एक जैसे तत्वो के क्रम जिसमे वो प्रकट होते है को परिवर्तित करती है उनको अनस्टेबल सॉर्टिंग कहा जाता है।



एक एल्गोरिथ्म की स्टेब्लिटी (Stability) मायने रखती है जब हम मूल तत्वों का क्रम बनाए रखना चाहते हैं उदाहरण के लिए एक टपल में।

अडप्टिव और नॉन–अडप्टिव सॉर्टिंग एल्गोरिथ्म(Adaptive and Non adaptive Sorting) :

यदि सॉर्टिंग एल्गोरिथ्म, सॉर्ट करने वाली लिस्ट में पहले से ही सॉर्टेड तत्वों का लाभ लेती है तब उसे अडप्टिव कहा जाता है। अर्थात सॉर्टिंग के दौरान यदि स्रोत (source) सूची में पहले से ही कुछ तत्व सॉर्टेड है तब अडप्टिव एल्गोरिथ्म इसे ध्यान में रखते हुए उनका क्रम पुनः नहीं बदलती।

एक नॉन–अडप्टिव सॉर्टिंग एल्गोरिथ्म सूची में पहले से ही सॉर्टेड तत्वो को ध्यान में नही रखती। वे तत्व सॉर्टेड है या नही की पुष्टि करने के लिए हर एक तत्व के क्रम को बदलती हैं।

महत्वपूर्ण शर्तें

सॉर्टिंग तकनीको पर चर्चा के दौरान आम तौर कुछ शब्दावली का प्रयोग किया जाता है, यहाँ उनका एक संक्षिप्त परिचय हैं:

बढता क्रम (Increasing Order):

मानों का एक अनुक्रम बढते हुए क्रम में कहा जाता है, यदि बाद का तत्व अपने पिछले वाले तत्व से अधिक है। उदाहरण के लिए, 1, 3, 4, 6, 8, 9, बढते क्रम में हैं, क्योंकि यहा हर अगला तत्व अपने पिछले वाले तत्व से अधिक है।

घटता क्रम (Decreasing Order):

मानों का एक अनुक्रम घटते हुए क्रम में कहा जाता है, यदि बाद का तत्व अपने पिछले वाले तत्व से कम है। उदाहरण के लिए, 9, 8, 6, 4, 3, 1, घटते क्रम में हैं क्योंकि यहा हर अगला तत्व अपने पिछले तत्व से कम है।

गैर–बढ़ता क्रम (Non- Increasing Order):

मानों का एक अनुक्रम गैर–बढ़ते हुए क्रम में कहा जाता है, यदि बाद का तत्व अपने पिछले वाले तत्व से कम या उसके बराबर है। यह क्रम तब होता है जब अनुक्रम में डुप्लिकेट मान हो। उदाहरण के लिए, 9, 8, 6, 3, 3, 1, गैर बढ़ते क्रम में हैं क्यों यहा हर अगला तत्व अपने पिछले तत्व से कम या उसके बराबर (3 के मामले में) है।

गैर–घटता क्रम (Non-Decreasing Order):

मानों का एक अनुक्रम गैर–घटते हुए क्रम में कहा जाता है, यदि बाद का तत्व अपने पिछले वाले तत्व से अधिक या उसके बराबर है। यह क्रम तब होता है जब अनुक्रम में डुप्लिकेट मान हो। उदाहरण के लिए, 1, 3, 3, 6, 8, 9, गैर–घटते क्रम में हैं क्यों यहा हर अगला तत्व अपने पिछले तत्व से अधिक या उसके बराबर (3 के मामले में) है।

बबल (Bubble) सॉर्ट:

बबल सॉर्ट एक साधारण सॉर्टिंग एल्गोरिथ्म है। यह सॉर्टिंग एल्गोरिथ्म, तुलना–आधारित एल्गोरिथ्म है जिसमें सन्निकट तत्वों के प्रत्येक जोड़े की तुलना की जाती है और अगर वे क्रम में नहीं है तब तत्वों को बदला जाता हैं। इस एल्गोरिथ्म कि औसत (average) और सबसे खराब (worst case) स्थिति मे कॉम्प्लेक्सिटी $O(n^2)$ है जहाँ n सॉर्ट किये जाने वाले तत्वों की संख्या हैं और इसलिए यह एल्गोरिथ्म बड़े डेटा सेट के लिए उपयुक्त नहीं है।

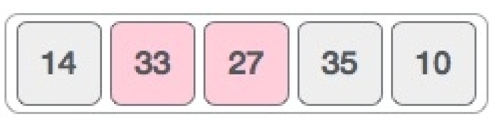
बबल सॉर्टिंग कैसे काम करती है?

