GRAPHICAL REPRESATION BUBBLE SORT

by Keerthi Kumar B

Submission date: 05-Nov-2019 02:29PM (UTC+0530)
Submission ID: 1207444158
File name: 1NH18CS726.pdf (482.85K)
Word count: 5090
Character count: 25063

NAME: KEERTHI KUMAR B

USN: 1NH18CS726

SEM: 3

SECTION: D

PROJECT: GRAPHICAL REPRESATION BUBBLE SORT

GRAPHICAL REPRESATION OF BUBBLE SORT

CHAPTER 1

INTRODUCTION

Sorting of an array is the method of arranging the data present in a array in an order by ascending or descending

What does sorting do

Eg: [52, 32, 71, 14, 9, 46, and 21]
Sorted order is: [9, 14, 21, 32, 46, 52, and 71]

1.1 Problem Definition

Aim behind implementation of this project is to make a clear understand ability, of various algorithms of data structures. Using the web page this will simulates the data structure operations such as - searching, sorting, insertion, and the deletion. In array, stack, queue, and linked list are present. Thus our web page also provides effective and efficient knowledge of the data structures. This also provides some of theoretical knowledge regarding the data structure.

We are given a set of array (or a list) of data. We are also given a way to "order" the elements present in the data. Now, we are asked to arrange the data as per, the listed order. As an example, we are given an array of integers: [8, 5, 6, 2, and 3]. we are given "order" as "smaller than". So, we are asked to arrange the elements of this an array in such a way that each element is smaller than its successor. Basically, we need to find a way to sort this array so that the final array obtained is [2, 3, 5, 6, and 8]. There are several methods /algorithms to achieve this ordered of the output array. Then this we can implement using bubble sort

1.2 Objectives.

To study how the operations on the data structure and algorithms are performed. And how the values are are compared in sorting algorithms and swapped. Total Number of comparison and the exchanges performed in a sorting algorithm. And the corresponding code performed while sorting so that we get a clear idea about various, data structures and operations on it. And how can we implement in a data structure.

1.3 HARDWARE REQUIRMENTS

Processor	: Any Processor above 300 MHz
RAM	: 512
Hard Disk	: 60 GB
Input device	: Standard Keyboard
Output device	: Monitor

1.4 SOFTWARE REQUIREMENTS

Operating system	: Windows XP
Front End	: ASP.Net 2.0
Server	: Internet

CHAPTER 2

2.1 Data Structure:

In computer science, a data structure is a particular way of storing and organizing data in a computer so that it can be used properly. Simulation of Data structures and Algorithms

Data structures are inevitable a part of programs. pc programs of times method knowledge thus we have a tendency to need economical ways that within which we will access or manipulate knowledge. Some applications could need modification {of knowledge ofknowledge of information} of times and in others new data is consistently extra or deleted. thus we want economical ways that of accessing knowledge thus on act thereon and build economical applications.

Data structures are of two types:
Primitive data structures and the non-primitive data structures.
Primitive data structures can also be directly manipulated by machine instructions.

Examples:
1.int
2. float
3. char
4. pointer

Non-primitive data structures cannot be directly manipulated using machine instructions...
Examples:
1.Arrays
2. lists
3. files

classified into
1.Linear lists
2. Non-linear lists.

->.Linear list consists of:
(a) Stack
(b) queues
(c) linked list.

-> Non-linear list consists of:

(a) trees
(b) graph.

ARRAY

You may have studied arrays that may be an organisation Associate in Nursing has advantage of accessing any part in constant time however adding part at the start or somewhere in an array may be an expensive operation as we've got to shift different components. Arrays could also be helpful once size of knowledge is fastened. equally we've got different knowledge structures every providing distinctive benefit. {we can wewill weare able to} opt for that organisation to use in our program in step with our demand once we have a tendency to are aware of completely different knowledge structures. Some algorithms conjointly uses specific organisation in their implementation. another knowledge structures are strings, joined lists, stack, queues, and trees. every of those have any varieties.

