Not optimization

Recursion
- Substitution
- Master Method
- Recurrence Tree

Divide & Conquer –

Sub problem

$\leftarrow T(n) = aT(n/b) + f(n)$

Greedy Method   Not Recursive

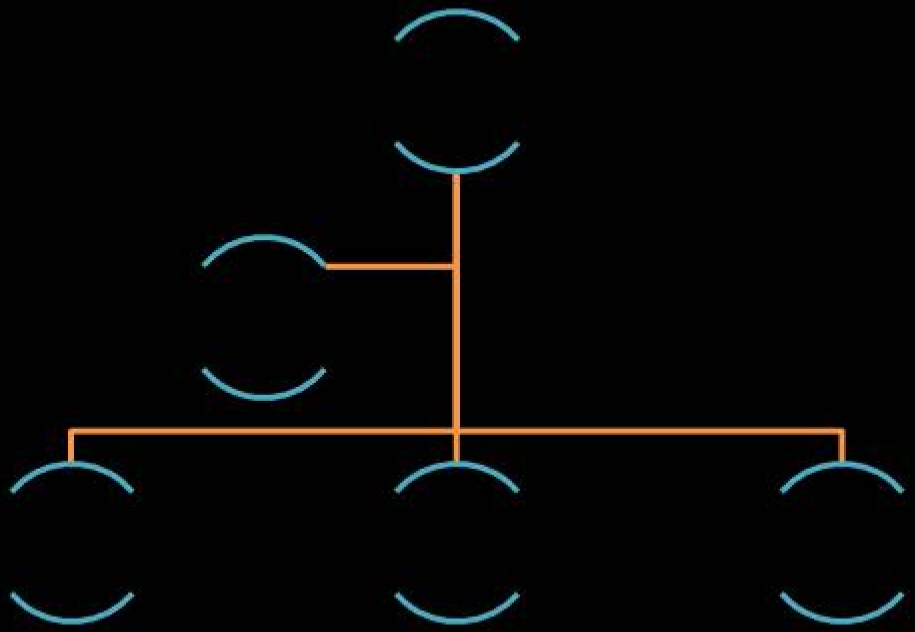
1. optimization

pick one solution

Depends upon the problem $\Leftarrow$ • ordering will be performed

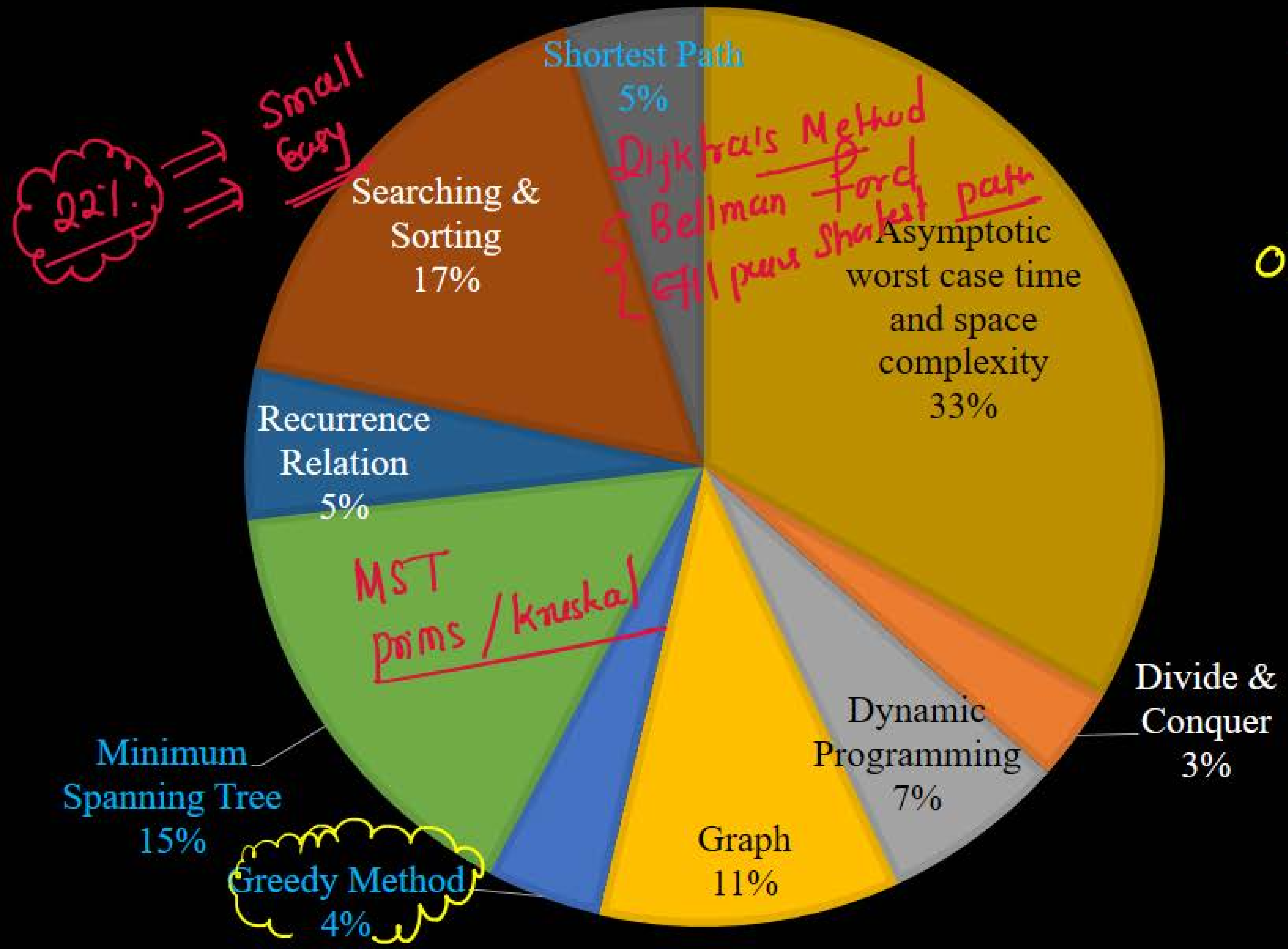
~~See~~ Select

2. Fesible : "Satisfy some constraints"



Greedy Method

Greedy Algorithm Weightage Analysis



- Maximization – profit
- Minimization : cost. time

Objective function

1. Greedy Method
   1. knapsack problem
   2. Job sequencing with deadline

- Data structure

that allows to select min or max. (Heap) Binary Heap

Greedy Method

1. Objective function
2. Constraints
3. fesible Solution
4. optimal Solution

Greedy Method

- The greedy method is perhaps the most straightforward design technique we consider in this text, and what's more it can be applied to a wide variety of problems. Most, though not all, of these problems have n inputs and require us to obtain a subset that satisfies some constraints.

Greedy Method

- **Feasible Solution:**

Bag capacity - 15 , Only M/c is available

Only one path is available.

only 9 months to prepare for examination

- Every Solution that ~~sasts~~ satisfied the constraints is called fesible Solution

Greedy Method

- **Feasible Solution:** Any subset that satisfies these constraints is called a *feasible* solution.

Greedy Method

- **Objective function:**

(Differ from one problem to another)

By designing algorithm what exactly we want to achieve.

Maximizing objective function (e.g. profit)

Minimize objective function (e.g. cost)

Greedy Method

- **Objective function:** We need to find a feasible solution that either maximizes or minimizes a given *objective function*.

Greedy Method

