جامعة المنارة

كلية: الهندسة

قسم: الهندسة المعلوماتية

اسم المقرر: الخوارزميات و بنى المعطيات2

رقم الجلسة (الثامنة)

عنوان الجلسة

خوارزميات الترتيب

الفصل الدراسي الثاني

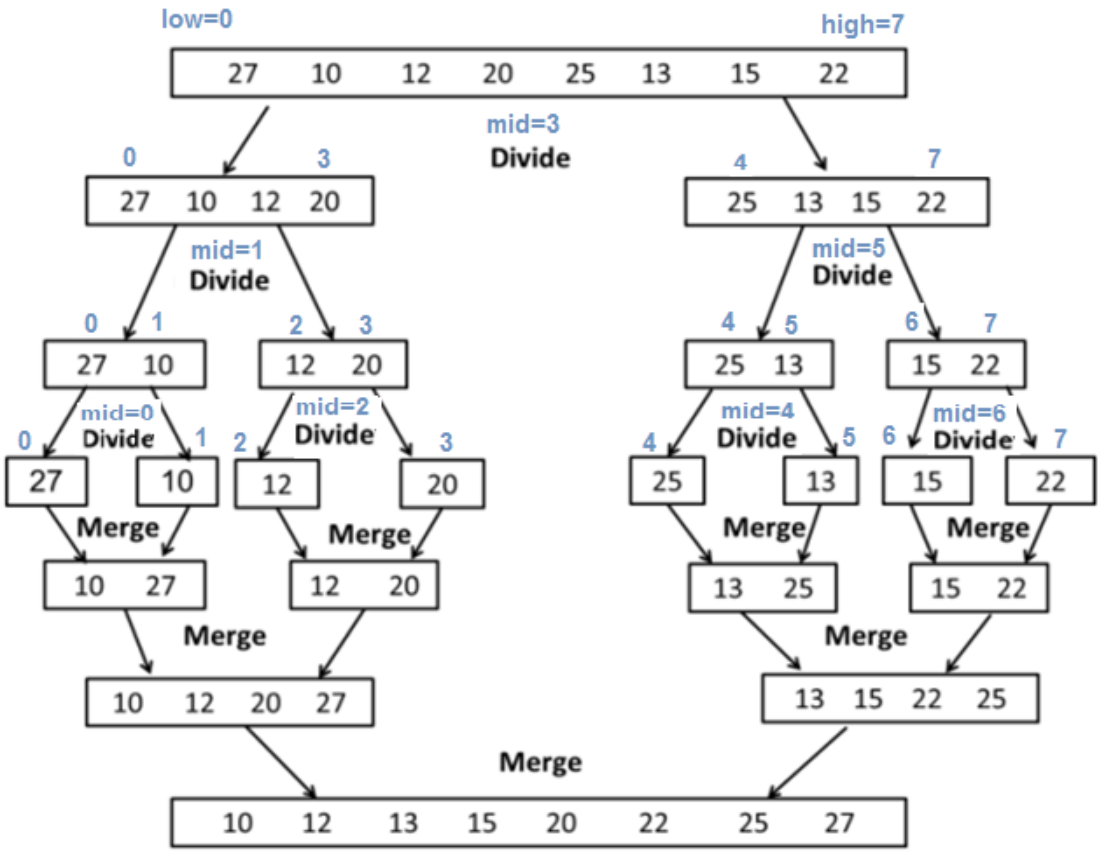
العام الدراسي 2023- 2024

الغاية من الجلسة

- ✓ تطبيق خوارزمية الترتيب بالدمج لترتيب عناصر مصفوفة أحادية .
- ✓ تطبيق خوارزمية الترتيب بالكومة لترتيب عناصر مصفوفة أحادية.
- ✓ تنفيذ الأكواد الخاصة بالخوارزميتين السابقتين.

خوارزمية الترتيب بالدمج Merge sort algorithm

مثال : طبق خوارزمية الفرز بالدمج لترتيب عناصر المصفوفة التالية : 27,10,12,20,25,13,15,22



