NCKU Programming Contest Training Course
Heap
2013/04/10

Pinchieh Huang (free999)

Pinchieh.huang@gmail.com

http://myweb.ncku.edu.tw/~p76014143/20130410_Heap.rar

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

Example

題目要你做的任務：把一些數加起來。但是這對你來說一定是太簡單了，所以讓我們加一些東西在裡面。

做加法要付出的代價（cost）定義為這2個數的總和，所以要加 1 和 10 所需付出的代價為 11 。假如你想要加 1, 2 和 3，那麼有以下幾種方法：

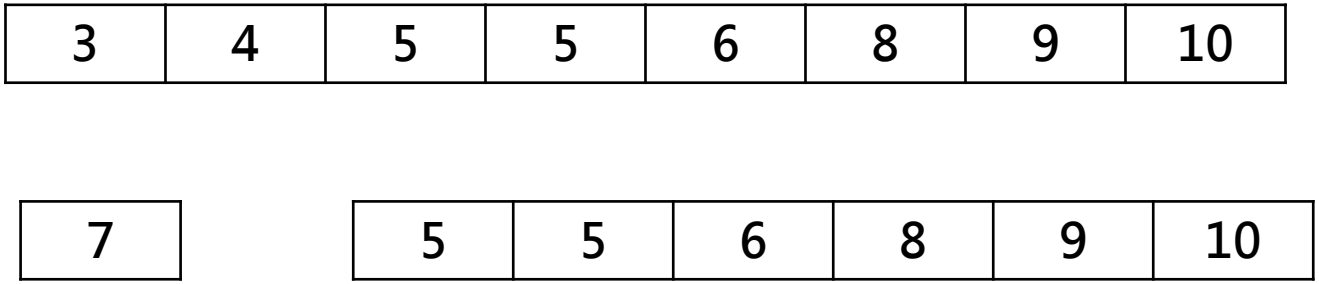
1 + 2 = 3, cost = 3
3 + 3 = 6, cost = 6
Total = 9

1 + 3 = 4, cost = 4
2 + 4 = 6, cost = 6
Total = 10

2 + 3 = 5, cost = 5
1 + 5 = 6, cost = 6
Total = 11

我希望你已經瞭解你的任務，就是把 N 個數加起來使得付出的代價最小。

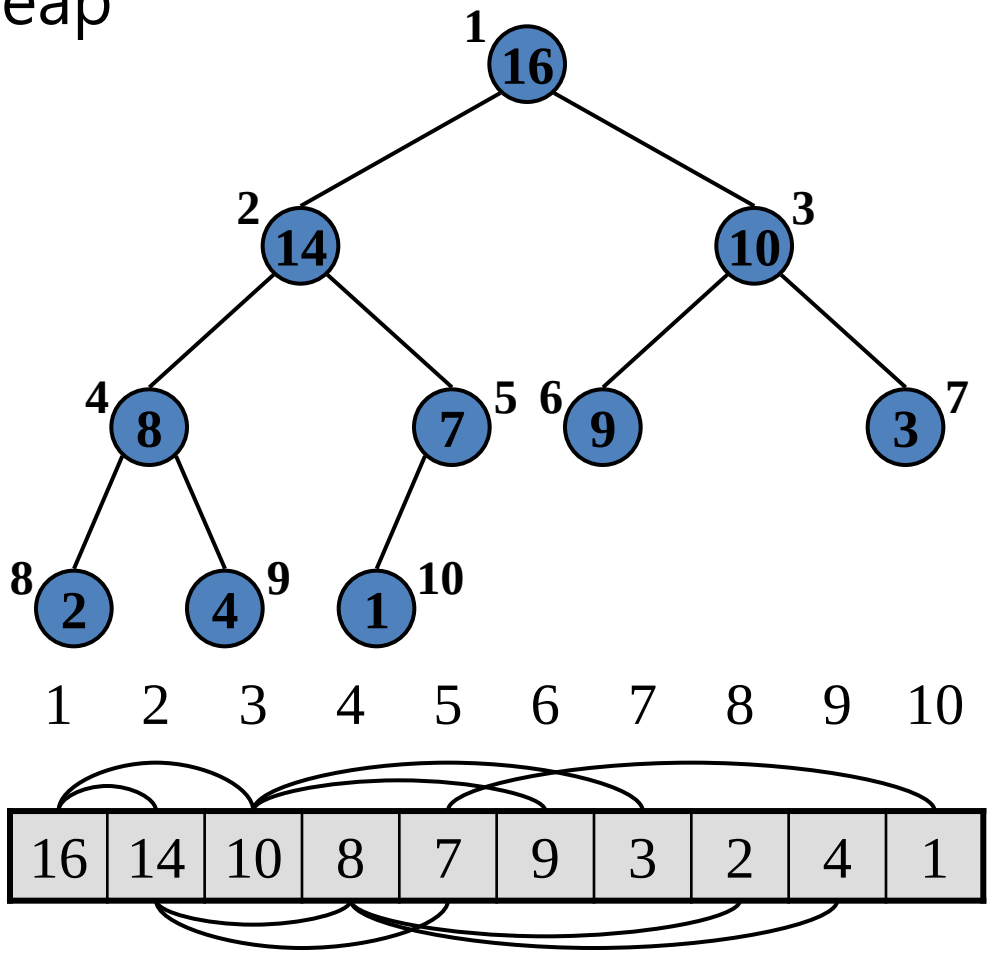
Example



SORT ?