15 — (4)

capacity object

- **Optimal solution**:

Every fesible solution satisfied the constraints

among fesible an optimal solution is the solution that maximizes or minimizes the objective function

Greedy Method

- **Optimal solution:** A feasible solution that does this is called an *optimal solution*. There is usually an obvious way to determine a feasible solution but not necessarily an optimal solution.

Greedy Method

- **Selection procedure:** :- Selection procedure is way to first order the given element based on objective function then Select the Solution & check whether the give solution is fesible or not. if feasible then put them in the Solution set

Greedy Method

- **Selection procedure:**

Greedy Method

- **Selection procedure**: *This is done by considering the inputs in an order determined by some selection procedure.*

Greedy Method

- **Selection procedure:** The greedy method suggests that one can devise an algorithm that works in stages, considering one input at a time. At each stage, a decision is made regarding whether a particular input is in an optimal solution. *This is done by considering the inputs in an order determined by some selection procedure.*

Greedy Method

- If the inclusion of the next input into the partially constructed optimal solution will result in an infeasible solution, then this input is not added to the partial solution. Otherwise, it is added. The selection procedure itself is based on some optimization measure.

Greedy Method

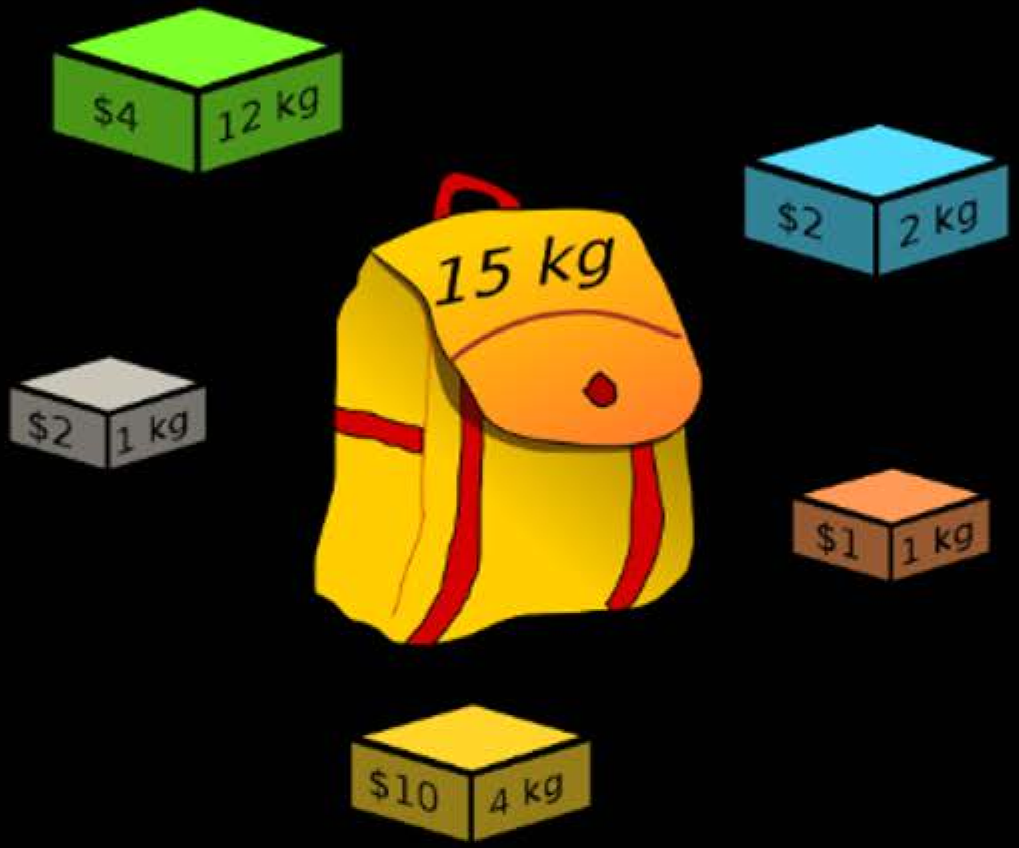
This measure may be the objective function. In fact, several different optimization measures may be plausible for a given problem. Most of these, however, will result in algorithms that generate suboptimal solutions. This version of the greedy technique is called the *subset paradigm*.

