INFORMATIKA
UMN

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

IF232

ALGORITHMS & DATA STRUCTURES

11

SORTING 1

DENNIS GUNAWAN

REVIEW

Heaps:

Binary Heaps

Applications of Heaps

OUTLINE

Bubble Sort

Selection Sort

Insertion Sort

Radix Sort

SORTING

- Sorting data: placing the data into a particular order such as **ascending** or **descending**
- Many programs execute more efficiently if the data they process are sorted before processing begins
- Other algorithms (some search algorithms) require input data to be in sorted lists

BUBBLE SORT

- A sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items, and swaps them if they are in the wrong order
- Sometimes referred to as sinking sort
- The smaller values gradually “bubble” their way upward to the top of the array like air bubbles rising in water, while the larger values sink to the bottom of the array

BUBBLE SORT

- Easy to program
- Runs slowly because every exchange moves an element only one position closer to its final destination

BUBBLE SORT EXAMPLE [ASCENDING]

1ST ITERATION

0	1	2	3	4	5
84	36	52	49	10	77
84	36	52	49	10	77
84	36	52	10	49	77
84	36	10	52	49	77
84	10	36	52	49	77
10	84	36	52	49	77

BUBBLE SORT EXAMPLE [ASCENDING]

2ND ITERATION

0	1	2	3	4	5
10	84	36	52	49	77
10	84	36	52	49	77
10	84	36	49	52	77
10	84	36	49	52	77
10	36	84	49	52	77

BUBBLE SORT EXAMPLE [ASCENDING]

3RD ITERATION

0	1	2	3	4	5
10	36	84	49	52	77
10	36	84	49	52	77
10	36	84	49	52	77
10	36	49	84	52	77

BUBBLE SORT EXAMPLE [ASCENDING]

4TH ITERATION

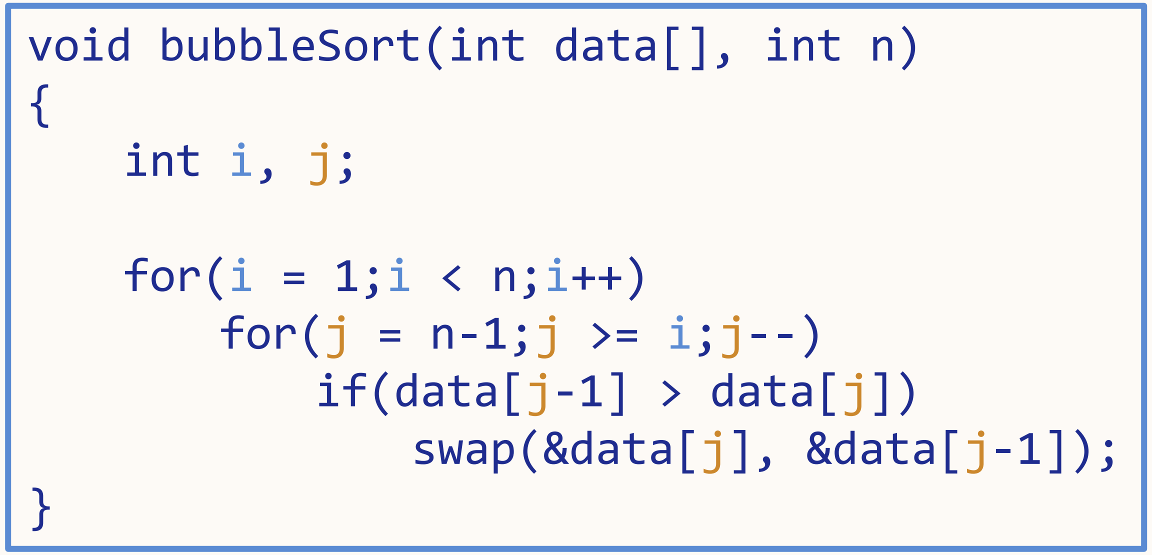
0	1	2	3	4	5
10	36	49	84	52	77
10	36	49	84	52	77
10	36	49	52	84	77

BUBBLE SORT EXAMPLE [ASCENDING]