STACK

A stack isadditionallyAssociateinNursing ordered assortment of components like array howeverit's a special feature that deletion and insertion of componentsmaybe done solely from one finish, referredtoasthe highestof the stack .
->It maybea non-primitive arrangement.
->It isadditionallyreferredtoas last in 1st out variety ofknowledge structure (LIFO).
->Exp. Train, stack of trays .
->At the time of insertion or removal of part base of stack remains same.
->Insertion of part istermedPush. .
-> Deletion of partis termed Pop.

Disadvantages
->First part can't be 1st accessed.
->Insertion and deletion of componentswillsolely be done at one finish.
->Stacks follow contiguous dynamic memory allocation

QUEUE

Queues area unit1st in 1st out variety ofknowledge structures. during a queue new componentsarea unitadditional to the queue from one finishreferred to asand also thecomponentsarea unitcontinuallyoff fromalternative end referred to as the front.
->New componentsarea unitadditional to the queue from one finishreferred to astail end.
->Elements area unitcontinuallyoff fromalternativefinishreferred to asfront.
->Eg: queue of railway reservation.
-> Queue's follow FIFO.

Disadvantages

->Insertion of componentswillsolely be done at tail end.

->Deletion of componentswillsolely be done at front.

->Queue follows contiguous memory allocation.

As mentioned higher than these area unit the disadvantages of stacks and queue. All the disadvantages of stacks and queue area unitthe benefits of coupled list, thuscoupled list are brought into image.

TREE

A tree data structure can be defined recursively (locally) as a collection of a nodes (starting at a root node), where each node of the data structure consisting of a value, together with a list of references to nodes as (the "children"), with the constraints that no reference as duplicated and none points to the root.

a tree can be defined abstractly as a whole (globally) as an ordered tree, with a value assigned to the each node. Both these perspectives are useful: while a tree can be analysed to mathematically as a whole, when actually represented as the data structure it is usually represented and it is worked with the separated by node (rather than the list of nodes and an adjacency list of edges between nodes, as one may be represent a digraph, for the instance). For example, looking at a tree as the whole, one can talk about "the parent node" of a given node, but in general as a data structure a given node only contains the list of its children, but does not contain a reference of that parent.

important terms with respect to tree.

Path – Path is the sequence of nodes along the edges of the tree.

Root – The node is at the top of the tree which is called root. There is only one root per tree and one path from the root node to any of the node.

Parent – Any node except the root node has one edge upward to the node called parent.

Child – The node below is a given node connected by its edge downward is called its child node.

Leaf – The node which doesn't have any of the child node is called the leaf node.

Sub tree – Sub tree represents the descendants of the node.

Visiting – Visiting refers the checking the value of a node when control is on the node.

Traversing – Traversing means passing through nodes in specific order.

Levels – Level of a node represents the generation of the node. If the root node is at the level 0, then its next child node is at level 1, its grandchild is at level 2, and goes on.

keys – Key represents a value of a node based on which a search operation is to be carried out of a node.

LINKED LIST

Linked lists square measurethe foremost simplest and most typical in the knowledge structures. they'll be wont to implement manydifferent common abstract knowledgevarieties, as well as lists stacks, queues, associative arrays, notwithstandingit's not uncommon to implement the oppositeknowledge structures directly while notemploying a list because the basis of associate implementation.

The good thing about a coupled list over a traditional array is that the list componentswillsimply be inserted or removed while not a reorganization of the whole structure as a result ofthe infothingsneedn't be hold on contiguously in memory or disk. coupled lists enable insertion and removal of nodes at any purposewithin the list, and might do thus with a continuingvariety of operations if the link previous to the link being supplemental or removed is maintained throughout list traversal.
In technology, one amongstthe foremost basic and elementaryknowledge structures is that coupledlist that functions equally to Associate in Nursing array. A coupled list may be a linear assortmentof informationcomponentsreferred to as nodes that time to consecutive node by suggests that of a pointer. The principal good thing about a coupled list over a traditional array is that the list componentsmay besimply be inserted or removed while not reallocation of the othercomponents. In some programming languages the dimensions of Associate in Nursing array may be a concern and one amongst the ways thatto beat that drawbackand permit dynamically allottedknowledgeis exploitation coupled lists. The disadvantage of the coupled lists is, once it involves reverse traversing. as an example, single coupled lists area unit cumbersome to navigate backwards and whereas doubly coupled lists area unit somewhat easier to browse, memory is wasted in allocating house for a back-pointer. So, in my project I'm going use the info structure referred to as doubly coupled list.