Greedy Method Control Abstraction

Greedy Method Control Abstraction

```
Algorithm GreedyMethod (a, n) {        find subset of n
// a is an array of n inputs
 Solution: =Ø;
 for i: =1 to n do{                            Ordering
  s: = select (a);
if (feasible (Solution, s)) then {
Solution: = union (Solution, s);
}
else
reject (); // if solution is not feasible reject it.
}
return solution;}
```

- optimal
- fesible
- objective function
- constraints
- Selection criteria



Knapsack Problem

Knapsack Problem

fractional

We are given a bag/knapsack with capacity of M.

We are given n objects where each object i is associated with profit p_i and weight w_i.

objective function: if an object i with capacity w_i is filled in the bag profit p_i is earned.

If fraction x_i of the object is ~~don~~ added then profit ~~wi~~ $p_i x_i$ will be earned $0 \le x_i \le 1$

objective function: find filling of knapsack so the maximum profit is earned.

Knapsack Problem

- **Problem statement:** Let us try to apply the greedy method to solve the knapsack problem. We are given n objects and a knapsack or bag. Object i has a weight w_i and the knapsack has a capacity m. If a fraction x_i, $0 \leq x \leq 1$, of object i is placed into the knapsack, then a profit of $p_i x_i$ is earned.

Knapsack Problem

- ***Objective function:***

Knapsack Problem

- ***Objective function:*** The objective is to obtain a filling of the knapsack that maximizes the total profit earned. Since the knapsack capacity is m, we require the total weight of all chosen objects to be at most m.

Constraints $\sum_{1 \le i \le n} x_i w_i \le M$

Knapsack Problem

- Maximize $\sum_{1 \leq i \leq n} p_i x_i$
- subject to $\sum_{1 \leq i \leq n} w_i x_i \leq M$
- $0 \leq x_i \leq 1, 1 \leq i \leq n$

Knapsack Problem

A feasible solution

An optimal solution

Knapsack Problem

A feasible solution (or filling) is any set (x1... ,xn) satisfying constraints and condition above. An optimal solution is a feasible solution for which maximized the objective function.

(0, 1, 1) – Not fesible
(1, 1, 1) – Not feasible

Example

Example-1 Knapsack capacity is m = 20, find the filling of the bag that maximizes the profit with following data given

$n = 3$, m = 20, $(p_1, p_2, p_3) = (25, 24, 15)$, and $(w_1, w_2, w_3) = (18, 15, 10)$.

profit Vector

weight Vector

(1) (1/2)

	1	2	3
wi	18	15	10
Pi	25	24	15
Pi/wi	1.38	1.6	1.5

Greedy about profit

25

wi 18 2

profit/weight $\frac{24}{15} \times 2$

$48/15 = 3.2 \Rightarrow 28.2$

Greedy about weight

profit per unit weight
arrang the object in
that order, 2, 3, 1
Select - 2 - profit $\boxed{24}$
Select - 3 m = 5
1/2 15 = 7.5 total = 31.5

Example

Example-2 Knapsack capacity is m = 15, find the filling of the bag that maximizes the profit

	(1)	(2/3)	(1)		(1)	(1)	(1)
Object No.	1	2	3	4	5	6	7
Profit $-p_i$	10	5	15	7	6	18	3
Weight-w_i	2	3	5	7	1	4	1
profit/weight	5	5/3 = 1.66	3	1	6	4.5	3

fesible

Objective-function

Constraints

profit - 6
weight - 14

profit - 6+10
weight - 12

profit - 6+10+18
weight - 8

profit - 6+10+18+15
3

profit 6+10+18+15+3
weight - 2

profit = 6+10+18+15+3+10/3 = $\boxed{55.33}$
weight (0)

3 — 5 — 5/3

∴ 2 — 10/3

Answer

Example-2 Knapsack capacity is m = 15, find the filling of the bag that maximizes the profit

Object	1	2	3	4	5	6	7
Profit –p_i	10	5	15	7	6	18	3
Weight-w_i	2	3	5	7	1	4	1

Answer 55.33

Job task

Sequencing – How job will be completed in order

your production manager

- There are n jobs given, & with each job j_i the profit p_i and deadline d_i is associated.
- The profit p_i is earned if job j_i completing by its deadline.
- To complete the job one has to Run the job on a m/c for 1 time unit

Only one m/c is available

Job Sequencing with Deadline

Problem Statement

Maximum profit earned

Problem Statement

- We are given a set of n jobs. Associated with job i is an integer deadline $di \geq 0$ and a profit $pi > 0$.
- For any job i the profit pi is earned iff the job is completed by its deadline.