हम उदाहरण के लिए एक अनसोर्टेड ऐरे ले रहे है। बबल सॉर्ट $O(n^2)$ समय लेती है, इसलिए हम इसे छोटा और सटीक रख रहे हैं।

बबल सॉर्ट, सबसे पहले दो तत्वों के साथ शुरू होती है, कोनसा बडा है यह जाँच करने के लिए उनकी तुलना करती है।

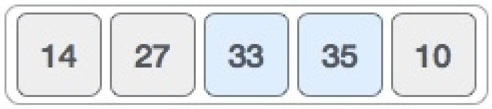
इस मामले में, 33 मान 14 से अधिक है, इसलिए यह पहले से सोर्टेड है। आगे हम 27 से 33 की तुलना करते है।

हम पाते हैं कि 27, 33 से छोटा है और इन दोनो मानो को बदली किया जाना चाहिए।



नई ऐरे इस तरह दिखनी चाहिए –

आगे हम 33 और 35 की तुलना मे पाते हैं कि दोनों पहले से ही सोर्टिड स्थितियों में हैं।



फिर हम अगले दो मानों, 35 और 10 को देखते है।

हम जानते हैं कि 10, 35 से छोटा है इसलिए वे सोर्टेड नहीं हैं।

हम इन मानों को स्वैप करते है। हम पाते हैं कि हम ऐरे के अंत तक पहुँच चुके हैं। एक पुनरावृत्ति (iteration) के बाद, ऐरे इस तरह दिखना चाहिए –

अब हम दिखा रहे हैं की एक ऐरे प्रत्येक पुनरावृत्ति के बाद किस तरह दिखना चाहिए। दूसरी पुनरावृत्ति के बाद, यह इस तरह दिखना चाहिए –

ध्यान दें कि प्रत्येक पुनरावृत्ति के बाद, ऐरे के अंत में कम से कम एक मान चलता जाता है।

और जब किसी स्वैप की आवश्यकता नहीं रहती तब बबल सॉर्ट यह जान जाता है कि ऐरे पूरी तरह से सॉर्ट हो गया है।

Algorithm:

हम यहा यह मान रहे कि तत्वों की लिस्ट एक ऐरे मे है और स्वैप फंक्शन ऐरे तत्वों को स्वैप करता है।

```
BubbleSort
for all elements of list
if list[i] > list[i+1]
swap(list[i], list[i+1])
end if
end for

return list

end BubbleSort
```

