Module 1: Approach to Programming

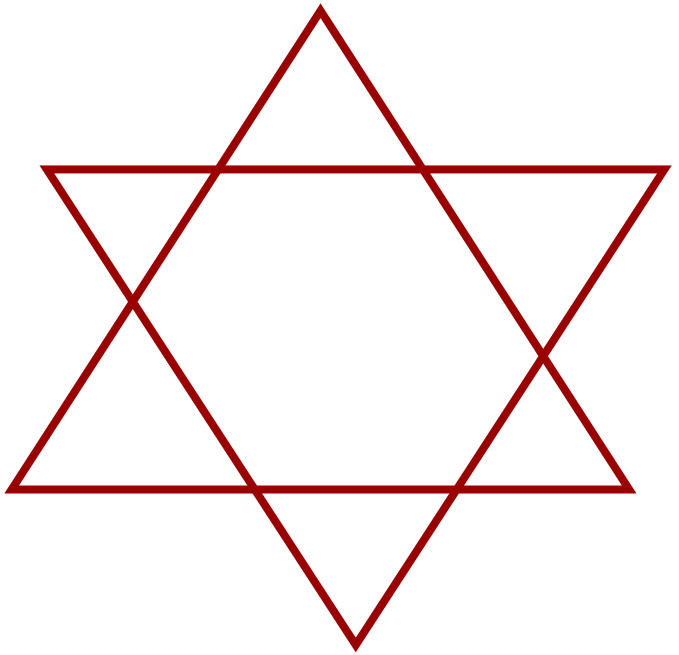
Murali P

Approach to Programming

- Top-down Design
- Correctness Issues
- Implementation Issues
- Efficiency & Complexity