Problem Statement

- How to Complete a job:

Problem Statement

- To complete a job, one has to process the job on a machine for one unit of time. Only one machine is available for processing jobs-
- A feasible solution for this problem is a subset *J* of jobs such that each job in this subset can be completed by its deadline.

Problem Statement

- Value of feasible solution

- An optimal solution

Problem Statement

- The value of a feasible solution J is the sum of the profits of the jobs in J, or $\sum_{i \in J} P_i$

feasible solution: Every set of jobs which completed by its deadline is a fesible Solution.

- An optimal solution is a feasible solution with maximum value. Here again, since the problem involves the identification of a subset, it fits the subset paradigm.

optimal solution: optimal solution is a ~~fesib~~ feasible solution that maximum profit.

Problem Statement

feasible Solution: Every Set of jobs which completed by its deadLine is a fesible Solution.

optimal Solution: optimal Solution is a ~~fesib~~ feasible solution that maximum profit.

Talking: ACE Live Class 1

(2,4) Not feasible

Example

deadLine is not exceeding 2

- Example Let n = 4, (p_1,p_2,p_3,p_4) = (100,10, 15,27) and (d_1,d_2,d_3,d_4) = (2, 1,2,1). The feasible solutions and their values are:

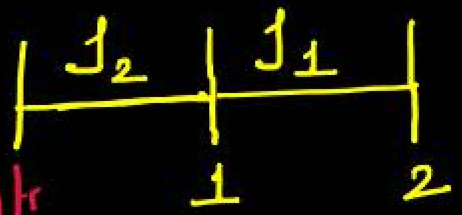
2 1

2,1,2,1

	feasible solution	processing sequence	value
1.	(1, 2)	2, 1 ✓	110
2.	(1,3)	(1. 3) (3, 1)	115
3.	(1, 4)	4-1	127
4.	(2, 3)	2.3	25
5.	(3, 4)	4 - 3	42

(2,2) ✓

(2,4)



Constraints

1. only one M/c available
2. Process the job for 1 time unit
3. Completing jobs by its deadline

- How many jobs can be complete

I have to run the job on a m/c for 1 time unit

I can only complete 2 jobs.

Example

- Example Let n = 4, (p_1,p_2,p_3,p_4) = (100.10, 15,27) and (d_1,d_2,d_3,d_4) = (2, 1,2,1). The feasible solutions and their values are:

	feasible solution	processing sequence	value
1.	(1, 2)	2, 1	110
2.	(1,3)	1, 3 or 3, 1	115
3.	(1, 4)	4, 1	127
4.	(2, 3)	2, 3	25
5.	(3, 4)	4, 3	42

Example

maximum profit

Task	J_1	J_2	J_3	J_4	J_5	J_6	J_7
Profit	35	30	25	20	15	12	5
Deadline	3	4	4	2	3	1	2

How many jobs can be completed?

Selection Criteria:

110 is maximum profit earned

- Ordering the job in decreasing order of profit.
- Selecting a job and putting the last so that initial slot can be filled by other jobs

GATE 2005

We are given 9 tasks $T_1, T_2, \ldots.. T_9$. The execution of each task requires one unit of time. We can execute one task at a time. Each task T_i has a profit P_i and a deadline d_i. Profit P_i is earned if the task is completed before the end of the d_i^{th} unit of time.

Order in decreasing order of profit

Task	T_1	T_2	T_3	T_4	T_5	~~T_6~~	T_7	T_8	T_9
Profit	15	20	30	18	18	~~10~~	$\boxed{23}$	16	25
Deadline	7	2	5	3	4	~~5~~	2	7	3

T_2	T_7	T_9	T_5	T_3	T_1	T_8
1	2	3	4	5	6	7

- which job will be left out - T_4, T_6
- Maximum profit earned

$$30 + 25 + 23 + 20 + 18 + 16 + 15 = 147$$

GATE 2005

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Profit	15	20	30	18	18	10	23	16	25
Deadline	7	2	5	3	4	5	2	7	3

1. Are all tasks completed in the schedule that gives maximum profit?

(a) All tasks are completed

(b) T_1 and T_6 are left out

(c) T_1 and T_8 are left out

(d) T_4 and T_6 are left out

GATE 2005

Task	**T_1**	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**	**T_7**	**T_8**	**T_9**
Profit	15	20	30	18	18	10	23	16	25
Deadline	7	2	5	3	4	5	2	7	3

Q. What is the maximum profit earned?

(a)147

(b)165

(c)167

(d)175

1. optimal merge pattern
2. Huffman code
3. Prims & Kruskal spanning tree
4. Dijktra's shortest path) Single Source shortest path

1. knapsack problem (fractional)
2. Job sequencing with deadlin

1. optimal solution
2. feasible st solution
3. objective function
4. ordering (selection criteria)
5. constraints

Q. How many different schedules are possible?

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Profit	15	20	30	18	18	10	23	16	25
Deadline	7	2	5	3	4	5	2	7	3

Question Homework

Q. We are given 9 tasks T_1, T_2, …. T_9. The execution of each task requires one unit of time. We can execute one task at a time. Each task T_i has a profit P_i and a deadline d_i. Profit P_i is earned if the task is completed before the end of the d_i^{th} unit of time.