On the opposite hand, straightforwardcoupled lists by themselves don'tenable random access to the info, or any kind ofeconomicalclassification. Thus, several basic operations — likegetting the last node of the list (assuming that the last node isn't maintained as separate node reference within the list structure), or finding a node that contains a given data point, or locating the place wherevera brand new node ought to be inserted — couldneed scanning most or all of the list components.

DOUBLE LINKEDLIST

Doubly coupled list may be acoupledarrangement that consists of a group of consecutivecoupled records referred to as nodes. every node contains 3 fields: 2coupled fields (references to the previous and to consecutive node within the sequence of nodes) and one knowledge field. the start and ending nodes' previous and next links, severally, purpose to some reasonablyexterminator, usually a NULL, to facilitate traversal of the list. the 2 node links permit traversal of the list in either direction. whereas adding or removing a node during a doubly coupled list needsever-changinga lot of links than identical operations on a individuallycoupled list, the operations area uniteasier and probablya lot ofeconomical (for nodes aside fromthe primary nodes) as a result ofthere's no ought to keep track of the previous node throughout traversal or no ought to traverse the list to search out the previous node, so its link may bechanged.

BUBBLESORT

Bubble type may be an algorithm, that is usually utilized in computing. Bubble type is predicated on the concept of repeatedly comparison pairs of adjacent components and so swapping their positions if they exist within the wrong order.
Bubble type, typicallysaid as sinking type, may be astraightforwardalgorithm that works by repeatedly stepping through the list to be sorted, scrutinyeverytry of adjacent things and swapping them if they'rewithin the wrong order. The labour under the list is continualtill no swaps square measurerequired,

which indicates that the list is sorted. The formula gets its name from the approach smaller components "bubble" to the highest of the list. as a result of it solely uses comparisons to control on components, it's a comparison sort has O(n2) time quality, creating it inefficient on massive lists, and usually performs worse than the similar insertion type. choicetype is noted for its simplicity, and its performance the benefits over additionaldifficult algorithms in some surethings, significantlywherever auxiliary memory is proscribed.
The formula divides the input list into 2 division : the sub list of things already sorted, thatis constructed up from left to right at the front (left) of the list, and therefore the sub list of things remaining to be sorted that occupy the remaining of the list. Initially, the sorted sub list is empty and therefore the unsorted sub list is that the entire input list.
The formulayield by finding the littlest (or largest, counting on sorting order) componentwithin the unsorted sub list, exchanging it with the left unsorted component (putting it in sorted order), and moving the sub list boundaries one component to the correct.

General disadvantages

Linked knowledge structures perhaps substantial memory allocation overhead (if nodes square measureallotted individually) and frustrate memory paging and processor caching algorithms .In some cases, coupledknowledge structures may additionally uses additional memory than competitive array structures. this is oftenas a result ofcoupledknowledge structures isn't contiguous. Instances of knowledgeis found everywherewithin the memory, in contrast to arrays.
In arrays, ordinalcomponentis accessed straightaway, whereasin a muchcoupledorganizationwe've to follow multiple pointers therefore thecomponentinterval varies in keeping withwhereverwithin the structure the component is.

INSERTION SORT

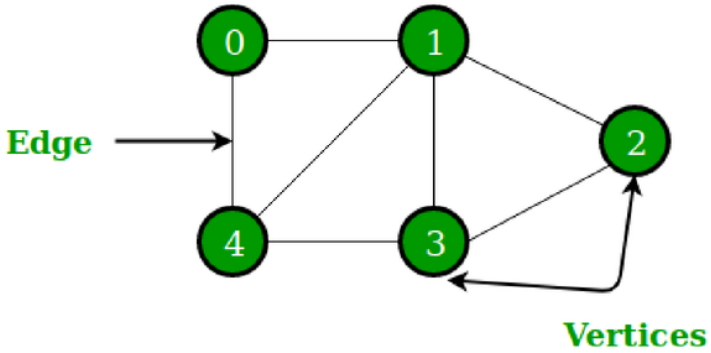
Insertion typemay be astraightforwardalgorithm that works the approachwe tend totype taking part in cardsin your hand.