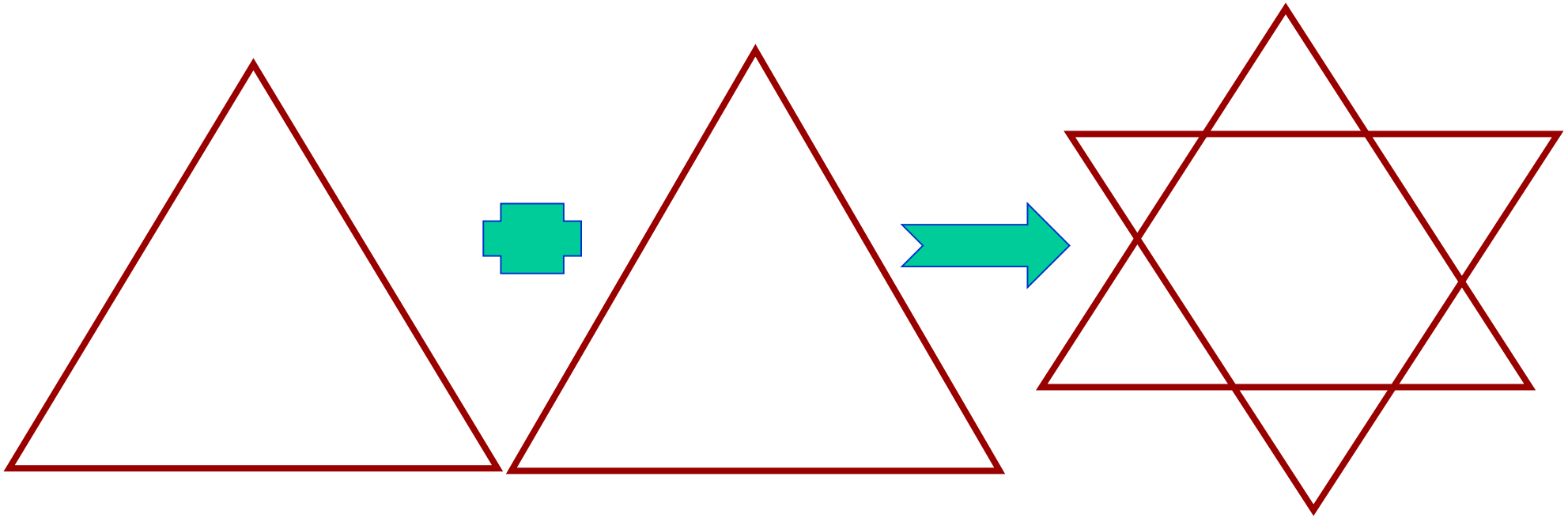
Design Approach

- How do you make



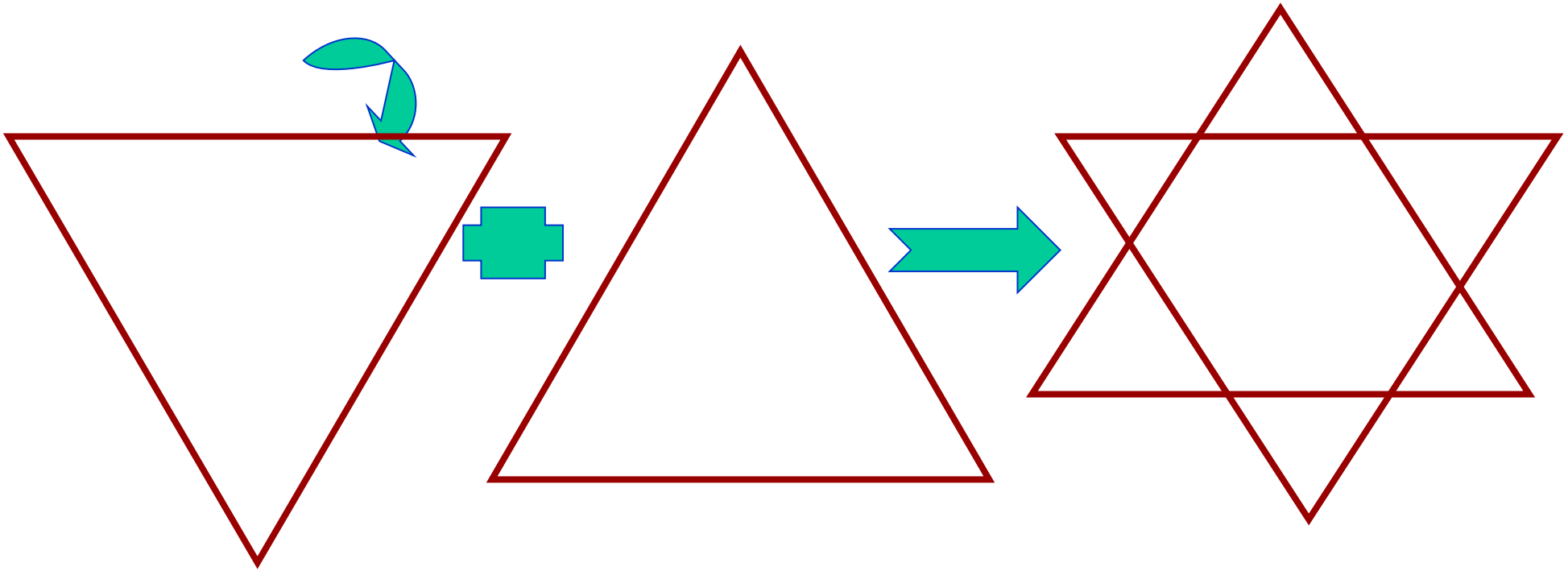
Design Approach

- One method (given triangles)