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Profit	15	20	30	18	18	10	23	16	25
Deadline	5	3	7	3	4	6	7	4	3

If we want to maximize the profit then Number of different schedule possible is

Heap

Binary Heap ~~Sec~~ Selection criteria

min to come out

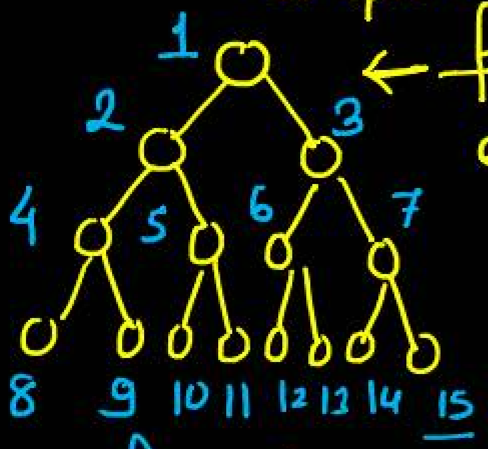
A Heap is a special Tree-based data structure in which the tree is a complete binary tree. Generally, Heaps can be of two types:

min/max

deletion

- Min Heap ✓
- Max heap ✓

Complete Binary tree



full Binary tree

← full Binary tree

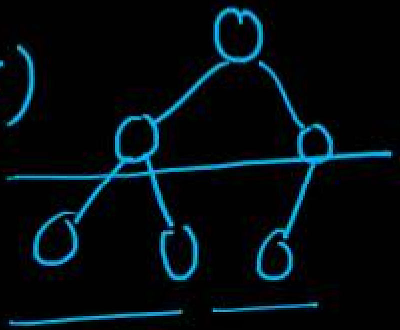
all Leaves at the Same Level

Each Node will have 0/2 children

Complete Binary tree is a tree in which every level is full except the Last. At the Last level the nodes will filled from ~~as~~ Left ~~as~~ Hand side

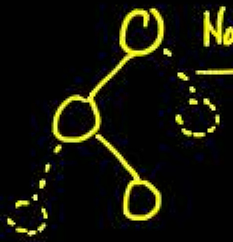
(I) Complete Binary tree

(I)

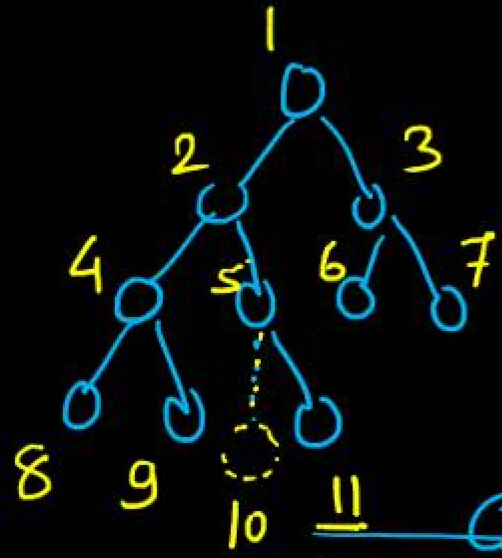


complete binary tree

No or Index

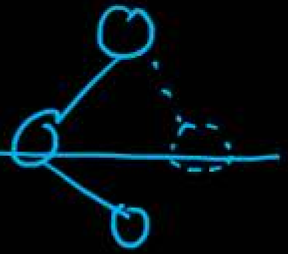


(II)



Complete Binary tree?