Heap

- A max-heap



made by electron & free999

Heap

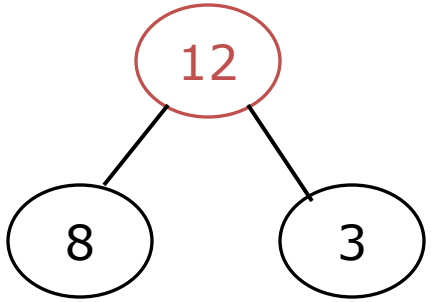
- Heap property
  - For max-heap (largest element at root), *max-heap property*: for all nodes i, excluding the root, $A[\text{PARENT}(i)] \geqq A[i]$.
  - For min-heap (smallest element at root), *min-heap property*: for all nodes i, excluding the root, $A[\text{PARENT}(i)] \leqq A[i]$.

- Maximum element of a max-heap is at the root.

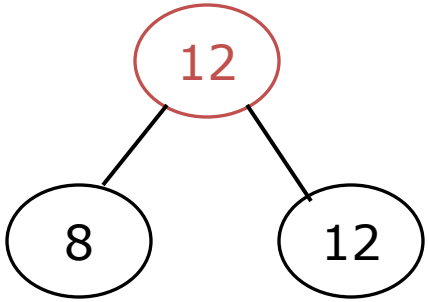
- The heap sort algorithm we' ll shoe uses max-heaps.

Heap

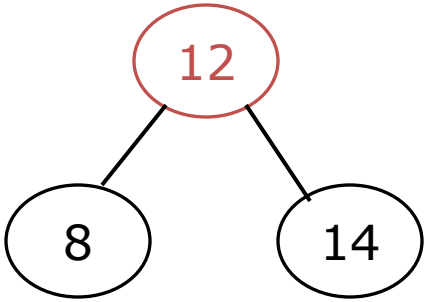
- Heap property



Red node has heap property



Red node has heap property



Red node does not have heap property

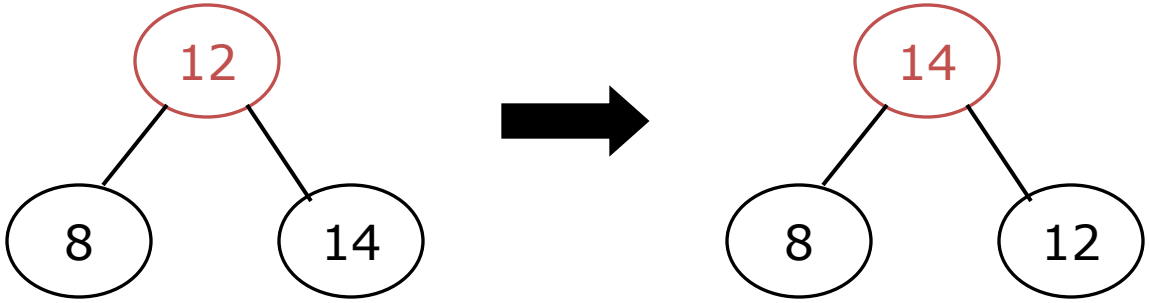
Heap

- Heap subroutine
  - heapify
    - adjust one path of the heap tree
  - build-heap
    - adjust the total heap tree

made by electron & free999

Heap

- Heapify



Blue node does not have heap property

Blue node has heap property

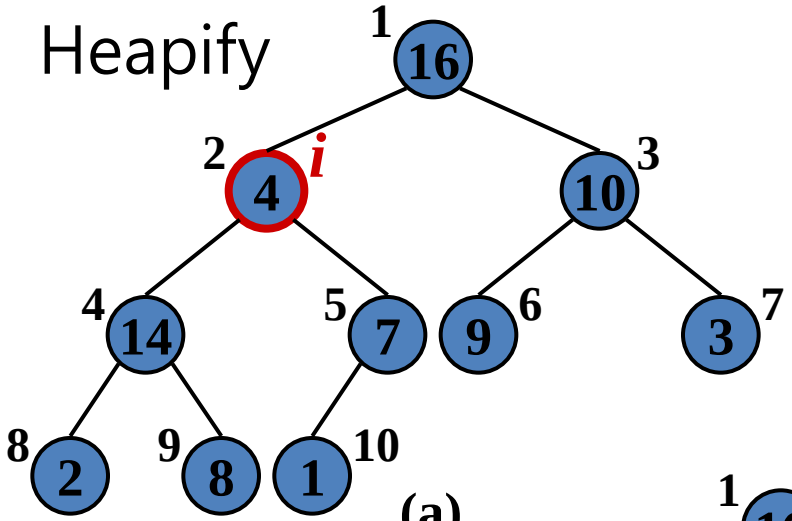
Heap

MAX-HEAPIFY(A, i, n)

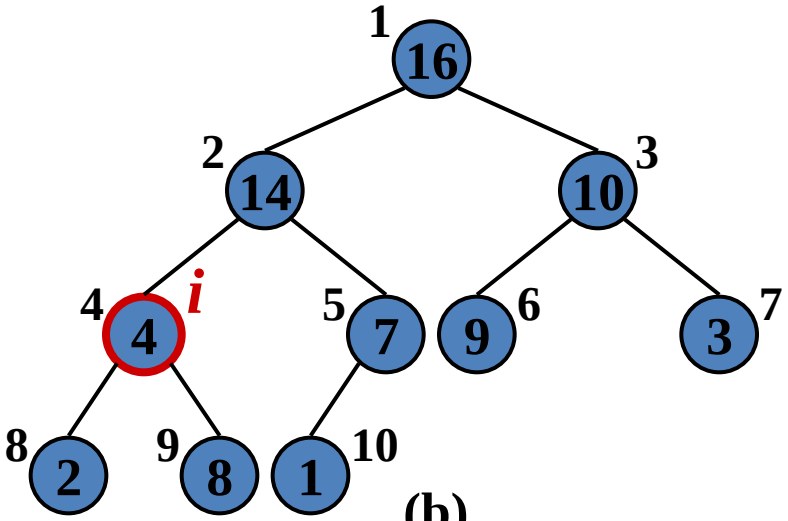
1. $l \leftarrow$ LEFT(i)
2. $r \leftarrow$ RIGHT(i)
3. **if** $l \leq n$ and $A[l] > A[i]$
4. **then** $largest \leftarrow l$
5. **else** $largest \leftarrow i$
6. **if** $r \leq n$ and $A[r] > A[largest]$
7. **then** $largest \leftarrow r$
8. **if** $largest \neq i$
9. **then** exchange $A[i] \leftrightarrow A[largest]$
10. MAX-HEAPIFY(A, $largest$, n)