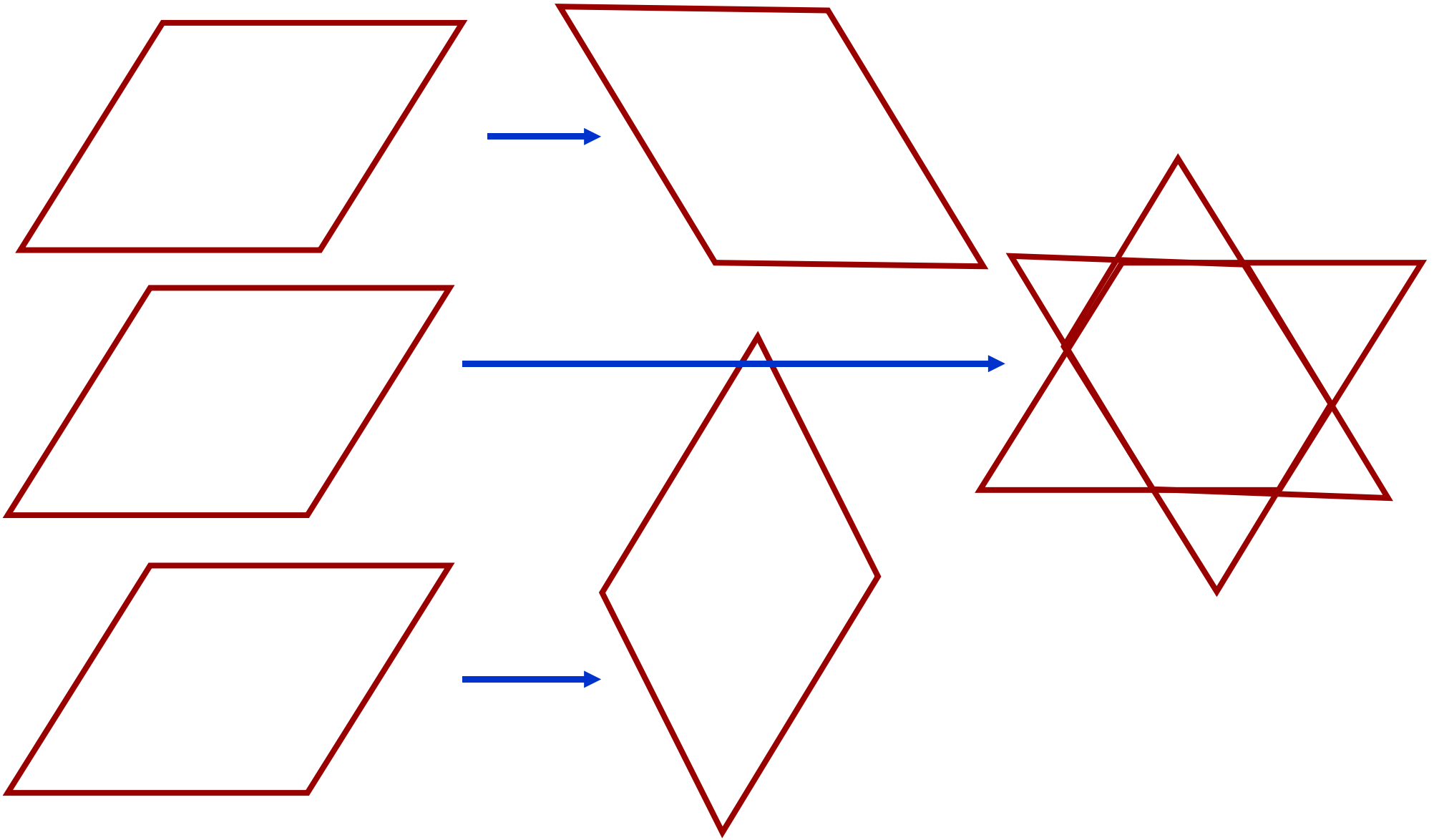
Design Approach

- One method (given triangles)



Design Approach

- Another method (given only trapeziums)