Not a complete binary tree



Not CBT

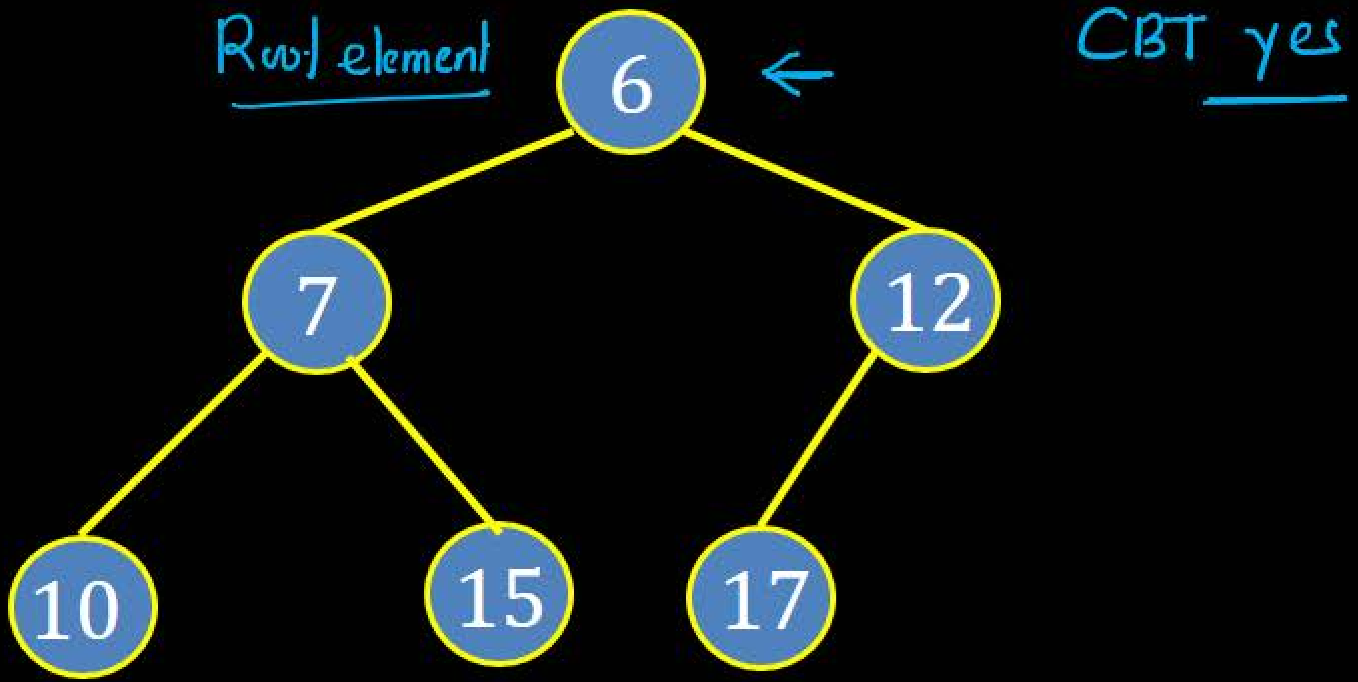
Not filled

- fill the node index wise
- there should be no empty position

Min Heap

Min-Heap: In a Min-Heap the key present at the root node must be minimum among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.

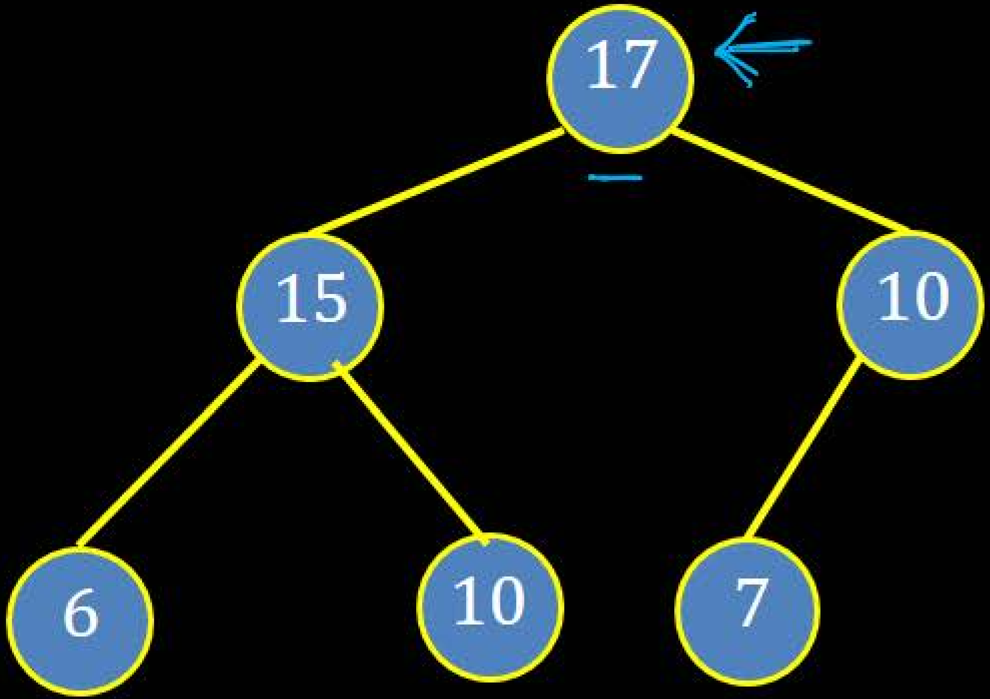
Min Heap



Max Heap

Max-Heap: In a Max-Heap the *key* present at the root node must be greatest among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.