Heap

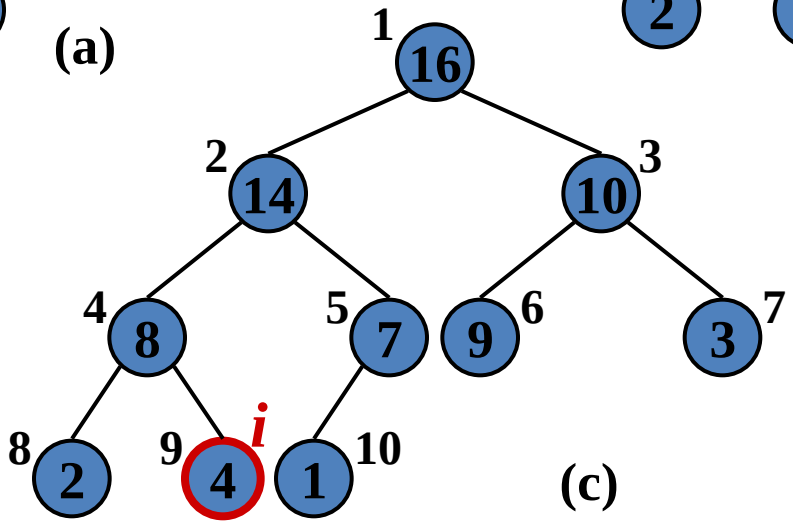
- Heapify



(a)



(b)

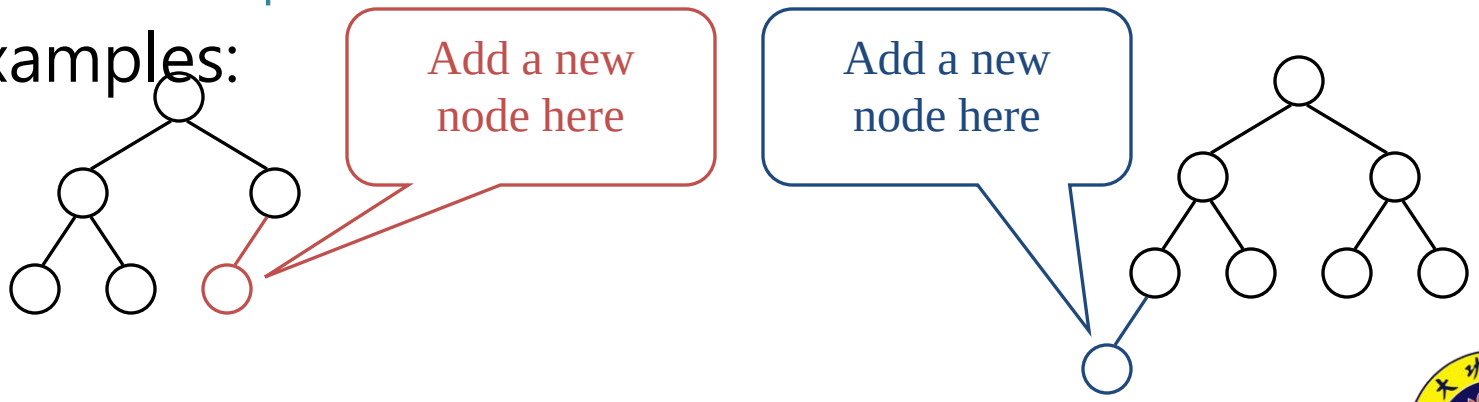


(c)

made by electron & free999

Heap

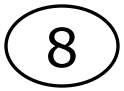
- Construct the heap from a tree
- A tree consisting of a single node is automatically a heap
- We construct a heap by adding nodes one at a time:
  - Add the node just to the right of the rightmost node in the deepest level
  - If the deepest level is full, start a new level
- Examples:



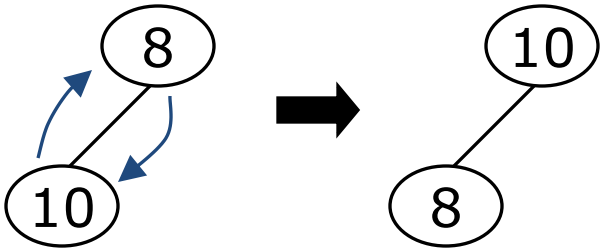
made by electron & free999

Heap

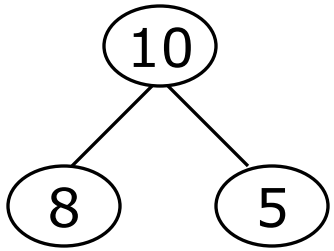
- Construct the heap from a tree



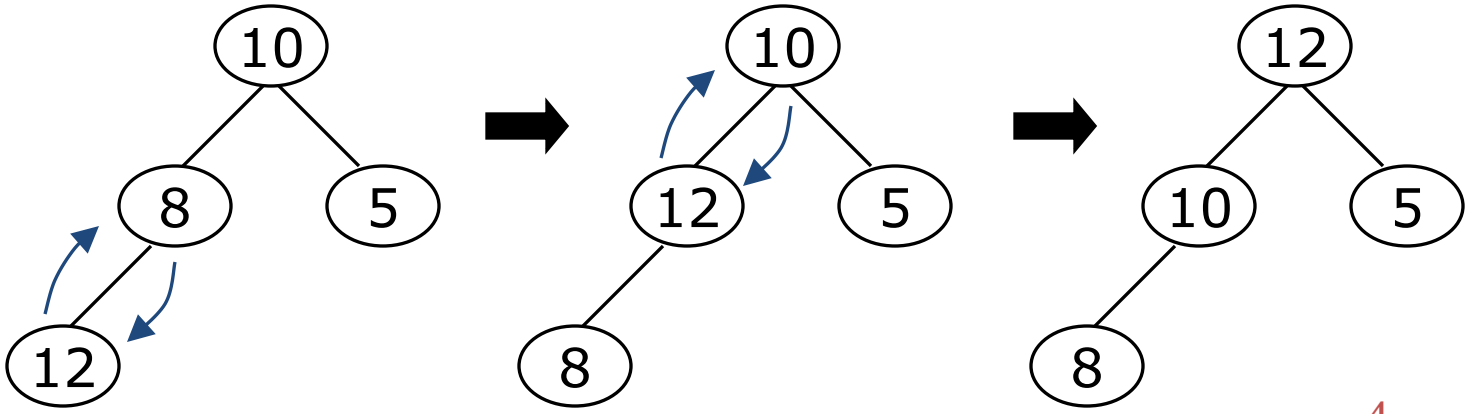
1



2



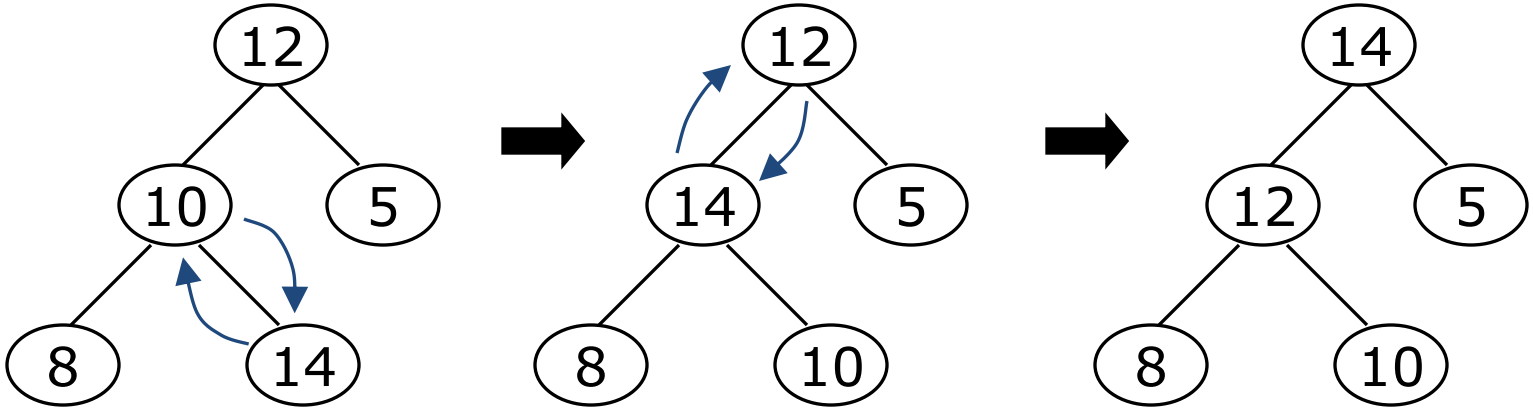
3



4

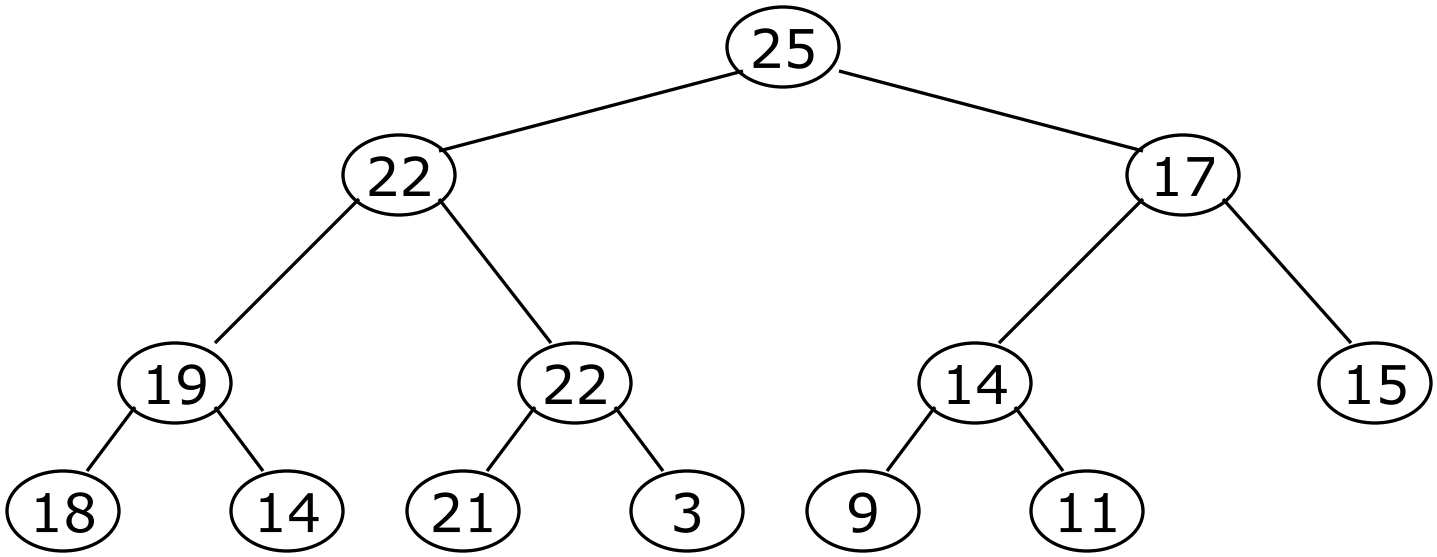