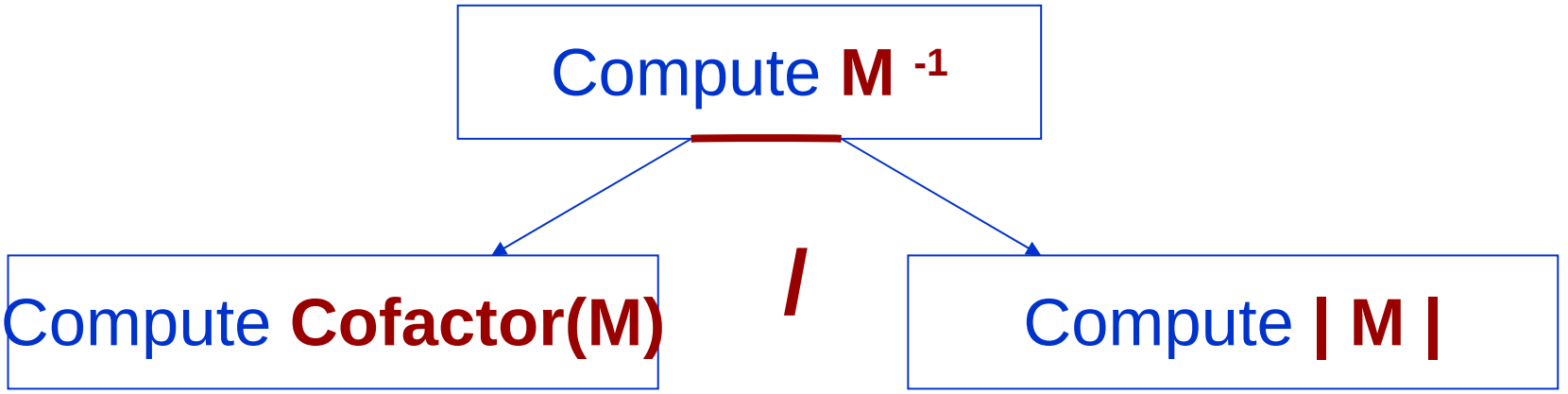
Top-down Design

- Design Approach

- Principle
  - Divide and Conquer
- Steps
  - Divide the problem into sub-problems
    - Basic figures
  - Solve the sub-problems
    - How to make triangles/trapeziums?
  - Combine the (sub-)solutions
    - Rotate appropriately
    - Merge them