البرنامج :

```
// C++ program for Merge Sort
#include <iostream>
using namespace std;

// A function to merge the two half into a sorted data.
void Merge(int *arr, int low, int high, int mid)
{
 // We have low to mid and mid+1 to high already sorted.
 int n1 = mid-low+ 1;
int n2 = high-mid;
```

```
int L[100], M[100];
for (int i = 0; i < n1; i++)
L[i] = arr[low + i];
for (int j = 0; j < n2; j++)
M[j] = arr[mid + 1 + j];
// Maintain current index of sub-arrays and main array
int i, j, k;
i = 0;
j = 0;
k = low;
while (i < n1 && j < n2) {
if (L[i] <= M[j]) {
arr[k] = L[i];
i++;
} else {
arr[k] = M[j];
j++;
}
k++;
}

while (i < n1) {
arr[k] = L[i];
i++;
k++;
}
while (j < n2) {
arr[k] = M[j];
j++;
k++;
}
}

void MergeSort(int *a, int low, int high)
{
 int mid;
 if (low < high)
 {
      mid=(low+high)/2;
      // Split the data into two half.
      MergeSort(a, low, mid);
      MergeSort(a, mid+1, high);

      // Merge them to get sorted output.
      Merge(a, low, high, mid);
 }
}
int main()
```

```
{
int n, i;
cout<<"\nEnter the number of data element to be sorted: ";
    cin>>n;
int arr[100];
for(i = 0; i < n; i++)
{
     cout<<"Enter element "<<i+1<<": ";
     cin>>arr[i];
}

MergeSort(arr, 0, n-1);

// Printing the sorted data.
cout<<"\nSorted Data\n ";
for (i = 0; i < n; i++)
        cout<<arr[i]<<" ";

return 0;
}
```

خوارزمية الترتيب بالكومة Heap sort algorithm

- ✓ تعتمد الخوارزمية على شجرة الكومة الثنائية Heap Binary Tree .
- ✓ تتميز شجرة الكومة بأنها Complete binary tree ، و تحقق خاصية أن قيمة كل عقدة أب تكون أكبر من قيمتي ابنيها اليساري و اليميني .
- ✓ تعتمد الخوارزمية على الخطوات التالية :

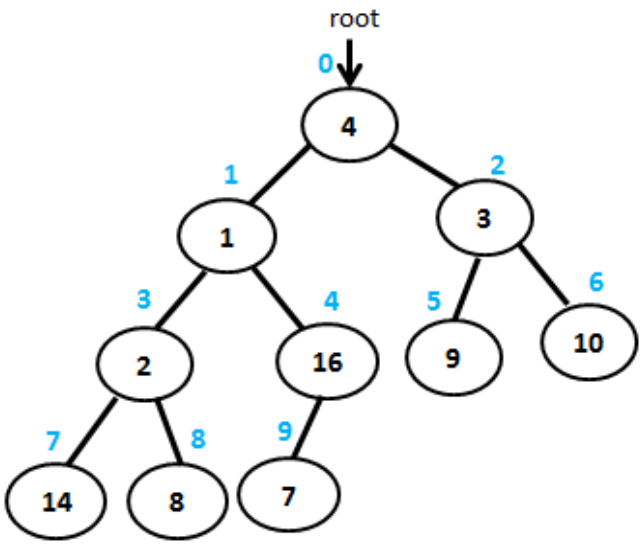
1. إنشاء شجرة كومة عظمى max heap tree من البيانات المدخلة.
2. يخزّن العنصر الأكبر كجذر للكومة، ثم يستبدل بالعنصر الأخير في الكومة ويتبع ذلك تقليل حجم الكومة بمقدار واحد، ومن ثم يستدعى تابع بناء الكومة heapify على جذر الشجرة .
3. يتم تكرار الخطوة رقم 2 مادام حجم الكومة أكبر من 1 .

مثال: قم بتطبيق خوارزمية Heap sort لترتيب عناصر المصفوفة التالية :

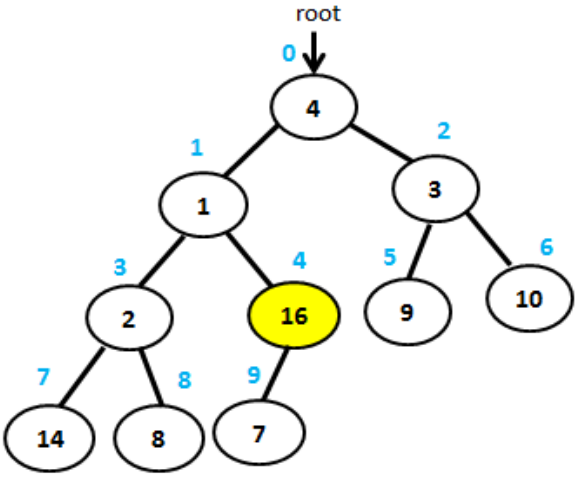
4	1	3	2	16	9	10	14	8	7

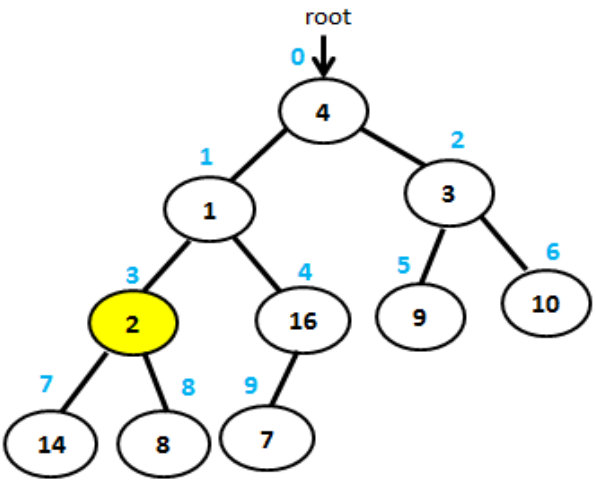
1- نقوم بتمثيل المصفوفة بشجرة ثنائية من خلال مايلي:

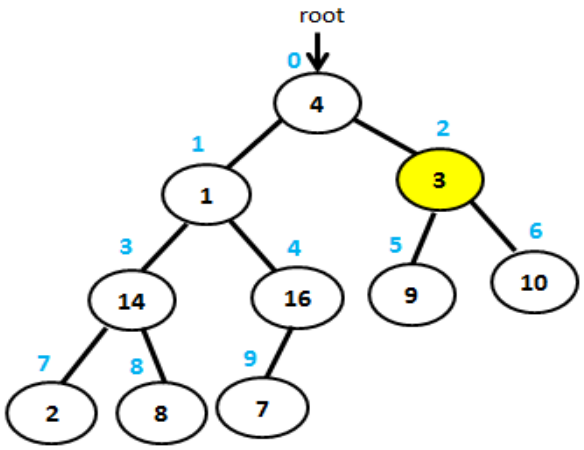
- ✓ يشكل العنصر arr[0] جذر الشجرة root .
- ✓ من أجل كل عنصر في المصفوفة ذي الدليل i يتم تمثيله بعقدة في الشجرة، و يكون العنصر من المصفوفة ذي الدليل 2i+1 هو الابن اليساري لها ، بينما يمثل العنصر من المصفوفة ذي الدليل 2i+2 الابن اليميني لها.

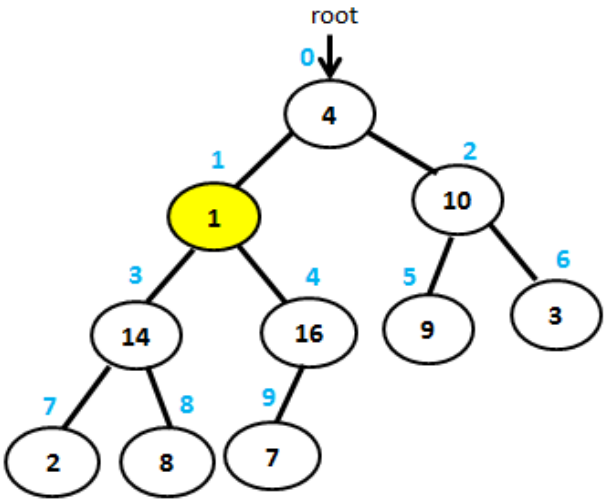


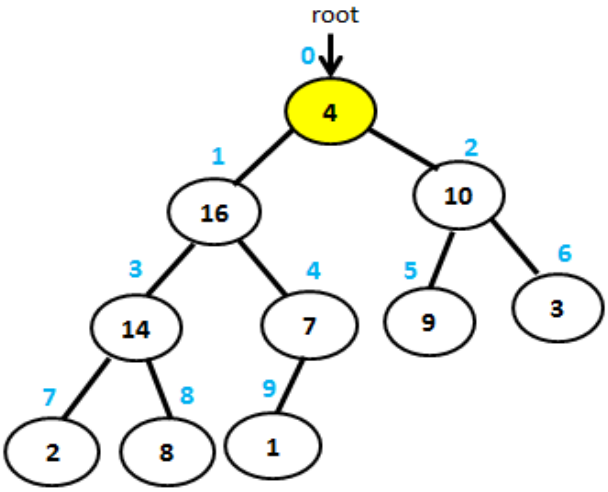
2- يتم بناء شجرة الكومة الكبرى max heap tree من خلال تطبيق تابع heapify على العقد الداخلية للشجرة(العقد التي لا تمثل أوراقاً) و يحسب دليل آخر عقدة داخلية في الشجرة بالعلاقة : $\frac{n}{2}$ - 1 حيث n عدد عقد الشجرة .