Heap

- Construct the heap from a tree



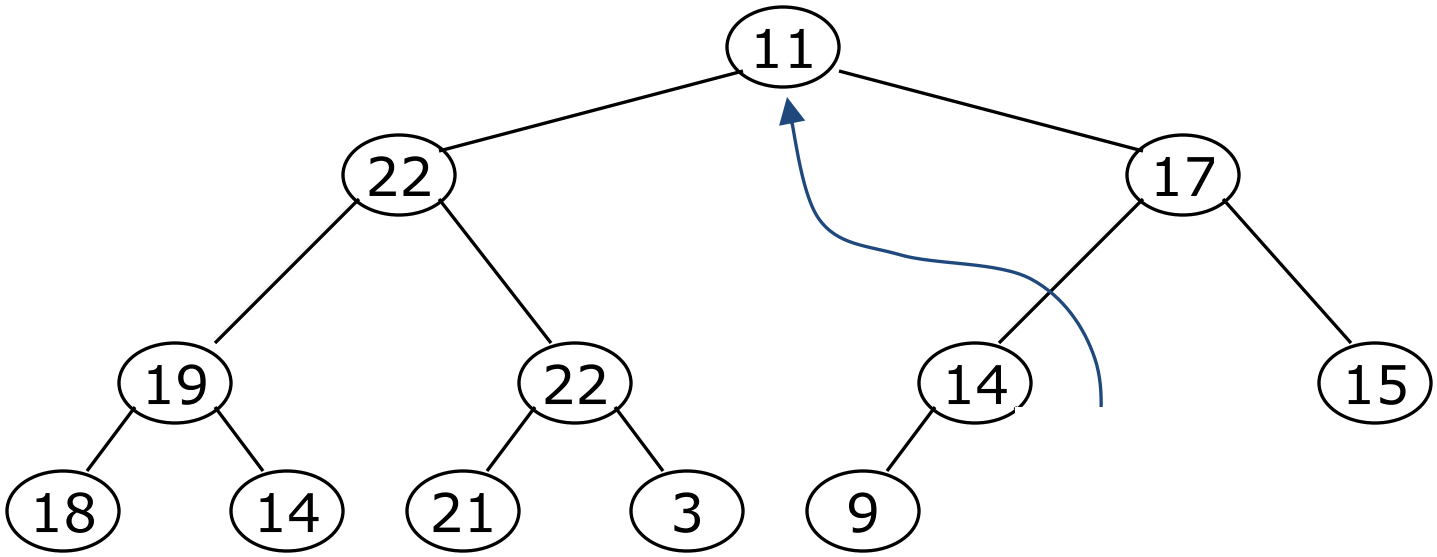
Heap

- Here’ s a sample binary tree after it has been heapified



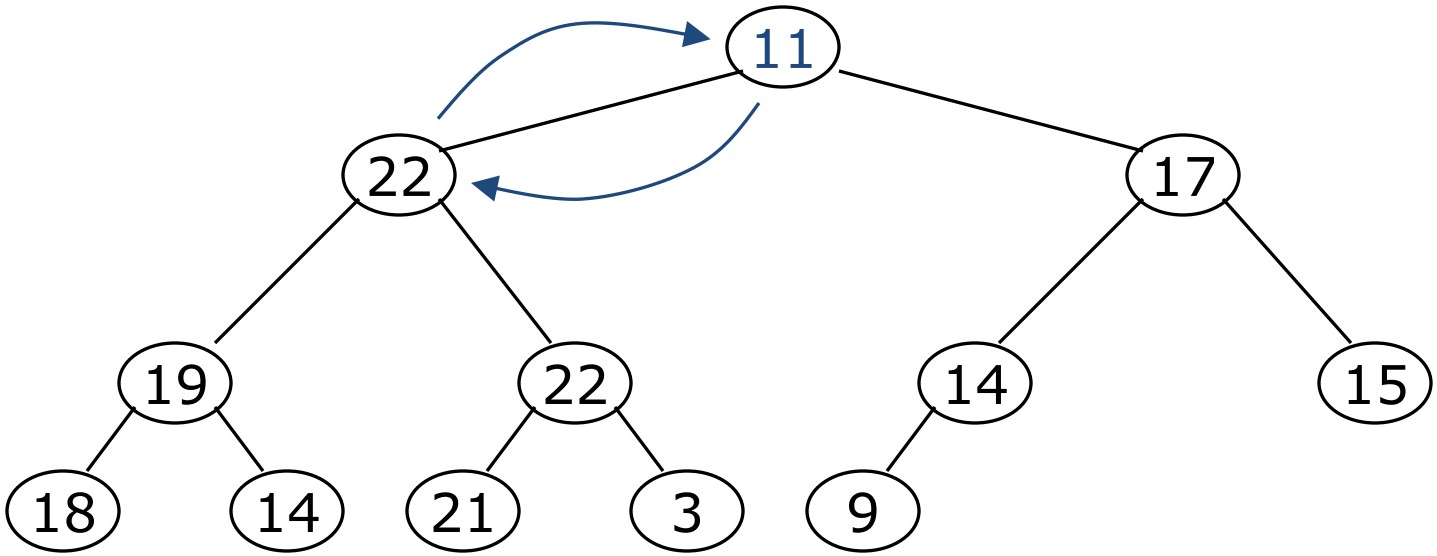
Heap

- Notice that the largest number is now in the root
- Suppose we *discard* the root:



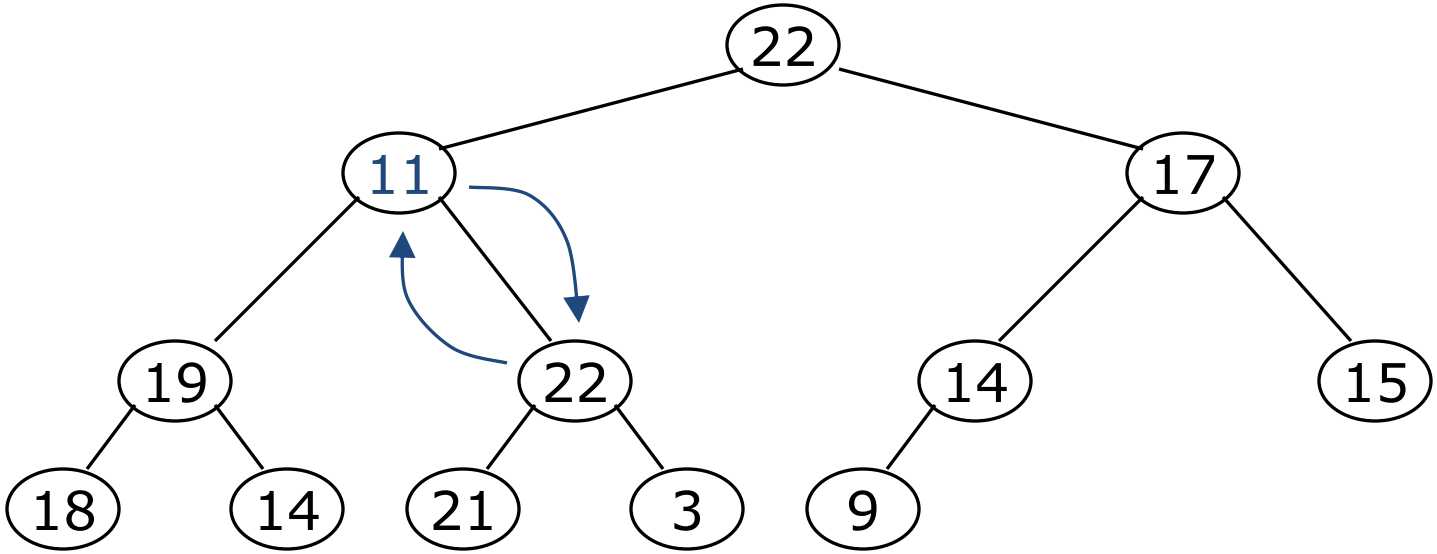
Heap

- Heapify from the root



Heap

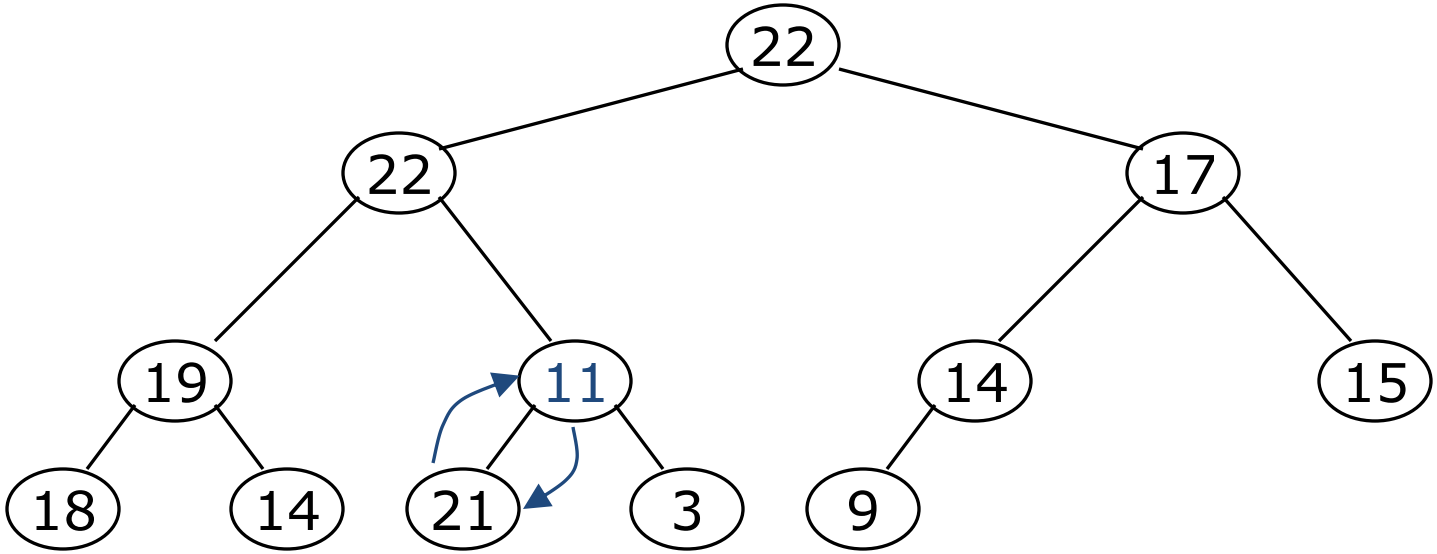
- Heapify from the root



made by electron & free999

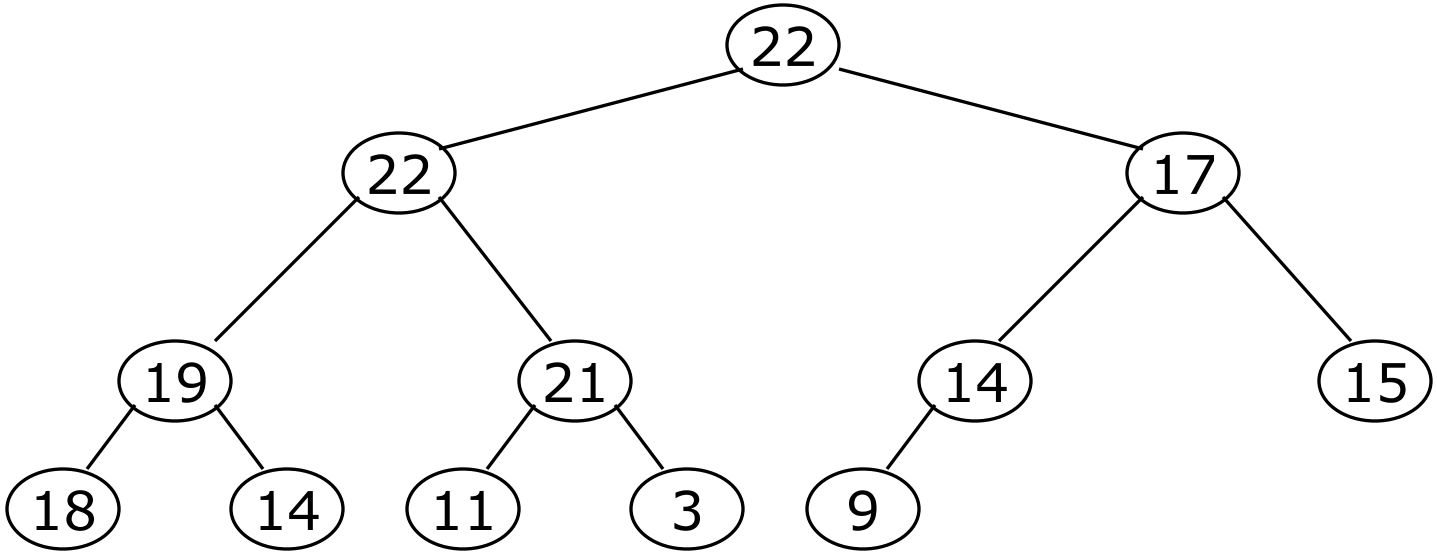
Heap

- Heapify from the root



Heap

- Heapify from the root



made by electron & free999

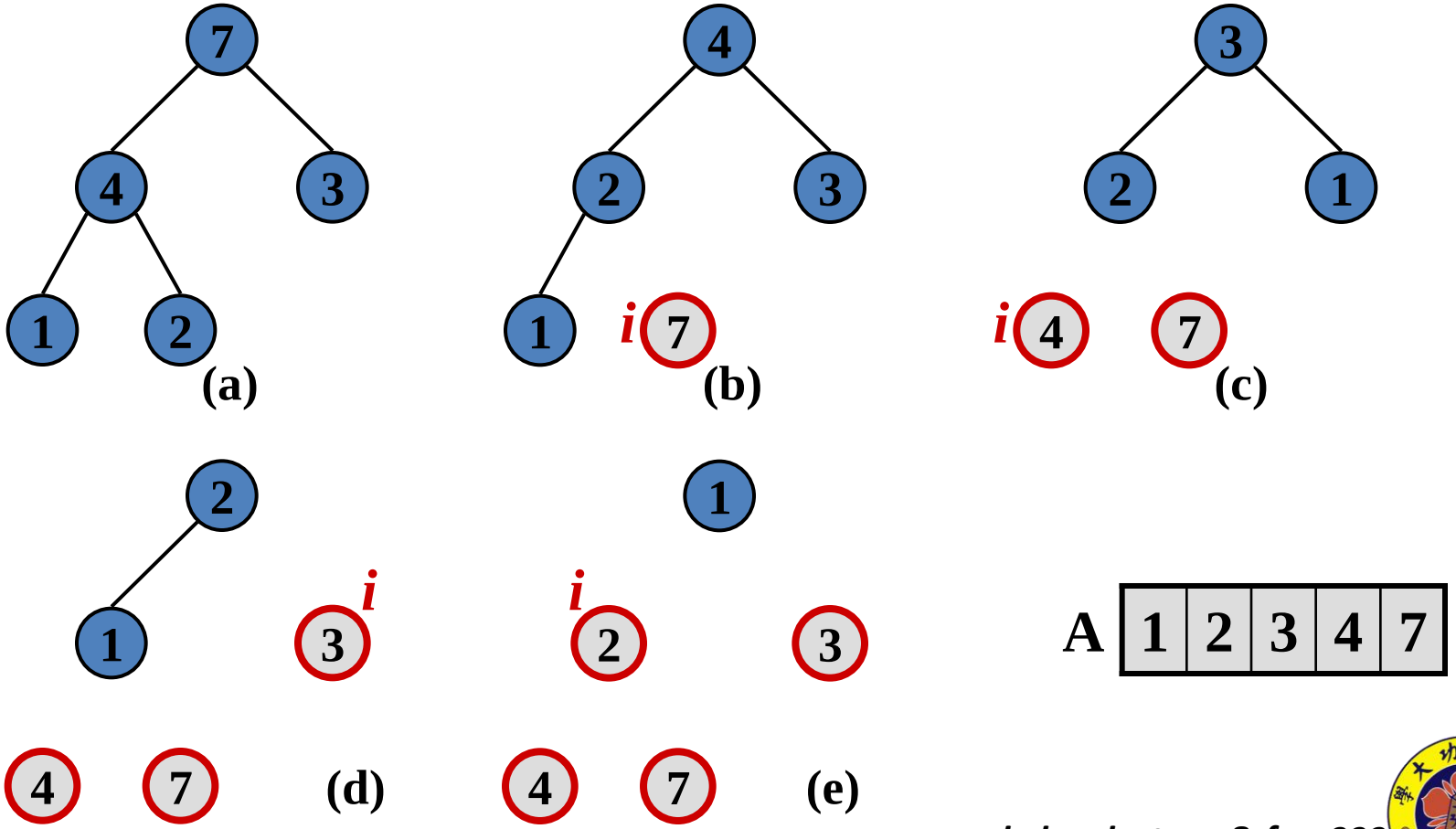
Heap

- Heap sort
  - pop the top element
  - pick the bottom-right most leaf
  - heapify from the root

1. BUILD-MAX-HEAP(A, n)
2. **for** $i \leftarrow n$ **downto** 2
3. **do** exchange $A[1] \leftrightarrow A[i]$
4. MAX-HEAPIFY(A, 1, $i-1$)

Heap

- Heap sort



Priority Queue

- **Heap implementation of priority queue**
  - Max-priority queues are implemented with max-heaps. Min-priority queues are implemented with min-heaps similarly.
- **Max Priority Queues**
  - Maintains a dynamic set of S of elements.
  - Each set element has a key: an associated value.
  - Max-priority queue supports dynamic-set operations:
    - INSERT(S, x): inserts element of S with largest key.
    - MAXIMUM(S): returns elements of S with largest key.
    - EXTRACT-MAX(S): removes and returns element of S with largest key.
    - INCREASE-KEY(S, x, k): increases value of element x' s key to k. Assume $k \geq x$' s current key value.

Priority Queue

- **Finding the maximum element**