बबल सॉट के लिए C प्रोग्राम:

```
#include <stdio.h>
#include <stdbool.h>

#define MAX 10

int list[MAX] = {1,8,4,6,0,3,5,2,7,9};

void display() {
int i;
printf("[");
// navigate through all items
for(i = 0; i < MAX; i++) {
printf("%d ",list[i]);
}
```

```
printf("]\n");
}

void bubbleSort() {
int temp;
int i,j;

bool swapped = false;

// loop through all numbers
for(i = 0; i < MAX-1; i++) {
swapped = false;

// loop through numbers falling ahead
for(j = 0; j < MAX-1-i; j++) {
printf("     Items compared: [ %d, %d ] ", list[j],list[j+1]);

// check if next number is lesser than current no
//   swap the numbers.
//  (Bubble up the highest number)

if(list[j] > list[j+1]) {
temp = list[j];
list[j] = list[j+1];
list[j+1] = temp;

swapped = true;
printf(" => swapped [%d, %d]\n",list[j],list[j+1]);
}else {
printf(" => not swapped\n");
}
```

```
}

// if no number was swapped that means
//   array is sorted now, break the loop.
if(!swapped) {
break;
}

printf("Iteration %d#: ",(i+1));
display();
}

}

main() {
printf("Input Array: ");
display();
printf("\n");

bubbleSort();
printf("\nOutput Array: ");
display();
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

Output

Input Array: [1 8 4 6 0 3 5 2 7 9]

Items compared: [1, 8] => not swapped

Items compared: [8, 4] => swapped [4, 8]

Items compared: [8, 6] => swapped [6, 8]

Items compared: [8, 0] => swapped [0, 8]

Items compared: [8, 3] => swapped [3, 8]

Items compared: [8, 5] => swapped [5, 8]

Items compared: [8, 2] => swapped [2, 8]

Items compared: [8, 7] => swapped [7, 8]

Items compared: [8, 9] => not swapped

Iteration 1#: [1 4 6 0 3 5 2 7 8 9]

Items compared: [1, 4] => not swapped

Items compared: [4, 6] => not swapped

Items compared: [6, 0] => swapped [0, 6]

Items compared: [6, 3] => swapped [3, 6]

Items compared: [6, 5] => swapped [5, 6]

Items compared: [6, 2] => swapped [2, 6]

Items compared: [6, 7] => not swapped

Items compared: [7, 8] => not swapped

Iteration 2#: [1 4 0 3 5 2 6 7 8 9]

Items compared: [1, 4] => not swapped

Items compared: [4, 0] => swapped [0, 4]

Items compared: [4, 3] => swapped [3, 4]

Items compared: [4, 5] => not swapped

Items compared: [5, 2] => swapped [2, 5]

Items compared: [5, 6] => not swapped

Items compared: [6, 7] => not swapped

Iteration 3#: [1 0 3 4 2 5 6 7 8 9]

Items compared: [1, 0] => swapped [0, 1]

Items compared: [1, 3] => not swapped

Items compared: [3, 4] => not swapped

Items compared: [4, 2] => swapped [2, 4]

Items compared: [4, 5] => not swapped

Items compared: [5, 6] => not swapped

Iteration 4#: [0 1 3 2 4 5 6 7 8 9]

Items compared: [0, 1] => not swapped

Items compared: [1, 3] => not swapped

Items compared: [3, 2] => swapped [2, 3]

Items compared: [3, 4] => not swapped

Items compared: [4, 5] => not swapped

Iteration 5#: [0 1 2 3 4 5 6 7 8 9]

Items compared: [0, 1] => not swapped

Items compared: [1, 2] => not swapped