Inverse of a Matrix M

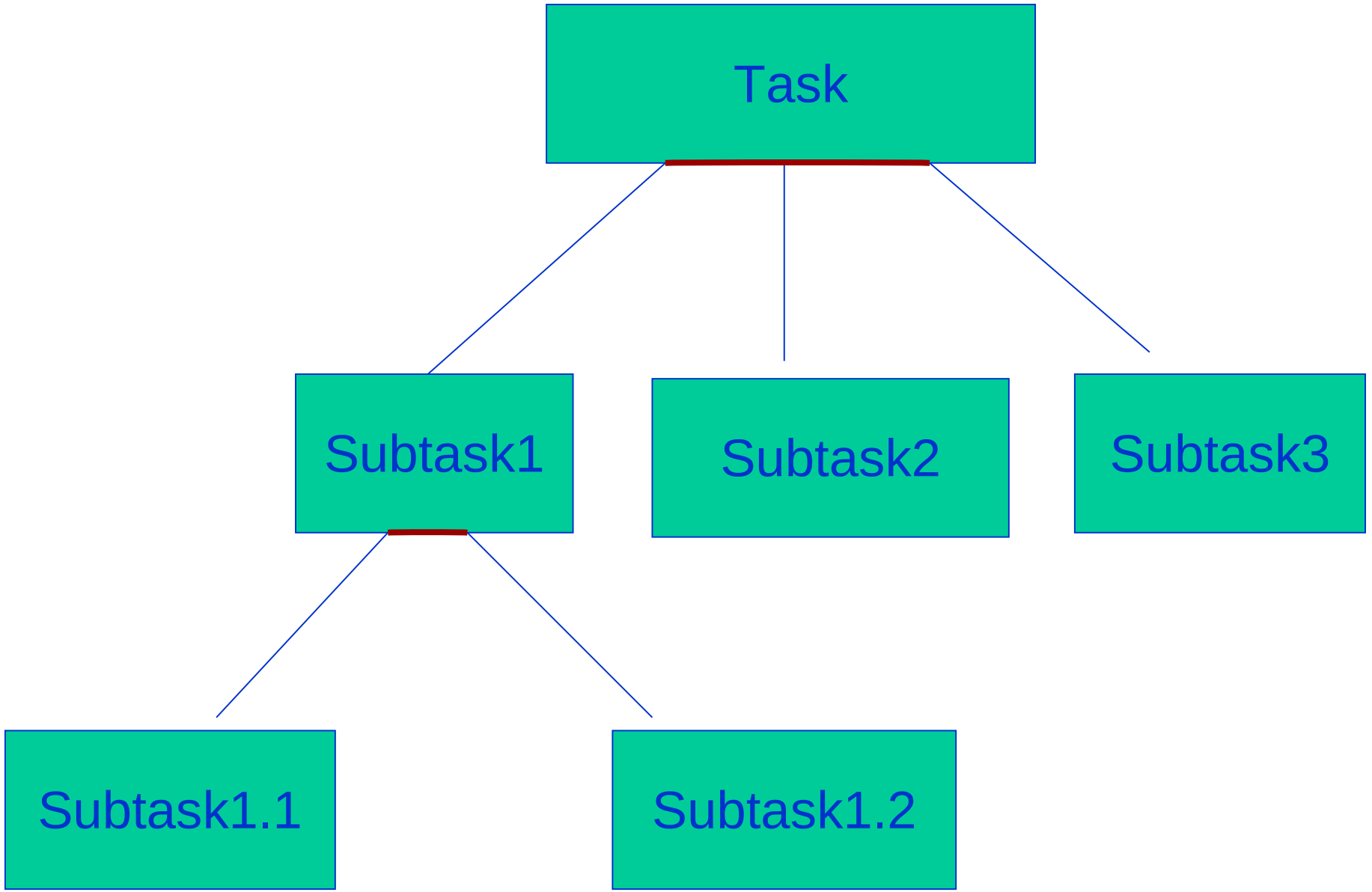
- Inverse(M) = Cofactor(M) / | M |



$$\text{coafactor}_{i,j}(M) = |\ M - i^{th}\ \text{row}, j^{th}\ \text{column}\ | * (-1)^{i+j}$$