QUICKSORT

Quicksort may be analgorithm developed by the Tony Hoare that, on average, makes O(n log n) comparisons to type n no of things. within the worst case, it makes O(n2) comparisons, though' the behaviour is rare. Quicksort is commonlyquicker in follow than the opposite O(n log n) algorithms .Additionally, quicksort's serial and localized memory of references work well with a cache. Quicksort may be a comparison type and, in economical implementations, isn't the stable type. Quicksort isenforcedvictimizationassociate in-place partitioning formula, therefore the entire typeisthrough withsolely O(log n) extrahouseemployed by ants stack throughout the rule.

Graph

Graph is the non-linear data structures consisting of nodes and the edges. The nodes are sometimes also referred to as vertices and edges these lines or arcs that connect any two nodes in a graph. More formally Graphs can be defined like
The Graph consists of finite set of vertices (nodes) and set of the Edges which connects a pair of the nodes.



In above Graph, the set of the vertices V = {0,1,2,3,4} and the set of edges E = {01, 12, 23, 34, 04, 14, 13}.

Basic Operations

Following are some of basic primary operations of the Graph −

Add a Vertex − Adds the vertex to graph.

Add the Edge − Adds an edge b/w the two vertices of a graph.

Display the Vertex − Displays the vertex of a graph.

GRAPHICS

Description

Data Structure, Computers Graphics and the Pattern Recognition focus on the computer graphics and the pattern recognition applications of the data structures methodology.

This book presents the design related principles and the research aspects of a computer graphic, system designs, data managements, and pattern recognition task. The topic includes the data structure design, concise structuring of geometric data for the computer aided design, and the data structures for pattern recognition algorithm. The survey of the data structures for the computer graphics systems, applications of relational data structures in computer graphics, and observations on linguistics for scene analysis are elaborated. text likewise covers the design of satellite graphics system, interactive images segmentation, surface representation for computer aided designs, and error-correcting parsing for the syntactic pattern recognitions.

This publication is valuable to the practitioner in the data structure, particularly those who are applying the real computer systems to problems involving images, speeches, and medical's data.

Pre-defined Functions Used:

setcurrentwindow():A function which is used to set the size of current window.

setcolor(n): A function which is used to change the color of cursor by changing the value of n.

delay(n): A function which is used to delay the program by n milliseconds. It is being used for slowing down the transitions speed

line(x1, y1, x2, y2): A function which is used to draw an line from point (x1, y1) to point (x2, y2). (0, 0) being the left top corner of the screen and bottom right be (n1, n2) where n1, n2 are the width and height of the current window. There are other graphics which can be applied to this line using setcolor().