  - Getting the maximum element is easy: it’ s the root.

  HEAP-MAXIMUM(*A*)

  1. **return** *A*[1]
  - *Time:* $\Theta(1)$
- **Extracting max element**

  Given the array A:
  - Make sure heap is not empty
  - Make a copy of the maximum element (the root).
  - Make the last node in the tree the new root.
  - Re-heapify the heap, with one fewer node.
  - Return the copy of the maximum element.

Priority Queue

HEAP-EXTRACT-MAX(A, n)

1. **if** $n < 1$
2. **then error** "heap underflow"
3. $max \leftarrow A[1]$
4. $A[1] \leftarrow A[n]$
5. MAX-HEAPIFY(A, 1, $n-1$) ▸ remakes heap
6. **return** max

C++ STL PQ

- STL Priority Queue

```
#include <queue>
struct NODE
{
    int v;
    bool operator < (const NODE &in) const{
        return v < in.v;
        // max heap, just remember this section of codes
    }                          //由小排到大  底是小 頂是大
};
```

C++ STL PQ

- STL Priority Queue

```
priority_queue<NODE> pq;
NODE tmp;
tmp.v = 10; pq.push (tmp);        // push
tmp.v = 1;    pq.push (tmp);
tmp.v = 5;    pq.push (tmp);
while(!pq.empty())
{
            // get the node in the top of max heap
    tmp = pq.top();
    printf("%d ", tmp.v );
    pq.pop();                  // pop off the top node
}
10 5 1                  // stdout
```

Example

Uva 10954 Add All