Max Heap



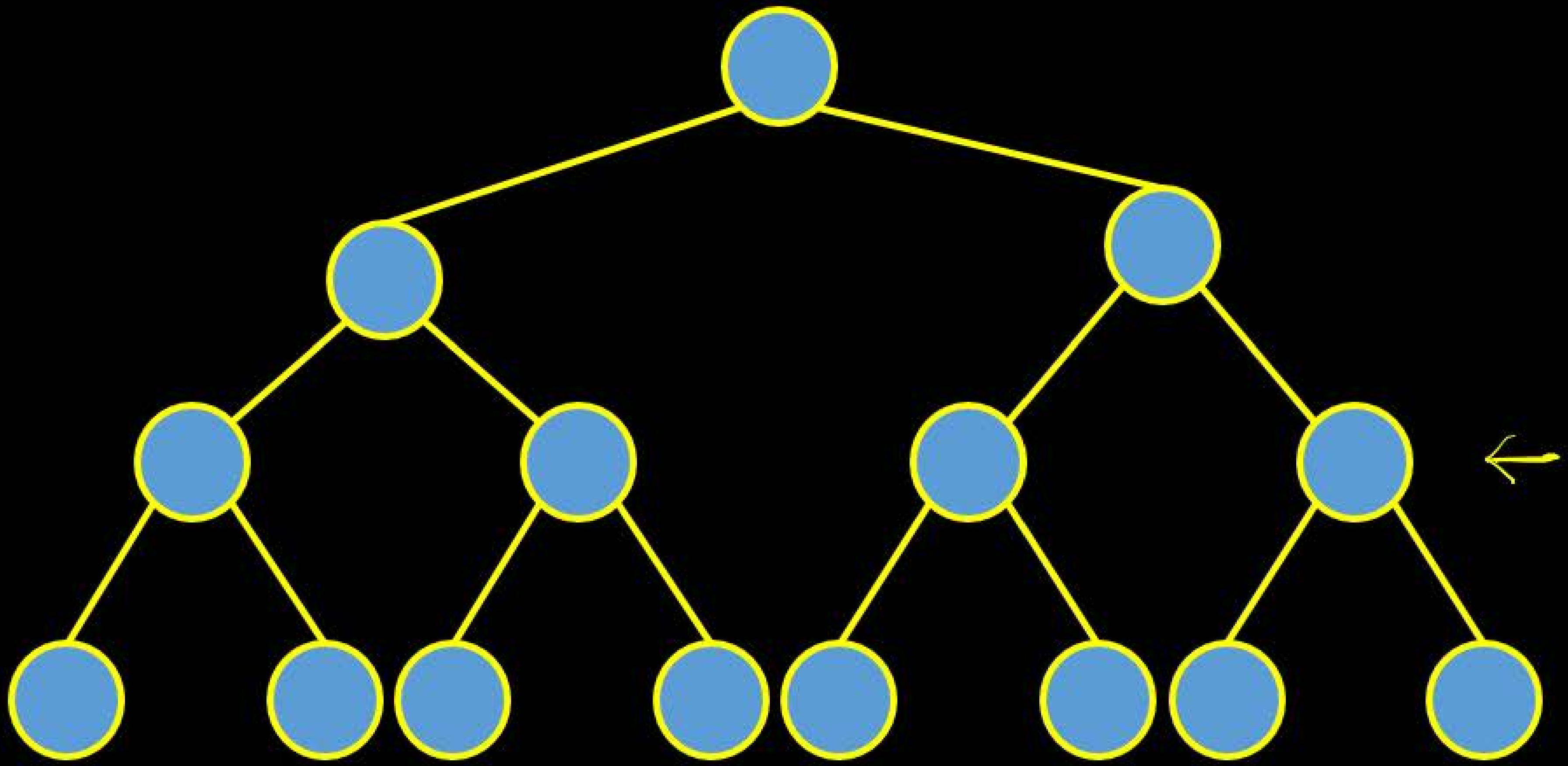
Every level this ~~prob~~ property will be satisfied