CHAPTER 3

Algorithm

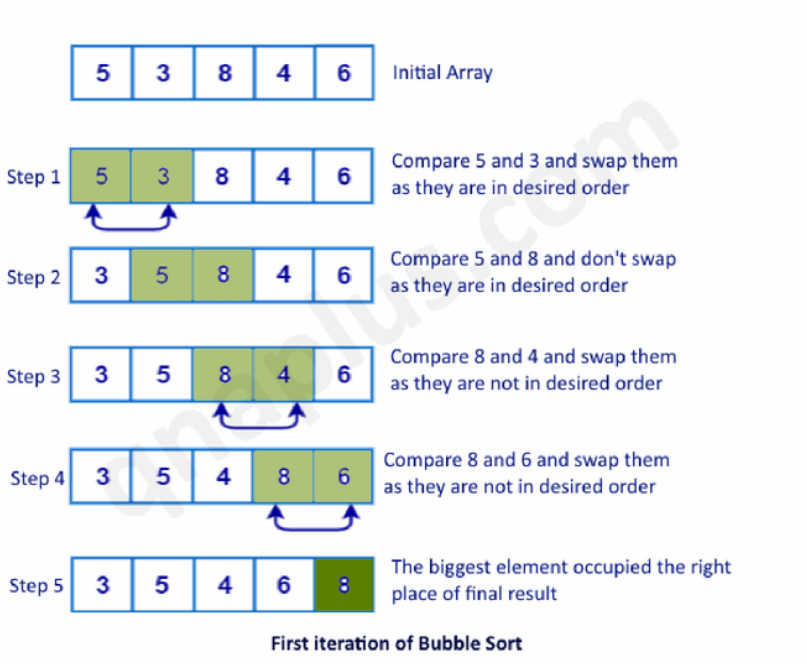
Continuously swap 2 adjacent components if they're not within the right order till no swapping is needed.
In programming, 2 nested loops square measuretypically used. Inner loop starts from the primary position and checks whether or not the component of current position is in desired order with subsequentcomponent. If not then it swaps them. Then the management moves to subsequent position and repeats constantissue. whenfinishof each iteration of the outer loop, one component is placed at the correct position.
for instance, if we tend tosquare measure sorting associate array in ascending order, then the most importantcomponentinvolves the last position when completion the primary iteration of the outer loop. Similarly, when completion of the second iteration, the second biggest component takes the second last position. during thisapproachwhen n-1 iterations, n-1 components take their right positions and therefore the remaining last componenthas got to be within the right position.

biggest component takes the last position of the array when completion of 1st iteration of the outer loop. Similarly, when completion of second iteration, the second biggest componentcan take the second last position. If we tend to continue this method for n-1 time, then all n componentscan take their right positions within the final sorted array. The human brain can easily process visuals in the spite of long codes to understand the algorithms. Bubble sort is the visualization has been implemented using graphics.h library. As we all know that bubble sort swaps the adjacent elements if they are unsorted and finally the larger one has been shifted towards the end of array in each pass. Sometimes, it becomes difficult to analyse the data in manually, but after plotting graphically is much easier to understand

In Associate in Nursing unsorted array of five components, begin with the primary 2 components and kind them in ascending order. (Compare the part to envision that one is greater).
Compare the second and third part to envision that one is bigger, and kind them in ascending order.
Compare the third and fourth part to envision that one is bigger, and kind them in ascending order.
Compare the fourth and fifth part to envision that one is bigger, and kind them in ascending order.
Repeat steps 1–5 till now a lot of swaps are needed.
Characteristics of Bubble Sort:
Large values are perpetually sorted 1st.
It solely takes one iteration to discover that a group is already sorted.

The best time quality for Bubble type is O(n). the common and worst time quality is $O(n^2)$.
The area quality for Bubble type is O(1), as a result of solely single extra memory area is needed.

.



First iteration of Bubble Sort

.

for comparing two of the types of data-set is also difficult with numbers if the size of the array is large.
The graphical representation of the randomly distributed numbers and reverse sorted numbers
The white line is used to represent the length of the number (9 being represented by 9 pixels vertically upwards) while it's a position represents its index in a array.
Graphically sorting can be shown simply through swapping the lines.
As we swap the numbers in bubble sort, a different colour line can be used to see the current index in array .
Here delay() can be increased to see the transition in a graph..

Bubble kindalgorithmic rule of ascending order can move higher worthcomponents to right and lower value components to left.
• For this we wantto matchevery adjacent components in associate degree array and

check they'reso as or not if not swap those 2components.
• After completion of 1st iteration highest worthwithin the array are going to bemoved to last or final position.
• In the worst case all the bestcomponents in array like [9 eightsevenfive utterly in reverse order thereforewe want to restate all of the weather in N times.

•Repeat this method N- one time in worst case to kind set of components.
•Worse case complexness : O(n2)
•In the simplest case state of affairs all the weathersquare measure already in thereforerted order so one iteration with no swaps needed.
•Best case complexity: O(n)
• Let U.S. see the graphical illustration of clarification of bubble kindalgorithmic rule in c programing language.