5TH ITERATION

0	1	2	3	4	5
10	36	49	52	84	77
10	36	49	52	77	84

BUBBLE SORT [ASCENDING]



```
void bubbleSort(int data[], int n)
{
    int i, j;

    for(i = 1;i < n;i++)
        for(j = n-1;j >= i;j--)
            if(data[j-1] > data[j])
                swap(&data[j], &data[j-1]);
}
```

OPTIMIZED BUBBLE SORT

- The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted

OPTIMIZED BUBBLE SORT [ASCENDING]

```
void bubbleSort(int data[], int n)
{
    int sorted = 0, i = 1, j;
    while(!sorted)
    {
        sorted = 1;
        for(j = n-1;j >= i;j--)
            if(data[j-1] > data[j])
            {
                swap(&data[j], &data[j-1]);
                sorted = 0;
            }
        i++;
    }
}
```

SELECTION SORT

- The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted subarray list, swapping it with the leftmost unsorted element, and moving the subarray list boundaries one element to the right
    - Find the lowest element
        - Scan all n elements, select the lowest element, and swap it into the first position
    - Find the next lowest element
        - Scan the remaining n-1 elements, select the lowest element, and swap it into the second position
    - And so on

SELECTION SORT EXAMPLE [ASCENDING]

1ST ITERATION

0	1	2	3	4	5
84	36	52	49	10	77
84	36	52	49	10	77
84	36	52	49	10	77
84	36	52	49	10	77
84	36	52	49	10	77
10	36	52	49	84	77

SELECTION SORT EXAMPLE [ASCENDING]

2ND ITERATION

0	1	2	3	4	5
10	36	52	49	84	77
10	36	52	49	84	77
10	36	52	49	84	77
10	36	52	49	84	77
10	36	52	49	84	77

SELECTION SORT EXAMPLE [ASCENDING]

3RD ITERATION

0	1	2	3	4	5
10	36	52	49	84	77
10	36	52	49	84	77
10	36	52	49	84	77
10	36	49	52	84	77

SELECTION SORT EXAMPLE [ASCENDING]

4TH ITERATION

0	1	2	3	4	5
10	36	49	52	84	77
10	36	49	52	84	77
10	36	49	52	84	77

SELECTION SORT EXAMPLE [ASCENDING]

5TH ITERATION

0	1	2	3	4	5
10	36	49	52	84	77
10	36	49	52	84	77
10	36	49	52	77	84

SELECTION SORT [ASCENDING]

```
void selectionSort(int data[], int n)
{
    int i, j, temp;
    for(i = 0;i < n-1;i++)
    {
        temp = i;
        for(j = i+1;j < n;j++)
            if(data[j] < data[temp])
                temp = j;
        if(temp != i)
            swap(&data[temp], &data[i]);
    }
}
```

INSERTION SORT

- When people manually sort cards in a bridge hand, most use a method that is similar to insertion sort
- Each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there
- It repeats until no input elements remain

INSERTION SORT EXAMPLE [ASCENDING]

1ST ITERATION

temp	0	1	2	3	4	5
	84	36	52	49	10	77
36	84		52	49	10	77
36		84	52	49	10	77
	36	84	52	49	10	77

INSERTION SORT EXAMPLE [ASCENDING]

2ND ITERATION

temp	0	1	2	3	4	5
	36	84	52	49	10	77
52	36	84		49	10	77
52	36		84	49	10	77
	36	52	84	49	10	77

INSERTION SORT EXAMPLE [ASCENDING]

3RD ITERATION

temp	0	1	2	3	4	5
	36	52	84	49	10	77
49	36	52	84		10	77
49	36	52		84	10	77
49	36		52	84	10	77
	36	49	52	84	10	77

INSERTION SORT EXAMPLE [ASCENDING]

4TH ITERATION

temp	0	1	2	3	4	5
	36	49	52	84	10	77
10	36	49	52	84		77
10	36	49	52		84	77
10	36	49		52	84	77
10	36		49	52	84	77
10		36	49	52	84	77
	10	36	49	52	84	77

INSERTION SORT EXAMPLE [ASCENDING]

5TH ITERATION

temp		0	1	2	3	4	5
		10	36	49	52	84	77
77		10	36	49	52	84	
77		10	36	49	52		84
		10	36	49	52	77	84