Items compared: [2, 3] => not swapped

Items compared: [3, 4] => not swapped

Output Array: [0 1 2 3 4 5 6 7 8 9]

चयन (Selection) सॉर्टिंग:चयन सॉर्ट एक सरल सॉर्टिंग एल्गोरिथ्म है।

यह एक इन–प्लेस तुलना–आधारित सॉर्टिंग एल्गोरिथ्म है इसमें सूची दो भागों में विभाजित होती है, सॉर्ट किया हुआ भाग बाईं ओर तथा सॉर्ट न किया हुआ भाग दायीं ओर रहता है। शुरू में, सॉर्ट किया गया हुआ भाग खाली रहता है और संपूर्ण सूची सॉर्ट न किये हुआ भाग में होती है। अवर्गीकृत (अनसोरटेड) ऐरे में से सबसे छोटा तत्व का चयन किया जाता है और इसे ऐरे में सबसे बाएँ तत्व के साथ बदली किया जाता है, और वह तत्व सॉर्ट किये हुआ ऐरे का एक हिस्सा बन जाता है। यह प्रक्रिया अवर्गीकृत ऐरे की सीमा को एक तत्व दायीं ओर बढाती चली जाती है। इस एल्गोरिथ्म कि औसत (average) और सबसे खराब (worst case) स्थिति मे कॉम्प्लेक्सिटी $O(n^2)$ है, जहाँ n सॉर्ट किये जाने वाले तत्वों की संख्या हैं और इसलिए यह एल्गोरिथ्म बड़े डेटा सेट के लिए उपयुक्त नहीं है।

चयन सॉर्टिंग कैसे काम करती है?

एक उदाहरण के रूप में निम्नलिखित दर्शाया हुए ऐरे पर विचार करें:

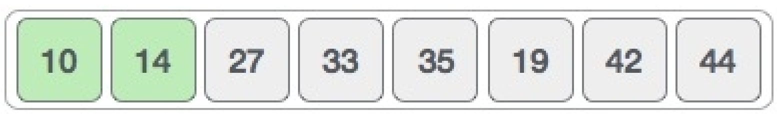
क्रमबद्ध लिस्ट में प्रथम स्थान के लिए, पूरी लिस्ट की क्रमिक रूप से जांच होती है। पहली स्थिति जहां 14 को वर्तमान में संग्रहित करना है, हम पूरी लिस्ट को सर्च करते है और पाते हैं कि 10 निम्नतम मान है।

इसलिए हम 14 को 10 से बदलते है। एक पुनरावृत्ति के बाद 10 जो कि लिस्ट में न्यूनतम मान है, सॉर्टेड लिस्ट में पहली स्थिति में दिखाई देता है।

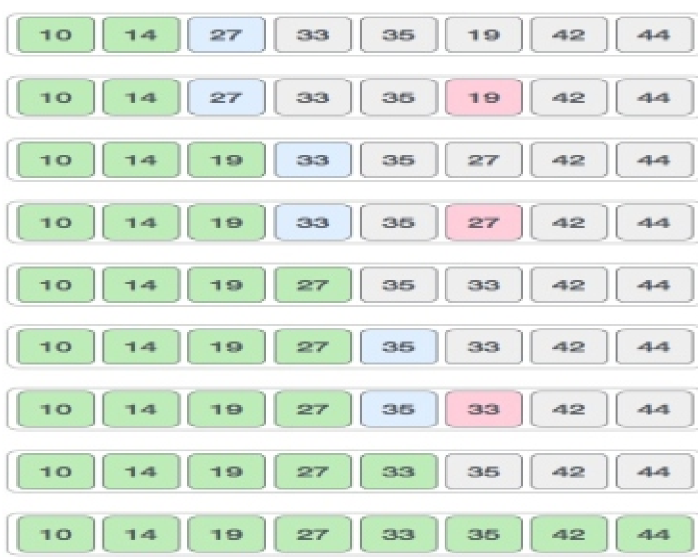
दूसरे स्थान के लिए जहां 33 है, हम एक रेखीय ढंग से बाकी लिस्ट की स्कैनिंग शुरू करते हैं।

हम पाते हैं कि 14 लिस्ट में दूसरा सबसे कम मान है और यह दूसरे स्थान पर होना चाहिए। हम इन मानों को स्वैप करते हैं।

दो पुनरावृत्तियों के बाद, दो कम से कम मान एक क्रमबद्ध ढंग से शुरुआत में आ जाते हैं।



यही प्रक्रिया ऐरे में बाकी के आइटम के लिए लागू कि जाती है। पूरी सॉर्टिंग प्रक्रिया का एक सचित्र चित्रण निम्नानुसार है:



एल्गोरिथ्म:

Step 1 – Set MIN to location 0

Step 2 – Search the minimum element in the list

Step 3 – Swap with value at location MIN

Step 4 – Increment MIN to point to next element

Step 5 – Repeat until list is sorted

चयन सॉट के लिए C प्रोग्राम:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 7
int intArray[MAX] = {4,6,3,2,1,9,7};
void printline(int count) {
int i;
for(i = 0;i <count-1;i++) {
printf("=");
}
printf("=\n");
}
void display() {
int i;
printf("[");
// navigate through all items
for(i = 0;i<MAX;i++) {
printf("%d ", intArray[i]);
}
printf("]\n");
}
void selectionSort() {
int indexMin,i,j;
// loop through all numbers
```

```
for(i = 0; i < MAX-1; i++) {
// set current element as minimum
indexMin = i;
// check the element to be minimum
for(j = i+1;j<MAX;j++) {
if(intArray[j] < intArray[indexMin]) {
indexMin = j;
}
}
if(indexMin != i) {
printf("Items swapped: [ %d, %d ]\n" , intArray[i], intArray[indexMin]);
// swap the numbers
int temp = intArray[indexMin];
intArray[indexMin] = intArray[i];
intArray[i] = temp;
}
printf("Iteration %d#:",(i+1));