1 All the header files are included and global variables are declared
2. All the user defined functions are also defined
3. Later in main function all the local variables in main are declared

4.A lop is created to see the desires numbers of inputs from the user
5.All the user defined function called like initigraph, settextstyle and outtextxy.where the graphic functions are defined and called
6.Later again the loop is called to set the object functionality
7.The bubble function is called where all the elements are sorted in ascending order

8.In bubble function a for loop is created with two loop conditions followed if conditions where two elements are compared at a time and arranged accordingly.
9.The same above process is display using graphics with some visual effects and a colours are initiated
10.With the help of object function the diagrammatic representation of the numbers are done. Inside object a circle function is called where the circular effect is created around the element.

11.Settextstyle function sets the style for the element so that the presentation will be effective
12.Flow function is used to move the elements around and arrange according to the ascending order
13.Setcolor function sets the colour to the diagram and texts. There is unique color-coded for all the colours in this function
14. Mixing function is used to check the ascending order of the function and mixing them in between the elements until the desired destination is reached
15. Delay function us used to delay the timings of all the process so that the process is visible completely

His algorithm follows the concept of iterating through the array from the first index to the last index and comparing adjacent elements and then swapping them if they appear in the wrong order. i.e. If the next element is smaller than of current element, they are swapped.

Let us understand this with the help of an ex. Let us take an input array such as – 9 3 1 7 2

The sorted output for this array is 1 2 3 7 9

Let us see the step by step of execution of the Bubble sort algorithm on the input array to get the sorted output.

Iteration 1: –9 3 1 7 2→ 3 9 1 7 2→ 3 1 9 7 2

Iteration 2: –3 1 2 9 7→ 3 1 2 7 9→ 1 3 2 7 9

Iteration 3: 1 3 2 7 9→ 3 1 2 7 9→ 2 1 3 7 9

Iteration 4: –1 2 3 7 9

CHAPATER 4

IMPLEMENTATION

```
void bubble sort(in sr, int t)
{
int sr = 0, j = 0;
int tmp = 0;

for(h=0;h<k-1;h++)
{
for (m =0; m < k-1; m++)
{ 10
if(Sr[m+1] < sr[m])
{
tmp = Sr[m];
Sr[m] = sr[m+1];
Sr[m+1] = tmp;
}
}
}
}
```

This is terriblysimpleto grasp variation. during this case, the outer loop and inner loop eachrestate n-1 times wherever n is length of the array. The inner loop starts from the primarycomponent and checks with followingcomponentwhether or not theysquare measure in correct order. If not, then it swaps the weather. during this example we have a tendency tosquare measure sorting the array in ascending order. therefore if thiscomponent (indexed by j) is larger than followingcomponent (indexed by j+1), then we have a tendency to swap them.

Then management moves to followingcomponent. during this case, we'll have (n-1)*(n-1) comparisons continuouslythat is a lot of the the utmost comparison needed by bubble kind even within the worst case. within the following sections, we'll see howeverwe will improve it to reducerange of comparisons.

Our implementation can further improve. In the previous implementations, we were reducing the inner loop iterations by one for every outer loop of iteration. But we can also remember where (at which index) we did the last swapping the inner loop. The rest of the array has already sorted. In the next iteration of the outer loop, we can stop the inner loop where the last swapping in previous iteration.

```
void bubble(int a)
```