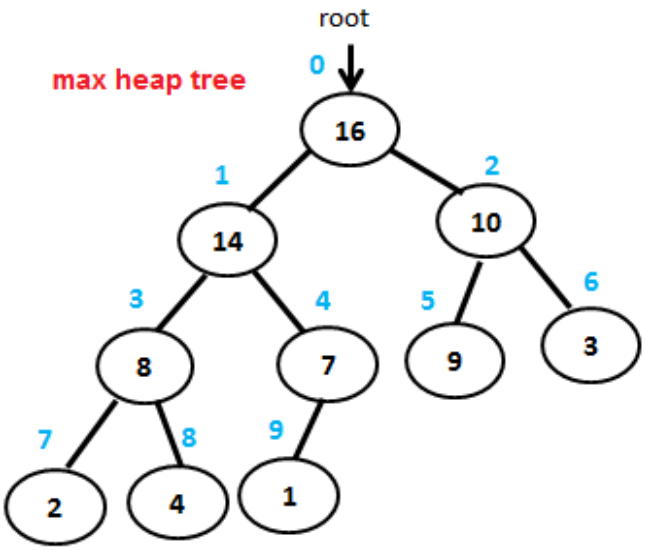




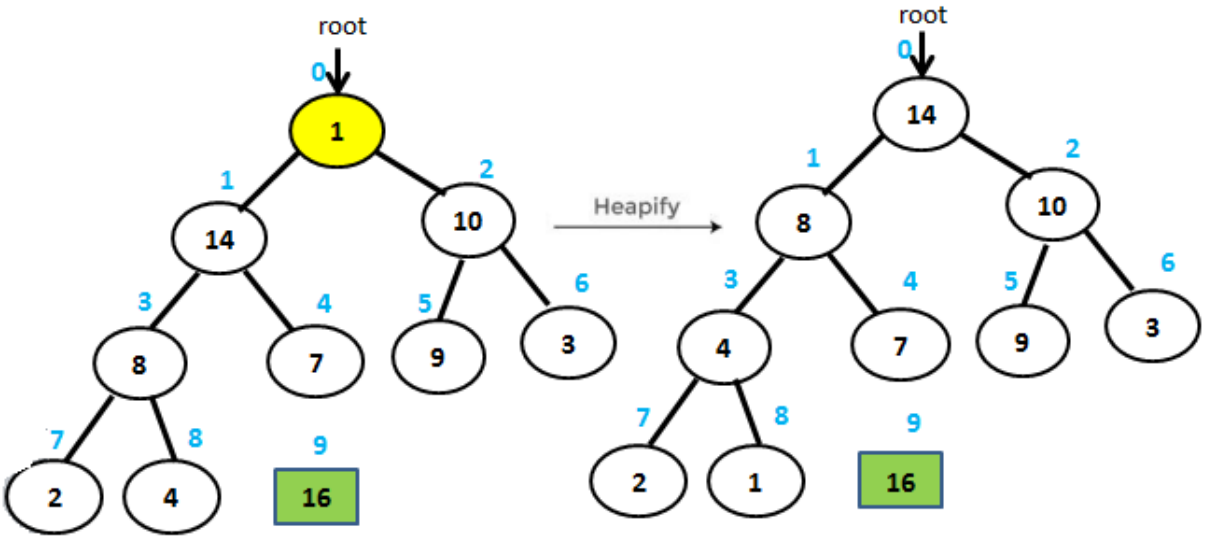

root
0
4
1
2
1
3
3
4
5
6
14
16
9
10
7
8
9
2
8
7

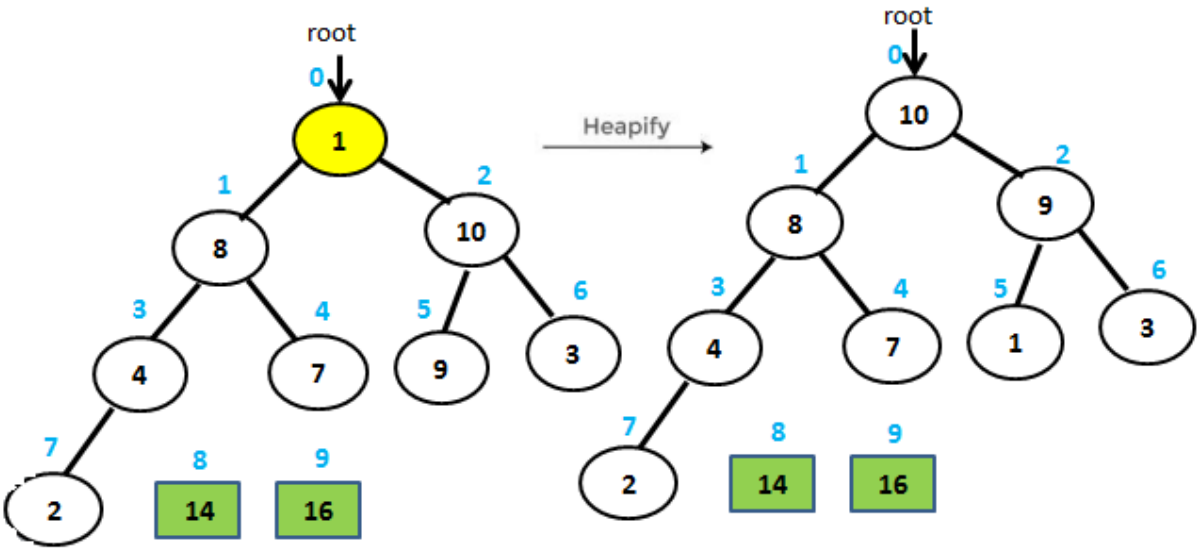

root
0
4
1
2
1
10
3
4
5
6
14
16
9
3
7
8
9
2
8
7