display();
}
}
main() {
printf("Input Array: ");
display();
printline(50);
selectionSort();
printf("Output Array: ");
display();
printline(50);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

Output

Input Array: [4 6 3 2 1 9 7]

===

Items swapped: [4, 1]

Iteration 1#:[1 6 3 2 4 9 7]

Items swapped: [6, 2]

Iteration 2#:[1 2 3 6 4 9 7]

Iteration 3#:[1 2 3 6 4 9 7]

Items swapped: [6, 4]

Iteration 4#:[1 2 3 4 6 9 7]

Iteration 5#:[1 2 3 4 6 9 7]

Items swapped: [9, 7]

Iteration 6#:[1 2 3 4 6 7 9]

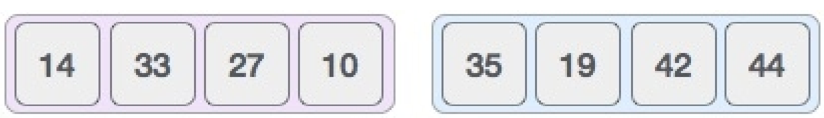
Output Array: [1 2 3 4 6 7 9]

मर्ज (Merge) सॉर्टिंग: मर्ज सॉर्ट डिवाइड (विभजित) एंड कॉन्कर (जीत) पर आधारित एक सॉर्टिंग तकनीक है। इसकी सबसे खराब मामले में (worst-case) कॉम्प्लेक्सिटी O(nlogn) होने के कारण यह सबसे अच्छी एल्गोरिथ्म में से एक है। मर्ज सॉर्ट पहले ऐरे को दो बराबर हिस्सों में तोडती है और फिर उन्हें एक क्रमबद्ध ढंग से जोड़ती है।

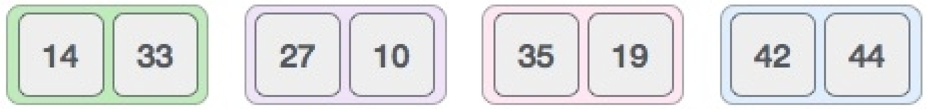
मर्ज सॉर्ट कैसे काम करती है?

मर्ज सॉर्ट को समझने के लिए हम एक निम्नलिखित अवर्गीकृत ऐरे लेते है

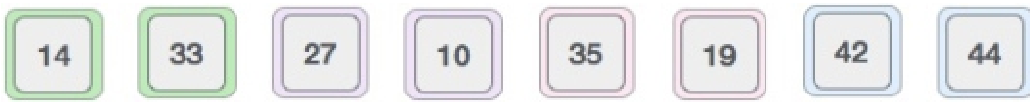
हम जानते हैं कि मर्ज सॉर्ट पहले पूरी ऐरे को पुनरावृत्तीय तरीके से बराबर हिस्सों में बांटती है जब तक कि परमाणु (atomic)या अविभाज्य मान प्राप्त नहीं हो जाते हैं। हम यहाँ देखते है कि 8 मानो की एक ऐरे 4 आकार की दो ऐरे में बंट गयी है।



यह मूल मानों की उपस्थिति के अनुक्रम को नहीं बदलता है। अब हम इन दो ऐरे को हिस्सों में विभाजित करते है।

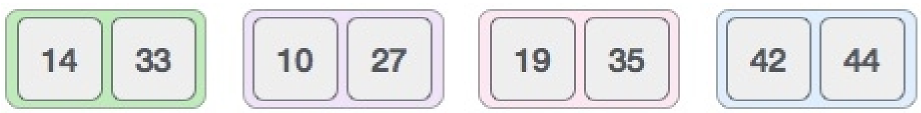


हम आगे इन ऐरे को ओर विभाजित करते है और हमे परमाणु मान प्राप्त होते है जिनको ओर अधिक विभाजित नही किया जा सकता।

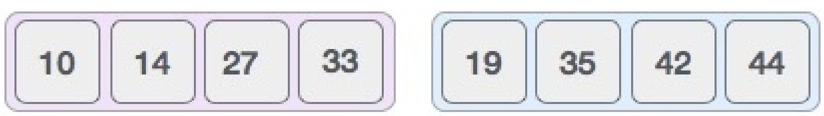


अब, हम उन्हें ठीक उसी तरीके से सम्मलित करते है जैसे उन्हें तोड़ा था। कृपया इन सूचियों को दिए गए रंग कोड पर ध्यान दें।

हम पहले प्रत्येक लिस्ट के तत्व की तुलना करते है और फिर एक क्रमबद्ध ढंग से उन्हें एक दूसरी लिस्ट में सम्मलित करते है। हम जानते हैं कि 14 और 33 सॉर्टेड स्थिति में ही हैं। हम 27 और 10 की तुलना करते है और 2 मानों की लक्ष्य लिस्ट में हम पहले 10 को डालते है और उसके पीछे 27 को। हम 19 और 35 का क्रम बदलते है जबकि 42 और 44 को क्रमिक रूप से रखा जाता है।



संयोजन चरण के अगले पुनरावृत्ति में, हम दो डेटा मानों की लिस्ट की तुलना करते है, और उन्हें एक सॉर्टेड क्रम में डेटा मानों की लिस्ट में मर्ज केर देते है।



अंतिम विलय के बाद, लिस्ट इस तरह दिखेगी –

Algorithm:

मर्ज सॉर्ट लिस्ट को पुनरावृत्तीय तरीके से बराबर हिस्सों में बांटती है जब तक कि उसे ओर अधिक विभाजित नही किया जा सकता। परिभाषा के अनुसार, अगर लिस्ट में केवल एक ही तत्व है, तो यह लिस्ट सॉर्टेड है। फिर, मर्ज सॉर्ट छोटी सॉर्टेड सूचियों को इस तरह सम्मलित करती है ताकि नयी बनने वाली सूचि भी सॉर्टेड ही रहे।

Step 1 – अगर लिस्ट में केवल एक ही तत्व है, तो यह लिस्ट सॉर्टेड है

Step 2 – लस्ट को पुनरावृत्तीय तरीके से दो बराबर हिस्सों में बांटना जब तक कि उसे ओर अधिक विभाजित नही किया जा सकता।

Step 3 – छोटी सॉर्टेड सूचियों को इस तरह सम्मलित करना ताकि नयी बनने वाली सूचि भी सॉर्टेड ही रहे।

मर्ज सॉर्ट के लिए C प्रोग्राम:

```
#include <stdio.h>
#define max 10
int a[10] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44 };
int b[10];
void merging(int low, int mid, int high) {
int l1, l2, i;
for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
if(a[l1] <= a[l2])
b[i] = a[l1++];
else
b[i] = a[l2++];
}
while(l1 <= mid)
b[i++] = a[l1++];

while(l2 <= high)
b[i++] = a[l2++];

for(i = low; i <= high; i++)
a[i] = b[i];
}
void sort(int low, int high) {
int mid;
if(low < high) {
mid = (low + high) / 2;
sort(low, mid);
sort(mid+1, high);
merging(low, mid, high);
} else {
return;
```

```
}
}
int main() {
int i;
printf("List before sorting\n");
for(i = 0; i <= max; i++)
printf("%d ", a[i]);
sort(0, max);
printf("\nList after sorting\n");
for(i = 0; i <= max; i++)
printf("%d ", a[i]);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
List before sorting
10 14 19 26 27 31 33 35 42 44 0
List after sorting
0 10 14 19 26 27 31 33 35 42 44
```

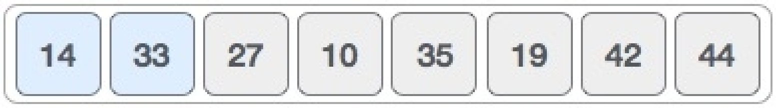
निवेशन (Insertion) सॉर्टिंग: यह एक इन–प्लेस तुलना–आधारित सॉर्टिंग एल्गोरिथ्म है। इसमें एक उप–लिस्ट बनाये रखी जाती है जो हमेशा सॉर्टेड रहती है। उदाहरण के लिए, एक ऐरे के निचले हिस्से को सॉर्टेड बनाए रखा जाता है। एक तत्व जिसे इस सॉर्टेड उप–लिस्ट में सम्मलित किया जाना है, इसकी उचित जगह सर्च करके फिर इसे वहाँ डाला जाता है। इसलिए इसका नाम, निवेशन सॉर्ट है।

ऐरे को क्रमिक रूप से सर्च किया जाता है और अवर्गीकृत आइटम को स्थानांतरित किया जाता हैं और उन्हें सॉर्टेड उप–लिस्ट में डाला (एक ही ऐरे में) जाता है। इस एल्गोरिथ्म कि औसत (average) और सबसे खराब (worst case) स्थिति मे कॉम्प्लेक्सिटी $O(n^2)$ है, जहाँ n सॉर्ट किये जाने वाले तत्वों की संख्या हैं और इसलिए यह एल्गोरिथ्म बड़े डेटा सेट के लिए उपयुक्त नहीं है।

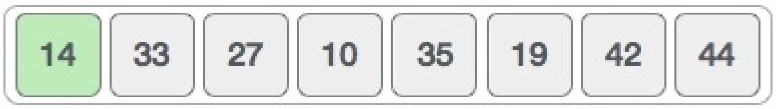
निवेशन सॉर्टिंग कैसे काम करती है?

हम उदाहरण के लिए एक अवर्गीकृत ऐरे ले रहे है।

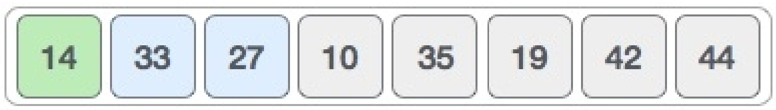
निवेशन सॉर्टिंग पहले दो तत्वों की तुलना करती है।



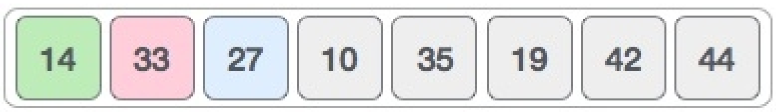
यहा 14 और 33 दोनों ही आरोही क्रम में पहले से ही हैं। अभी के लिए, 14 सॉर्टेड उप–लिस्ट में है।



निवेशन सॉर्टिंग आगे 33 की 27 से तुलना करती है।

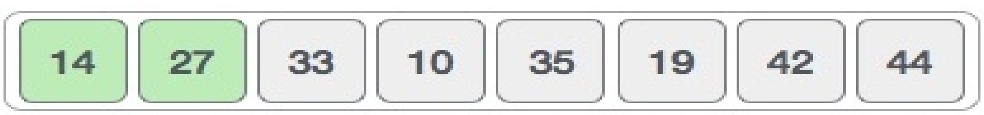


और पाती है कि 33 सही स्थिति में नहीं है।

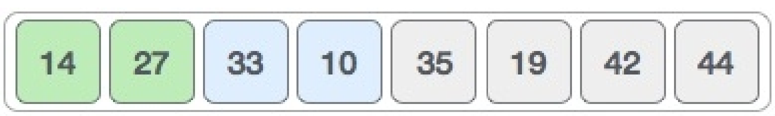


यह 33 को 27 के साथ स्वैप करती है।

यह सॉर्टेड उप–लिस्ट के सभी तत्वों के साथ की जाँच करती है। यहाँ हम देखते सॉर्टेड उप–लिस्ट में केवल एक तत्व 14 है और 27, 14 से अधिक है, इसलिए सॉर्टेड उप–लिस्ट की अदला–बदली के बाद भी यह सॉर्टेड रहेगी।



अब तक सॉर्टेड उप–लिस्ट में 14 और 27 है। इसके बाद, यह 33 की 10 से तुलना करती है।



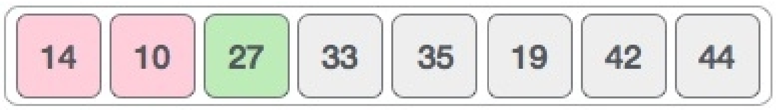
यह मान एक सॉर्ट क्रम में नहीं हैं।

इसलिए हम उन्हें स्वैपकरते है।

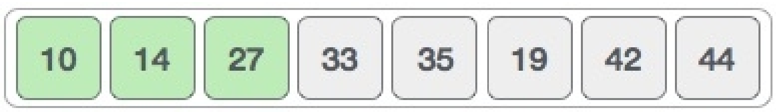
हालांकि, स्वैपिंग 27 और 10 को अवर्गीकृत बनाता है।

इसलिए, हम उन्हें भी स्वैप करते है।

फिर हम 14 और 10 को अवर्गीकृत क्रम में पाते हैं।



हम उन्हें फिर से स्वैप करते है। तीसरी पुनरावृत्ति के अंत तक सॉर्टेड उप लिस्ट मे 4 मान हो जाते है।



इस प्रक्रिया तब तक जारी रहती है जब तक सभी अवर्गीकृत मान सॉर्टेड उप–लिस्ट में शामिल नहीं हो जाते है।

Algorithm:

अब हम यह सॉर्टिंग तकनीक कैसे काम करती है कि एक बड़ी तस्वीर जानते है, इसलिए हम निवेशन (Insertion) सॉर्टिंग के सरल स्टेप्स प्राप्त कर सकते हैं।

चरण 1 –अगर लिस्ट में केवल एक ही तत्व है, तो यह लिस्ट सॉर्टेड है।

चरण 2– अगला तत्व लेवे।

चरण 3–सॉर्टेड उप सूची के सभी तत्वों के साथ की तुलना करें।

चरण 4 – सॉर्टेड उप सूची के उन सभी तत्वों को शिफ्ट करे जो सॉर्ट किये जाने वाले मान से अधिक है।

चरण 5– मान सम्मिलित करें।

चरण 6– दोहराएँ जब तक सूची सॉर्ट नही हो जाता है।

निवेशन (Insertion)सॉर्टिंग के लिए C प्रोग्राम:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 7
int intArray[MAX] = {4,6,3,2,1,9,7};
```

```
void printline(int count) {
int i;
for(i = 0;i <count-1;i++) {
printf("=");
}
printf("=\n");
}
void display() {
int i;
printf("[");
// navigate through all items
for(i = 0;i<MAX;i++) {
printf("%d ",intArray[i]);
}
printf("]\n");
}
void insertionSort() {
int valueToInsert;
int holePosition;
int i;
// loop through all numbers
for(i = 1; i < MAX; i++) {
// select a value to be inserted.
valueToInsert = intArray[i];
// select the hole position where number is to be inserted
holePosition = i;
// check if previous no. is larger than value to be inserted
while (holePosition > 0 && intArray[holePosition-1] > valueToInsert) {
intArray[holePosition] = intArray[holePosition-1];
holePosition--;
printf(" item moved : %d\n" , intArray[holePosition]);
}
if(holePosition != i) {
printf(" item inserted : %d, at position : %d\n" ,
valueToInsert,holePosition);
// insert the number at hole position
intArray[holePosition] = valueToInsert;
}
printf("Iteration %d#:",i);
```

```
display();
}
}
main() {
printf("Input Array: ");
display();
printline(50);
insertionSort();
printf("Output Array: ");
display();
printline(50);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
Input Array: [4 6 3 2 1 9 7 ]
==================================================
Iteration 1#:[4 6 3 2 1 9 7 ]
item moved : 6
item moved : 4
item inserted : 3, at position : 0
Iteration 2#:[3 4 6 2 1 9 7 ]
item moved : 6
item moved : 4
item moved : 3
item inserted : 2, at position : 0
Iteration 3#:[2 3 4 6 1 9 7 ]
item moved : 6
item moved : 4
item moved : 3
item moved : 2
item inserted : 1, at position : 0
Iteration 4#:[1 2 3 4 6 9 7 ]
Iteration 5#:[1 2 3 4 6 9 7 ]
```