- Insert : Cost
- Deletion : Cost

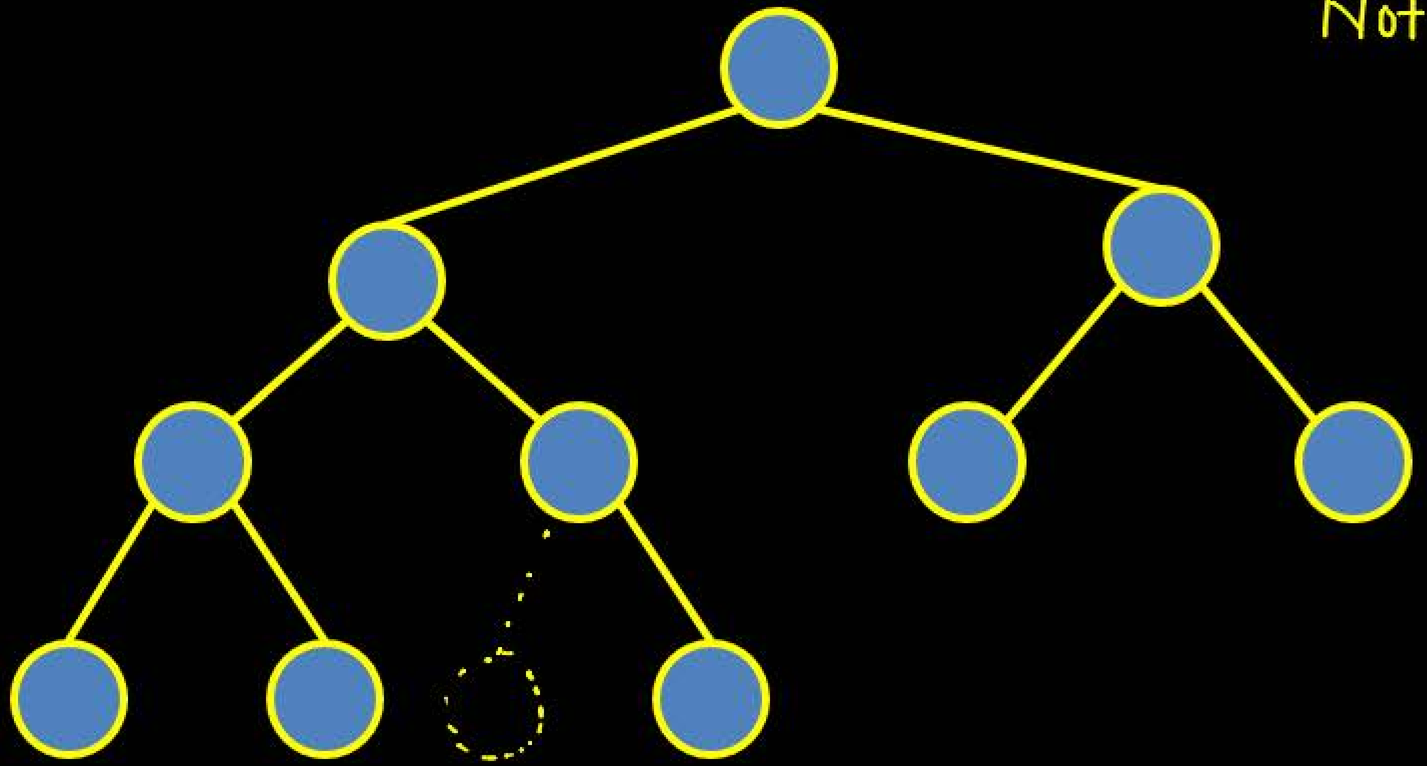
Complete Binary Tree

A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

Complete Binary Tree

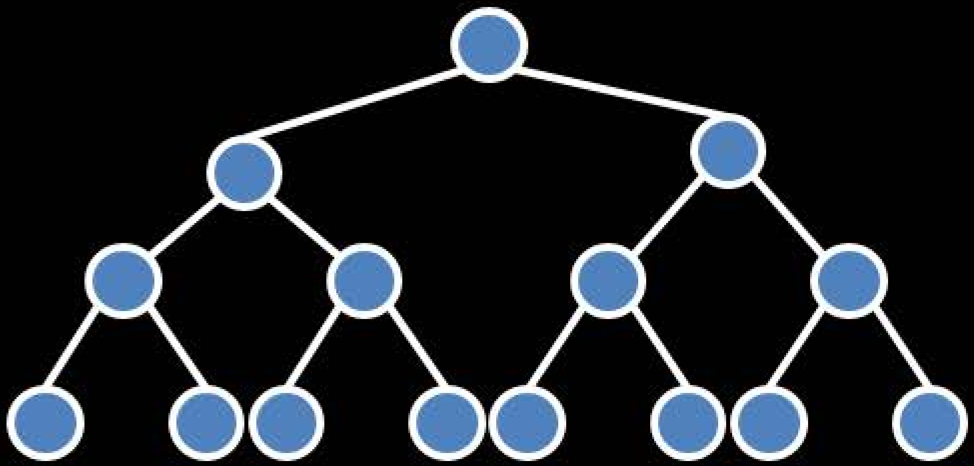


Not a Complete Binary Tree

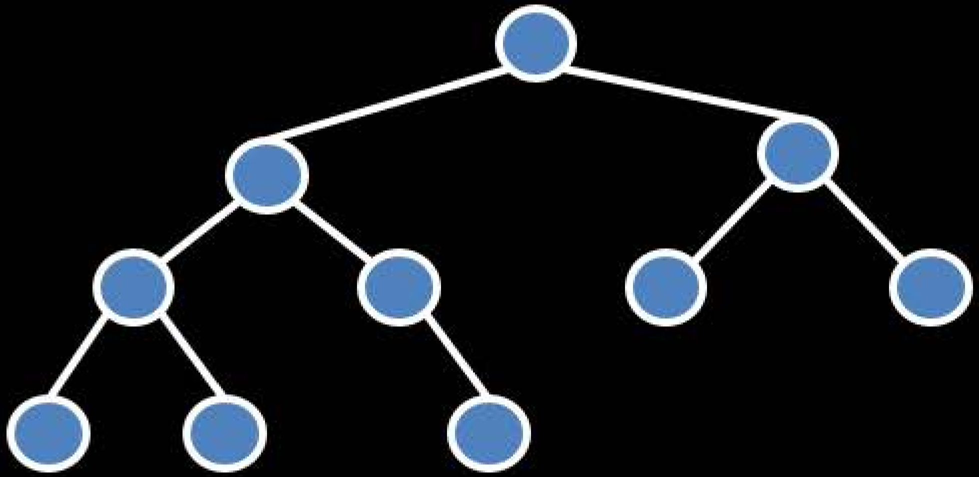


Complete Binary Tree

- A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.



Not a Complete Binary Tree



Question

The number of nodes a heap of height k can hold is

(A) 2^k to 2^{k+1}- 1

(B) 2^{k+1} to 2^{k+1}- 1

(C) 2^{k-1} to 2^{k+1}- 1

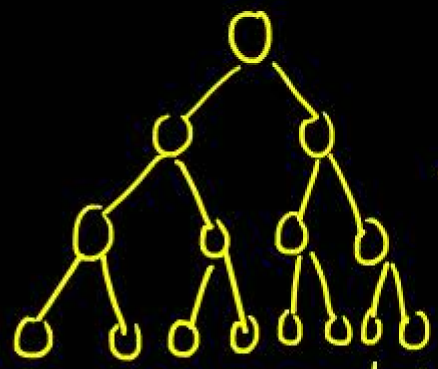
(D) 2^k -1to 2^{k+1}- 1

(A)

Binary tree k

k=3

Range - Minimum No. of Nodes
Maximum No. of Nodes

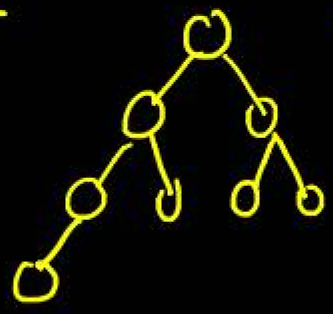


maximum No. of Nodes

$$2^0+2^1+2^2+2^3 = \frac{1(2^4-1)}{2-1}$$

$$= 2^{3+1}-1$$

$$= 2^{k+1}-1$$



$$2^0+2^1+2^2+\dots+2^{k-1}+1$$

$$\frac{1(2^k-1)}{2-1}+1$$

$$2^k-1+1 = \boxed{2^k}$$