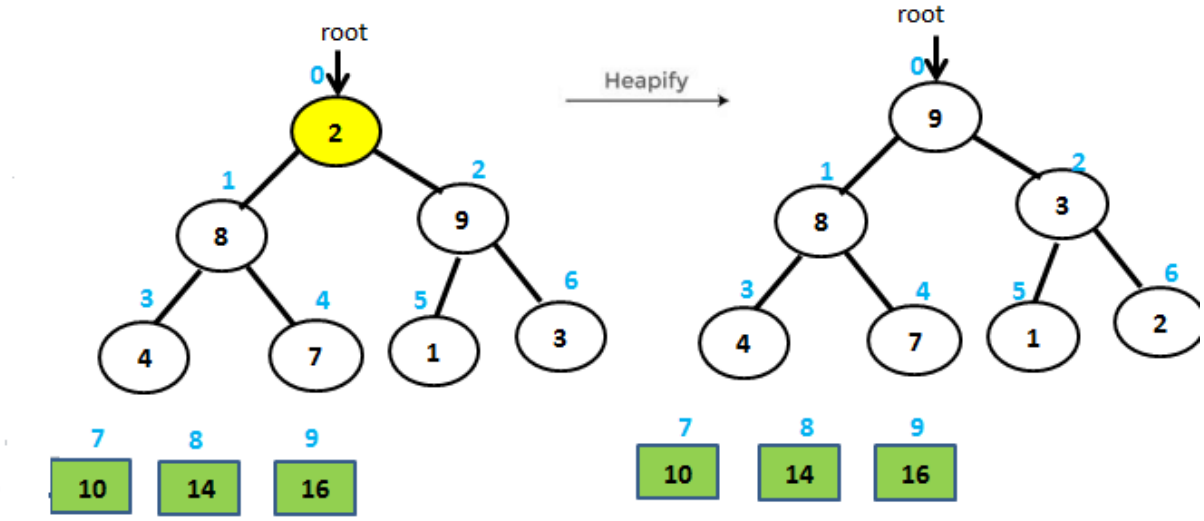

root
0
4
1
2
16
10
3
4
5
6
14
7
9
3
7
8
9
2
8
1

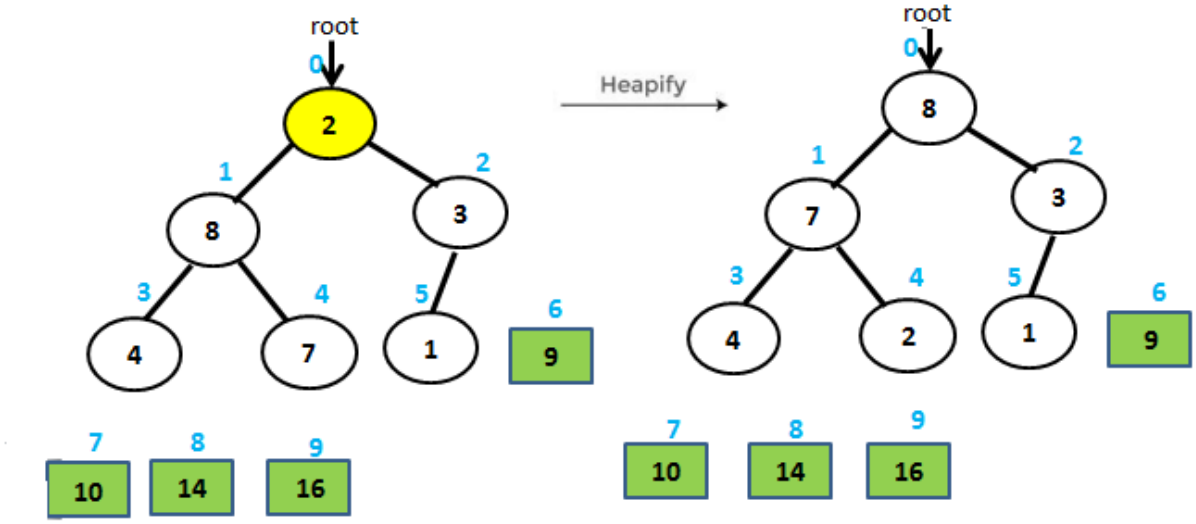


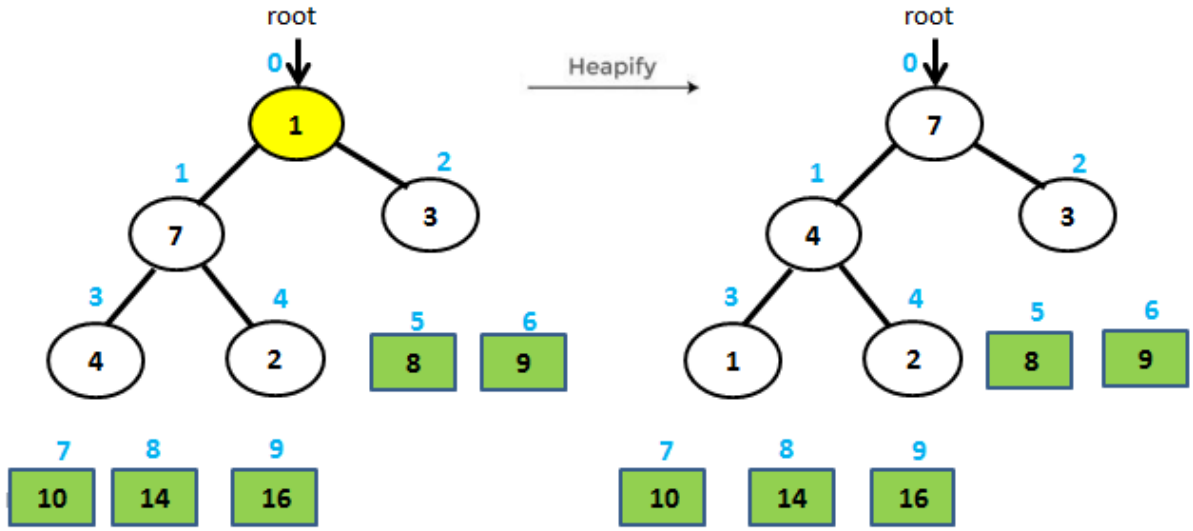
3- بعد الحصول على شجرة الكومة الكبرى فإن أكبر عنصر في الشجرة يتوضع في الجذر، لذلك يتم التبديل بين قيمة الجذر و قيمة آخر عنصر في الكومة، ثم يتم حذف العنصر الأخير من الكومة من خلال تقليل حجم الكومة بمقدار واحد، وبعد ذلك يتم استدعاء تابع heapify على جذر الشجرة لضبط خاصية الكومة ، و يتم تكرار الخطوة 3 طالما حجم الكومة أكبر من 1.