item moved : 9

item inserted : 7, at position : 5

Iteration 6#:[1 2 3 4 6 7 9]

Output Array: [1 2 3 4 6 7 9]

त्वरित (Quick) सॉर्टिंग: त्वरित एक प्रकार की अत्यंत कुशल सॉर्टिंग एल्गोरिथ्म है और डेटा के ऐरे को छोटे ऐरे में विभाजन करने पर आधारित है। एक बड़ा ऐरे दो ऐरे में विभाजित किया जाता है, जिनमें से एक ऐरे में निर्धारित मान (जिसके आधार पर विभाजन किया जाता है और इसे पाइवोट कहते हैं) की तुलना में छोटे मान रखता है, और दूसरे ऐरे में पाइवोट मान से अधिक मानों को रख जाता है।

त्वरित सॉर्टिंग एक ऐरे को विभजित करती है और उसके बाद दो परिणामस्वरूप सब–ऐरे को सॉर्ट करने के लिए खुद को बारी बारी से दो बार कॉल करती है। यह एल्गोरिथम बड़े आकार के डेटा सेट के लिए काफी कुशल है।

इस एल्गोरिथ्म कि औसत (average) और सबसे खराब (worst case) स्थिति मे कॉम्प्लेक्सिटी O(nlogn) है, जहाँ n सॉर्ट किये जाने वाले तत्वों की संख्या हैं।

त्वरित सॉर्टिंग में विभाजन:

निम्नलिखित उदाहरण यह बताता हैं कि कैसे एक ऐरे में पाइवोट मान को सर्च किया जाता है।

पाइवोट मान लिस्ट को दो भागों में बटता हैं। और बारी बारी से, हम प्रत्येक उप–सूचियों के लिए पाइवोट मान का पता लगाते है जब तक की सभी सूचियों में केवल एक ही तत्व नहीं रह जाता ।

A[6]	A[7]	A[8]	A[9]	A[10]	A[11]	A[12]
98	84	65	108	60	96	72
72	84	65	108	60	96	98
72	84	65	98	60	96	108
72	84	65	96	60	98	108

पाइवोट एल्गोरिथमः

चरण 1 –सबसे अधिक सूचकांक मान को पाइवोट चुने।

चरण 2 – धुरी को छोड़कर सूची के दाईं तथा बायीं ओर इंगित करने के लिए दो वेरिएबल ले

चरण 3 – बांया वेरिएबल कम सूचकांक को इंगित करता है

चरण 4 – जबकि दांया वेरिएबल उच्च सूचकांक को इंगित करता है

चरण 5 – जब तक बांये वेरिएबल की वैल्यू पाइवोट से कम है दांये चले

चरण 6 – जब तक दांये वेरिएबल की वैल्यू पाइवोट से अधिक है बांये चले

चरण 7 – यदि दोनों चरण 5 और चरण 6 मैच नहीं करते तो बांये और दांये को स्वैप करे

चरण 8 – यदि बांया ≥दांया पॉइंट जहा वे मिलते है नया पाइवोट होगा

त्वरित सॉर्टिंग एल्गोरिथमः

त्वरित एल्गोरिथम का उपयोग बारी बारी से करते है जब तक की हम छोटे संभव विभाजन तक नहीं पहुच जाते। फिर प्रत्येक विभाजन पर त्वरित सॉर्ट की कार्रवाई की जाती है। हम निम्न रूप में त्वरित सॉर्ट की एल्गोरिथम को परिभाषित करते है –

चरण 1 – सबसे दाएँ सूचकांक मान को पाइवोट बनाए।

चरण 2 – पाइवोट मान का उपयोग कर ऐरे का विभाजन करे।

चरण 3 –रिक्रसीवेली बाएँ विभाजन पर त्वरित सॉर्ट लगाये।

चरण 4 – रिक्रसीवेली दॉये विभाजन पर तत्वरित सॉर्ट लगाये।

त्वरित सॉर्टिंग के लिए C प्रोग्राम:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 7
int intArray[MAX] = {4,6,3,2,1,9,7};
void printline(int count) {
int i;
for(i = 0;i <count-1;i++) {
printf("=");
}
printf("=\n");
}
void display() {
int i;
printf("[");
// navigate through all items
for(i = 0;i<MAX;i++) {
printf("%d ",intArray[i]);
}
printf("]\n");
```