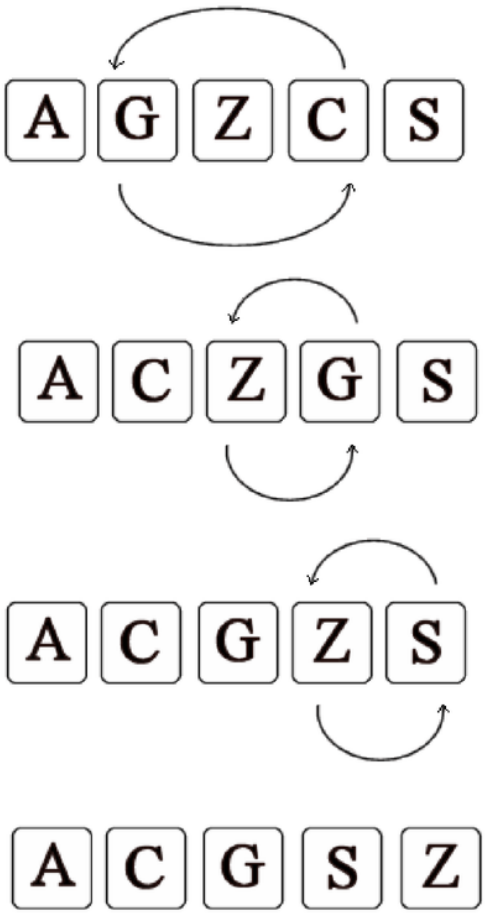
```
{

int i,j,t;

for(i=0;i<n;i++)

for(j=i;j<n-1;j++)

if(a[i]>a[j+1])

{

flow(i,j+1);

t=a[i];

a[i]=a[j+1];

a[j+1]=t;

}

}
```

Here the *newp* variable remembers where the inner loop did last swapping. Next time, the inner loop will go up to that point (up to the *newp*). This is the most optimized implementation in bubble sort as per the number of comparisons is concerned.

The bubble sort is also known as *ripple* sort. The bubble sort is probably the first reasonably complex module that any beginning of the programmer has to write. It is a very simple construct which introduces the student to the fundamentals of how sorting has to work.

A bubble sort makes use of an array and some sort of the "swapping" mechanism. Most programming languages have a built-in function to swap elements of the array. Even if a swapping function does not exist, only a couple of extra lines of code are required to be stored in one array element in a temporary field in order to swap a second element into its place. Then the first element has moved out of the temporary field and back into the array at the second element's position.

Here is a simple example of how a bubble sort works: Suppose you have a row of children's toy blocks with letters on theres. They are in random order and you wish to arrange them in alphabetical order left to right.




```
void object(int x,int y,int no)
{
Char a[8];
sprintf(a,"%d",no);
circle(x,y,25);
settextstyle(8,0,6);
outtextxy(x-4,y-50,e);
}
```

The above given pseudo code is the one of the important part of the project. This function used to create the circle of desired size .this function is also used to set the font size , direction and size of an character. This function is also used to represent the coordinates of the elements.

```
void flow(int g,int i)
{
int k;
for(k=0;k<50;k++)
{
setcolor(WHITE);
object(200+g*20,750+i*8,a[i]);
delay(25);
setcolor(0);
object(200+i*20,750+i*8,a[i]);
}
setcolor(WHITE);
object(200+g*20,750+i*8,a[i]);
for(k=0;k<50;k++)
{
setcolor(WHITE);
object(200+g*50,750+i*8,a[i]);
delay(20);
setcolor(0);
object(200+i*20,750+i*8,a[i]);
}
setcolor(WHITE);
object(200+i*50,750+i*8,a[i]);
mixing(g,i);
for(k=50;k>0;k--)
{
setcolor(WHITE);
object(200+g*20,750+i*8,a[i]);
delay(10);
setcolor(0);
object(200+g*20,750+i*8,a[i]);
}
setcolor(WHITE);
object(200+g*20,750+i*8,a[i]);
for(k=50;k>0;k--)
{
setcolor(WHITE);
object(200+i*20,750+i*8,a[i]);
delay(15);
setcolor(0);
object(200+i*20,750+i*8,a[i]);
}
```

```
setcolor(WHITE);
object(200+i*20,750+i*8,a[i]);
}
```

This above function is used to set colours of different formats. This function includes a setcolor format which is defined in graphics.h which is used to set colours consecutively white after 50 msec the colour changes to black, thus it creates a flow of blinking an object.

```
void mixing(int i ,int g)
{
int i;
For (i=0; i< (i-g)*50; i++)
{
setcolor(WHITE);
object(200+g*20+i,750,a[g]);
object(200+i*20-i,750,s[i]);
delay(200);
setcolor(0);
object(200+g*20+i,750,a[g]);
object(200+i*20-i,750,s[i]);
}
setcolor(WHITE);
Object(200+g*20+i,750,a[g]);
Object(200+i*20-i,750,s[i]);
}
```

This function is similar to the above flow function, but in this case it mixes the elements according to the sort and set the colour again to white after 20 msec it changes into black. the only difference between the above two functions is that it takes more time than the previous function.