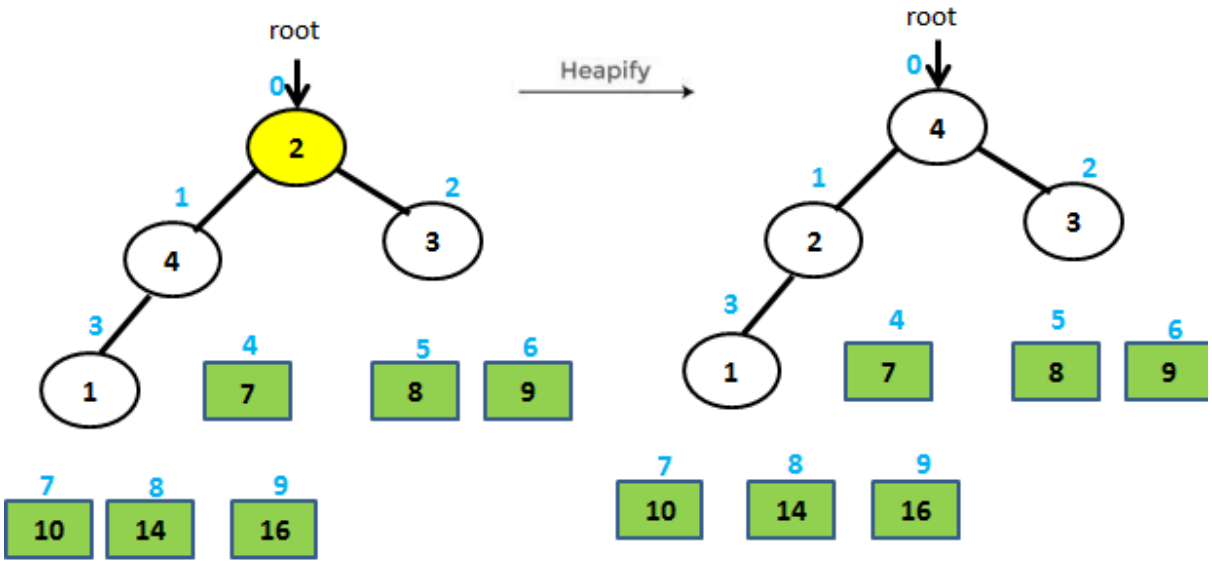


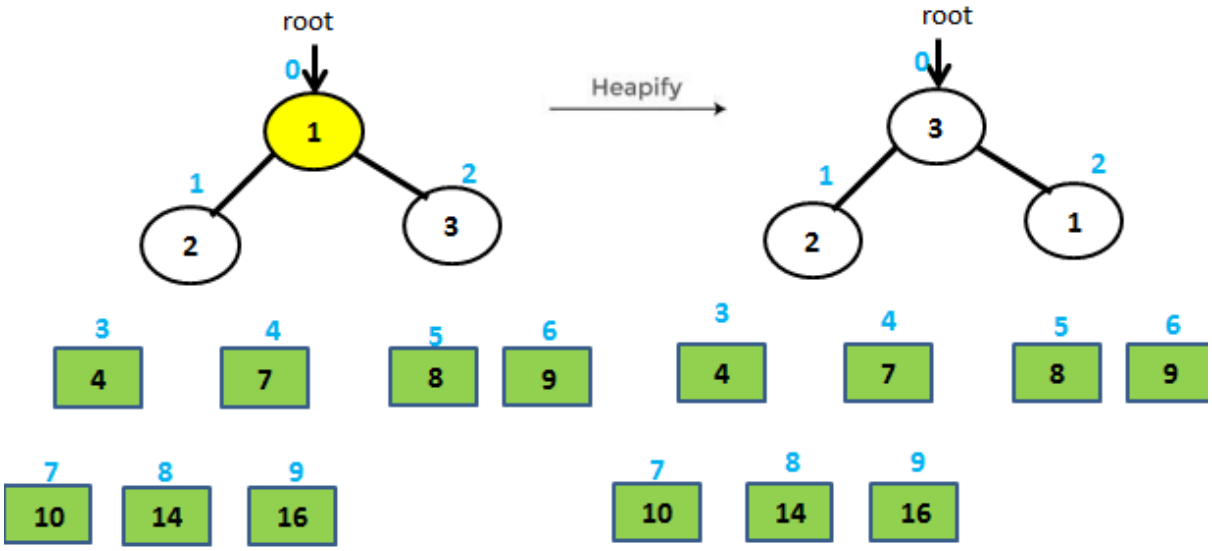

root
0
1
1
8
2
10
3
4
4
7
5
9
6
3
7
2
8
14
9
16
Heapify
root
0
10
1
8
2
9
3
4
4
7
5
1
6
3
7
2
8
14
9
16