INSERTION SORT [ASCENDING]

```
void insertionSort(int data[], int n)
{
    int i, j, temp;
    for(i = 1;i <= n-1;i++)
    {
        temp = data[i];
        for(j = i-1;j >= 0 && data[j] > temp;j--)
            data[j+1] = data[j];
        data[j+1] = temp;
    }
}
```

RADIX SORT

- Radix sort is a linear sorting algorithm and uses the concept of sorting names in alphabetical order
- When we have a list of sorted names, the radix is 26 (or 26 buckets)
  - There are 26 letters in the English alphabet
- While sorting the numbers, sorting is done on each of the digits in the number
  - Ten buckets: each for one digit (0, 1, 2, …, 9)
  - The number of passes will depend on the length of the number having maximum number of digits
- Radix sort is also known as bucket sort

RADIX SORT (LSD) EXAMPLE [ASCENDING]

1ST ITERATION

Sort the numbers given below using radix sort.

345	654	924	123	567	472	555	808	911
0	1	2	3	4	5	6	7	8

0			
1	911		
2	472		
3	123		
4	654	924	
5	345	555	
6			
7	567		
8	808		
9			

1st Iteration Result

911	472	123	654	924	345	555	567	808
0	1	2	3	4	5	6	7	8

RADIX SORT (LSD) EXAMPLE [ASCENDING]

2ND ITERATION

911	472	123	654	924	345	555	567	808
0	1	2	3	4	5	6	7	8

0	808		
1	911		
2	123	924	
3			
4	345		
5	654	555	
6	567		
7	472		
8			
9			

2nd Iteration Result

808	911	123	924	345	654	555	567	472
0	1	2	3	4	5	6	7	8

RADIX SORT (LSD) EXAMPLE [ASCENDING]

3RD ITERATION

808	911	123	924	345	654	555	567	472
0	1	2	3	4	5	6	7	8

0			
1	123		
2			
3	345		
4	472		
5	555	567	
6	654		
7			
8	808		
9	911	924	

3rd Iteration Result (**Final Sorted Result**)

123	345	472	555	567	654	808	911	924
0	1	2	3	4	5	6	7	8

RADIX SORT (LSD) [ASCENDING]

```
Step 1:  Find the largest number in ARR as LARGE
Step 2:  [INITIALIZE] SET NOP = Number of digits in LARGE
Step 3:  SET PASS = 0
Step 4:  Repeat Step 5 while PASS <= NOP-1
Step 5:     SET I = 0 and INITIALIZE buckets
Step 6:     Repeat Steps 7 to 9 while I < N-1
Step 7:        SET DIGIT = digit at PASSth place in ARR[I]
Step 8:        Add ARR[I] to the bucket numbered DIGIT
Step 9:        INCREMENT bucket count for bucket numbered DIGIT
            [END OF LOOP]
Step 10:    Collect the numbers in the bucket
         [END OF LOOP]
Step 11: END
```

PRACTICE

EXERCISES

1. Trace the execution of the sorting algorithms below on the following list to get the values in **ascending** and **descending** order.

77	49	25	12	9	33	56	81
0	1	2	3	4	5	6	7

a. Bubble sort

b. Selection sort

c. Insertion sort

d. Radix sort (LSD)

e. Radix sort (MSD)

EXERCISES

2. Trace the execution of the sorting algorithms below on the following list to get the values in **ascending** and **descending** order.

card	bake	done	dear	band	bear	cake	deal	came
0	1	2	3	4	5	6	7	8

a. radix sort (LSD)
b. radix sort (MSD)

REFERENCES

- Deitel, P. and Harvey Deitel (2022), C How to Program (9th Edition), Pearson Education.
- Thareja, R. (2014), Data Structures Using C (2nd Edition), India: Oxford University Press.

NEXT

Sorting:

Merge Sort

Quick Sort

Shell Sort

Heap Sort

VISION

To become an **outstanding** undergraduate Computer Science program that produces **international-minded** graduates who are **competent** in software engineering and have **entrepreneurial spirit** and **noble character**.

MISSION

1. To conduct studies with the best technology and curriculum, supported by professional lecturer
2. To conduct research in Informatics to promote science and technology
3. To deliver science-and-technology-based society services to implement science and technology

Without hard work,
nothing grows but weeds.

UMN
UNIVERSITAS MULTIMEDIA NUSANTARA

INFORMATIKA UMN

Have patience.
All things are difficult before they become easy.