Yup!! The problem name reflects your task; just add a set of numbers. But you may feel yourselves condescended, to write a C/C++ program just to add a set of numbers. Such a problem will simply question your erudition. So, let’ s add some flavor of ingenuity to it.

Addition operation requires cost now, and the cost is the summation of those two to be added. So, to add **1** and **10**, you need a cost of **11**. If you want to add **1**, **2** and **3**. There are several ways –

Example

Uva 10954 Add All

1 + 2 = 3, cost = 3
3 + 3 = 6, cost = 6
Total = 9
1 + 3 = 4, cost = 4
2 + 4 = 6, cost = 6
Total = 10
2 + 3 = 5, cost = 5
1 + 5 = 6, cost = 6
Total = 11

I hope you have understood already your mission, to add a set of integers so that the cost is minimal.

Example

Uva 10954 Add All

Input

Each test case will start with a positive number, **N (2 ≤ N ≤ 5000)** followed by **N** positive integers (all are less than **100000**). Input is terminated by a case where the value of **N** is zero. This case should not be processed.

Output

For each case print the minimum total cost of addition in a single line.

Sample Input	Output for Sample
3 1 2 3 4 1 2 3 4 0	9 19

Homework

- UVA (total 3 problems)
  - 10954、501、11995
- POJ (total 3 problems)
  - 2431、3345、2442

made by electron & free999

Thank You For Attention!

made by electron & free999