Conclusion

Bubble sort is a fairly simple algorithm. It forms an interesting example of how simple computations can be used to perform more complex tasks. However, there is one issue with the algorithm - it is relatively slower compared to other sorting algorithms. To understand that, let us take a look at the loops involved - there are 2 loops:

First, the outer loop of variable i that goes from i = 0 to i = n - 1.
For each iteration of the outer i loop, the inner loop of variable j goes from j = 0 to j = n - i - 2.
We can consolidate the number of iterations to see that:

When i = 0, the inner j loop goes from j = 0 to j = n - 2
When i = 1, the inner j loop goes from j = 0 to j = n - 3
When i = 2, the inner j loop goes from j = 0 to j = n - 4
When i = n - 2, the inner j loop goes from j = 0 to j = 0
We can sum this up to see that the total iterations are (n - 2) + (n - 3) + (n - 4) ... + 1 + 0 = (n - 2) * (n - 3) / 2 = (n2 - 5n + 6) / 2 = n2/2 - 2.5n + 3As can be seen, this term is proportional to n2 (the largest power of n is n2). Mathematically, this is stated as - bubble sort algorithm is of O(n2) complexity. This isn't the best because when n is large (say n = 106), n2 is huge (n2 = 1012). Therefore, it will take a lot of iterations for the algorithm to complete. This is undesirable. There are some better algorithms like merge sort in C, etc. that take O(nlog2n) iterations. logn is much smaller than n. As an example, when n = 230 (which is approximately 109), log2n is just 30). Nevertheless, bubble sort is an interesting algorithm and is a great way for beginners to understand how sorting works.

GRAPHICAL REPRESATION BUBBLE SORT

ORIGINALITY REPORT

7%	%	7%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Ching-Kuang Shene. "A comparative study of linked list sorting algorithms", 3C ON-LINE, 1996 Publication	2%
2	Mustafa Aksu, Ali Karcı. "Pyramid Search: Skip Ring Data Structure-Based New Searching Algorithm", Journal of Circuits, Systems and Computers, 2018 Publication	1%
3	Wen Bo Wang, Mao Lin Huang, Liangfu Lu, Jinson Zhang. "Improving Performance of Forensics Investigation with Parallel Coordinates Visual Analytics", 2014 IEEE 17th International Conference on Computational Science and Engineering, 2014 Publication	1%
4	"Preface", Elsevier BV, 1977 Publication	1%
5	Sa Ban-Hao, Cheuk-Yin Wong. "Transverse-momentum distribution of produced particles in	1%

ultrarelativistic nucleus-nucleus collisions", Physical Review D, 1985
Publication

---

6 Karsten Schmidt, Theo Härder. "Usage-driven storage structures for native XML databases", Proceedings of the 2008 international symposium on Database engineering & applications - IDEAS '08, 2008 — 1%
Publication

---

7 Steven Margerm, Amirali Sharifian, Apala Guha, Arrvindh Shriraman, Gilles Pokam. "TAPAS: Generating Parallel Accelerators from Parallel Programs", 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2018 — <1%
Publication

---

8 Aksu, Mustafa, and Ali Karci. "Skip List Data Structure Based New Searching Algorithm and Its Applications: Priority Search", International Journal of Advanced Computer Science and Applications, 2016. — <1%
Publication

---

9 T. Yagiu. "A predicate-logical method for modelling design objects", Artificial Intelligence in Engineering, 1989 — <1%
Publication

---

10 Dimitrie D. Stancu. "The remainder in the

	approximation by a generalized Bernstein operator: a representation by a convex combination of second-order divided differences", Calcolo, 1998 Publication	<1%
11	D. C. Brown. "Design considerations for Picture Production in a Natural Language graphics system", ACM SIGGRAPH Computer Graphics, 7/1/1981 Publication	<1%
12	George H. Duffey. "Chapter 2 Quantization of Rotatory Motion", Springer Nature, 1984 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		