```
}
void swap(int num1, int num2) {
int temp = intArray[num1];
intArray[num1] = intArray[num2];
intArray[num2] = temp;
}
int partition(int left, int right, int pivot) {
int leftPointer = left -1;
int rightPointer = right;
while(true) {
while(intArray[++leftPointer] < pivot) {
//do nothing
}
while(rightPointer > 0 && intArray[--rightPointer] > pivot) {
//do nothing
}
if(leftPointer >= rightPointer) {
break;
} else {
printf(" item swapped :%d,%d\n",
intArray[leftPointer],intArray[rightPointer]);
swap(leftPointer,rightPointer);
}
}
printf(" pivot swapped :%d,%d\n", intArray[leftPointer],intArray[right]);
swap(leftPointer,right);
printf("Updated Array: ");
display();
return leftPointer;
}
void quickSort(int left, int right) {
if(right-left <= 0) {
return;
} else {
```

```
int pivot = intArray[right];
int partitionPoint = partition(left, right, pivot);
quickSort(left,partitionPoint-1);
quickSort(partitionPoint+1,right);
}
}
main() {
printf("Input Array: ");
display();
printline(50);
quickSort(0,MAX-1);
printf("Output Array: ");
display();
printline(50);
}
```

जब उपरोक्त कोड कम्पायल और रन होगा तो आउटपुट निम्नानुसार होगाः

```
Input Array: [4 6 3 2 1 9 7 ]
==================================================
pivot swapped :9,7
Updated Array: [4 6 3 2 1 7 9 ]
pivot swapped :4,1
Updated Array: [1 6 3 2 4 7 9 ]
item swapped :6,2
pivot swapped :6,4
Updated Array: [1 2 3 4 6 7 9 ]
pivot swapped :3,3
Updated Array: [1 2 3 4 6 7 9 ]
Output Array: [1 2 3 4 6 7 9 ]
```

महत्वपूर्ण बिंदु

- सॉर्टिंग एल्गोरिथ्म डेटा को एक विशेष क्रम में व्यवस्थित करने का तरीका निर्दिष्ट करती है।
- बबल सॉर्ट एक साधारण सॉर्टिंग एल्गोरिथ्म है। यह सॉर्टिंग एल्गोरिथ्म, तुलना–आधारित एल्गोरिथ्म है जिसमें सन्निकट तत्वों के प्रत्येक जोड़े की तुलना की जाती है।
- मर्ज सॉर्ट डिवाइड (विभजित) एंड कॉन्कर (जीत) पर आधारित एक सॉर्टिंग तकनीक है। इसकी सबसे खराब मामले में (worst-case) कॉम्प्लेक्सिटी O(n log n) होने के कारण यह सबसे अच्छीएल्गोरिथ्म में से एक है।
- त्वरित एक प्रकार की अत्यंत कुशल सॉर्टिंग एल्गोरिथ्म है और डेटा के ऐरे को छोटे ऐरे में विभाजन करने पर आधारित है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्र1 बबल एल्गोरिथ्म की जटिलता है

(अ) O(N) (ब) $O(N^2)$

(स) O(logN) (द) O(NlogN)

प्र 2 मर्ज एल्गोरिथ्म की जटिलता है

(अ)O(N) (ब) $O(N^2)$

(स) O(logN) (द) O(NlogN)

प्र3 चयन एल्गोरिथ्म की जटिलता है

(अ) O(N) (ब) $O(N^2)$

(स) O(NlogN) (द) O(logN)

प्र4 कौन सा अच्छा सॉर्टिंग एल्गोरिथ्म है।

(अ) चयन सॉर्टिंग (ब) निवेशन सॉर्टिंग

(स) त्वरित सॉर्टिंग (द) कोई नहीं

प्र5 त्वरित क्रमबद्ध एल्गोरिथ्म की जटिलता है।

(अ) O(N) (ब) O(logN)

(स) $O(N^2)$ (द) O(NlogN)

लघुत्तरात्मक प्रश्न

प्र1 सॉर्टिंग क्या है?

प्र2 स्थिर सॉर्टिंग क्या है?

प्र3 इन–प्लेस सॉर्टिंग क्या है?

प्र4 त्वरित एल्गोरिथ्म के लिए सबसे खराब स्थिति (worst case) का रन टाइमहै।

प्र5 त्वरित एल्गोरिथ्म के लिए सबसे खराब स्थिति (worst case) है।

निबंधात्मक प्रश्न

प्र1 मर्ज सॉर्टिंग विस्तार में समझाईए।

प्र2 कोनसी सबसे अच्छी सॉर्टिंग एल्गोरिथ्म हैं और क्यों हैं?

प्र3 त्वरित सॉर्टिंग समझाईए।

प्र4 selection और Insertion सॉर्टिंग के बीच अंतर?

प्र5 स्थिर और अस्थिर सॉर्टिंग के बीच क्या अंतर है?

उत्तरमाला

उत्तर 1: ब उत्तर 2: दउत्तर 3: ब

उत्तर 4: स उत्तर 5: द