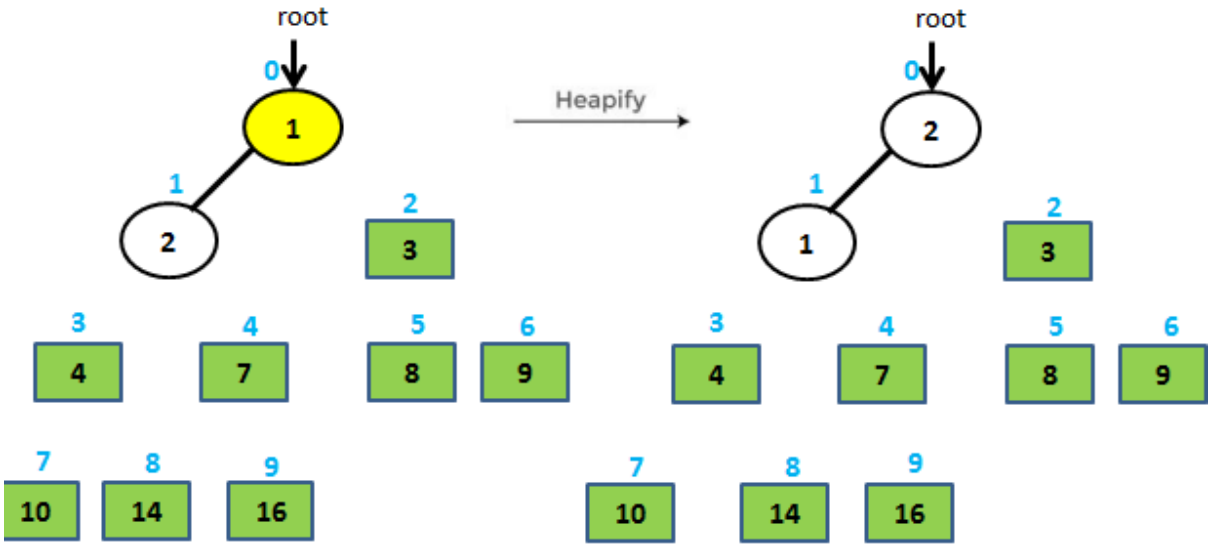

root
0
2
1
8
2
9
3
4
4
7
5
1
6
3
7
10
8
14
9
16
Heapify
root
0
9
1
8
2
3
3
4
4
7
5
1
6
2
7
10
8
14
9
16

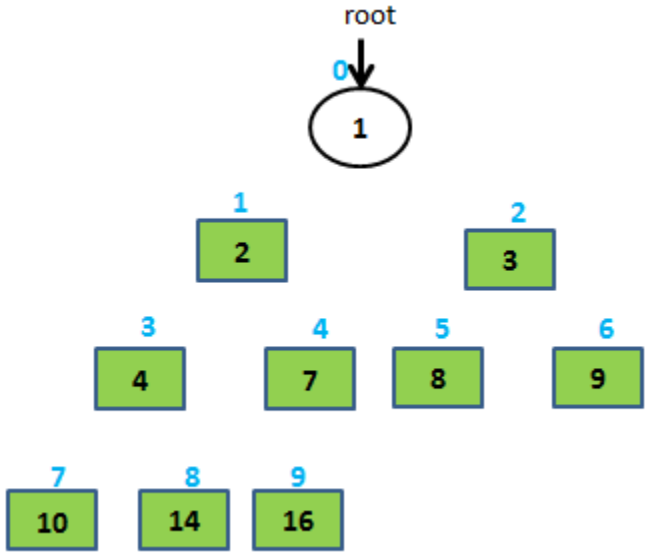

root
0
2
1
8
2
3
3
4
4
7
5
1
6
9
7
10
8
14
9
16
Heapify
root
0
8
1
7
2
3
3
4
4
2
5
1
6
9
7
10
8
14
9
16


root
0
1
1
7
2
3
3
4
4
2
5
8
6
9
7
10
8
14
9
16
Heapify
root
0
7
1
4
2
3
3
1
4
2
5
8
6
9
7
10
8
14
9
16


root
0
2
1
4
2
3
3
1
4
7
5
8
6
9
7
10
8
14
9
16
Heapify
root
0
4
1
2
2
3
3
1
4
7
5
8
6
9
7
10
8
14
9
16


root
0
1
1
2
2
3
3
4
4
7
5
8
6
9
7
10
8
14
9
16
Heapify
root
0
3
1
2
2
1
3
4
4
7
5
8
6
9
7
10
8
14
9
16





sorted array

0	1	2	3	4	5	6	7	8	9
1	2	3	4	7	8	9	10	14	16

البرنامج :

```
// C++ program for implementation of Heap Sort
#include <iostream>

using namespace std;

// To heapify a subtree rooted with node i which is
// an index in arr[]. n is size of heap
void heapify(int arr[], int n, int i)
{
    int largest = i; // Initialize largest as root
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    // If left child is larger than root
    if (left < n && arr[left] > arr[largest])
        largest = left;
        // If right child is larger than largest so far
    if (right < n && arr[right] > arr[largest])
        largest = right;

    // If largest is not root
    if (largest != i) {
        swap(arr[i], arr[largest]);

        // Recursively heapify the affected sub-tree
        heapify(arr, n, largest);
    }
}

// main function to do heap sort
void heapSort(int arr[], int n)
{
    // Build heap (rearrange array)
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // One by one extract an element from heap
    for (int i = n - 1; i > 0; i--) {
        // Move current root to end
        swap(arr[0], arr[i]);

        // call max heapify on the reduced heap
        heapify(arr, i, 0);
    }
}
```

```
/* A utility function to print array of size n */
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; ++i)
        cout << arr[i] << " ";
    cout << "\n";
}

// Driver code
int main()
{
      int n, i;
  cout<<"\nEnter the number of data elements to be sorted: ";
    cin>>n;
  int arr[100];
  for(i = 0; i < n; i++)
  {
    cout<<"Enter element "<<i+1<<": ";
    cin>>arr[i];
  }

    heapSort(arr, n);

    cout << "Sorted array is \n";
    printArray(arr, n);
}
```

تمرين غير محلول :

طبق كل من خوارزميتي الترتيب بالدمج و الترتيب بالكومة لترتيب المصفوفة التالية :

81	89	9	11	14	76	54	22