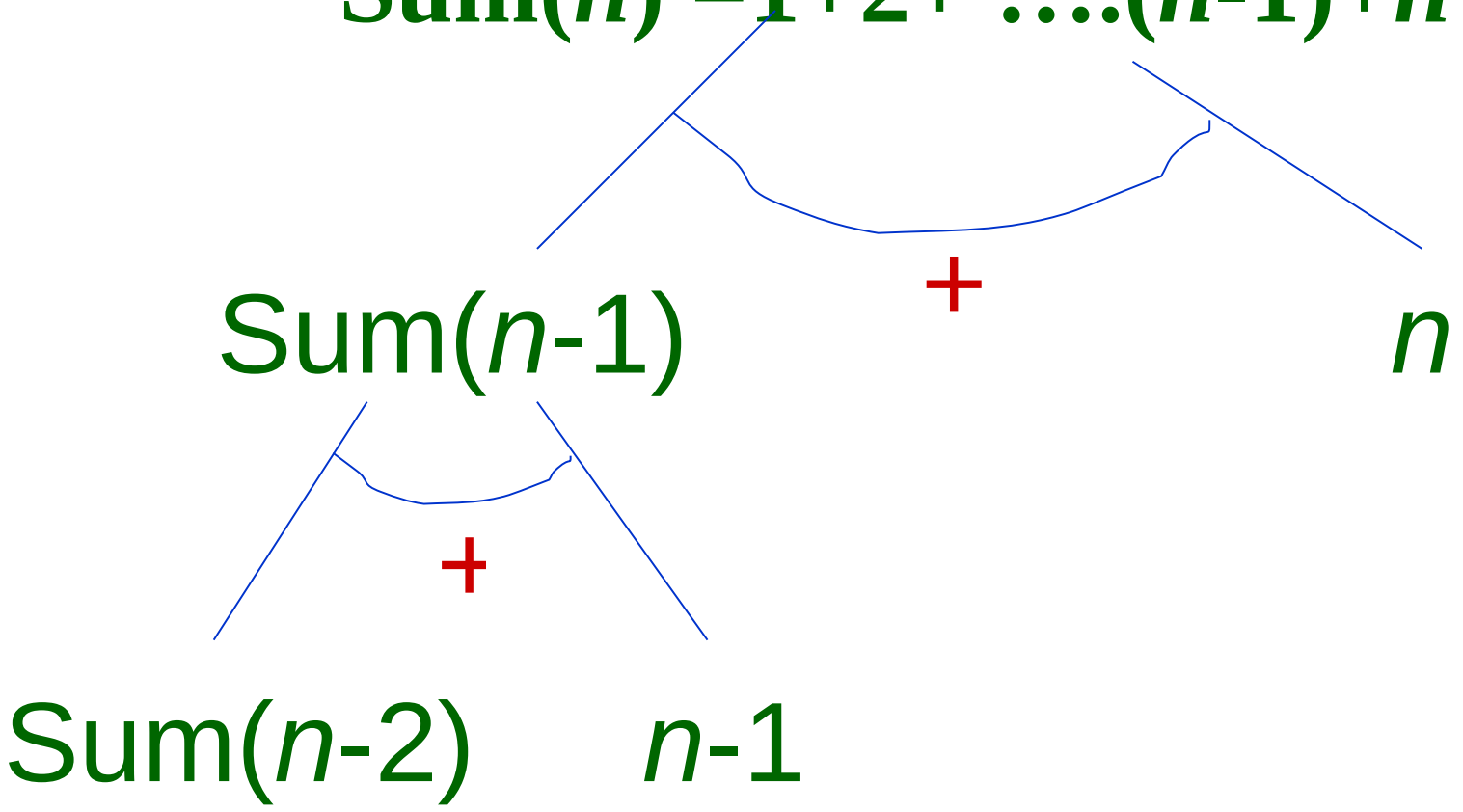
Top-down Design

- How do you solve the sub-problem?
- Divide-and-conquer again!
- Steps
  1. Divide the problem into sub-problems
  2. Repeat step 1 for each sub-problem
     until the problem is small enough to have an atomic solution.
  3. Combine the solutions.


Task
Subtask1
Subtask2
Subtask3
Subtask1.1
Subtask1.2

Example:

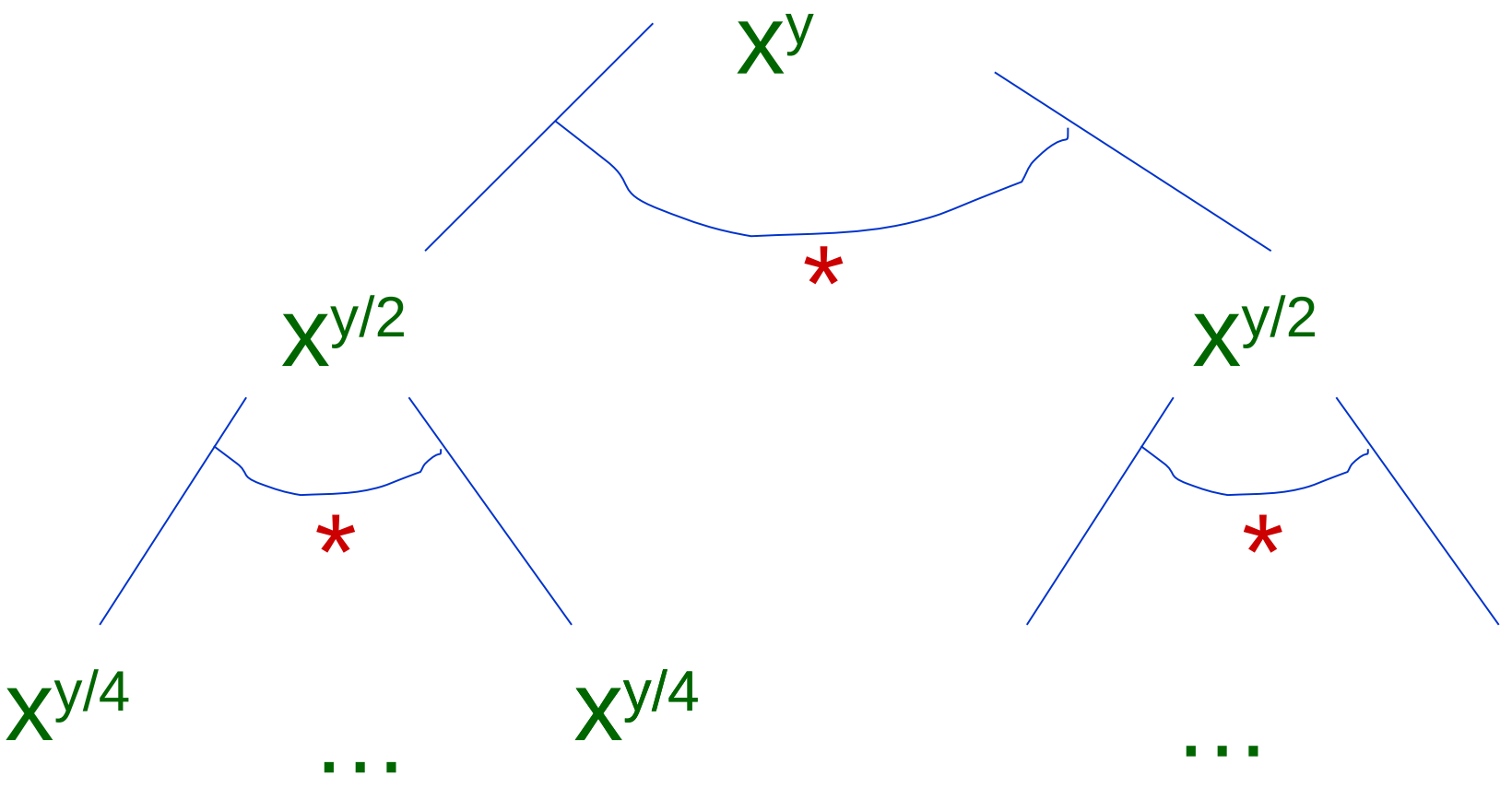
Sum(*n*) =1+2+ ….(*n*-1)+*n*



……

Example – x^y

- Consider x^y, where $y=2^k$



Example

- Problem: Compute x^y , for any $y \geq 0$
- (Top-down Design) Solution:
  - Divide y into binary components $y_0 y_1$, y_2, … $y_{k.-1}$
  - each y_j is 0 or a power of 2 and their sum is y.
  - To combine solutions: $x^y = x^{y0*y1} * x^{y2} * \ldots * x^{yk}$
  - Each sub-problem x^{yj} has a known solution.

Top-down Design – Example

- Algorithm for x^y
  - Initially temp=1;
  - Extract a power of 2 from y, say P.
  - Compute x^P and multiply this to temp
  - Repeat above steps until nothing to extract.

Top-down Design – Example

- Algorithm for x^y

```
Y_next = y; Power = 1; Result=1;
while (Y_next > 0) do
   if (Y_next mod 2 == 1)
   then Result = Result * pow(x, Power);
   Power = 2 * Power;
   Y_next = Y_next / 2;
endwhile;
```

Illustration: x^9

Iteration	Result	Power	Y_next
0	1	1	9
1	9%2==1→1*x^1	2	9/2=4
2	Condition fails →x	2*2=4	4/2=2
3	Condition fails →x	2*4=8	2/2=1
4	1%2==1 →x*x^8	2*8=16	1/2=0

Illustration: x^8

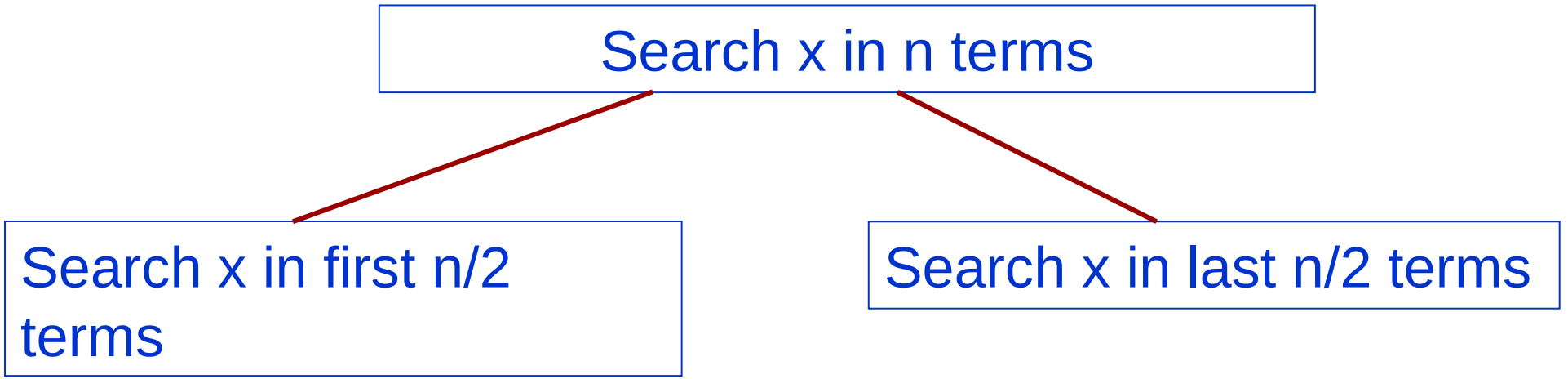
Iteration	Result	Power	Y_next
0	1	1	8
1	8%2==0 Result=1 Condition fails	2	8/2=4
2	4%2==0 Result=1 Condition fails	2*2=4	4/2=2
3	2%2==0 Result=1 Condition fails	2*4=8	2/2=1
4	1%2==1 →$1*x^8$	2*8=16	1/2=0

Divide and Conquer Example

Problem: Searching an Ordered list

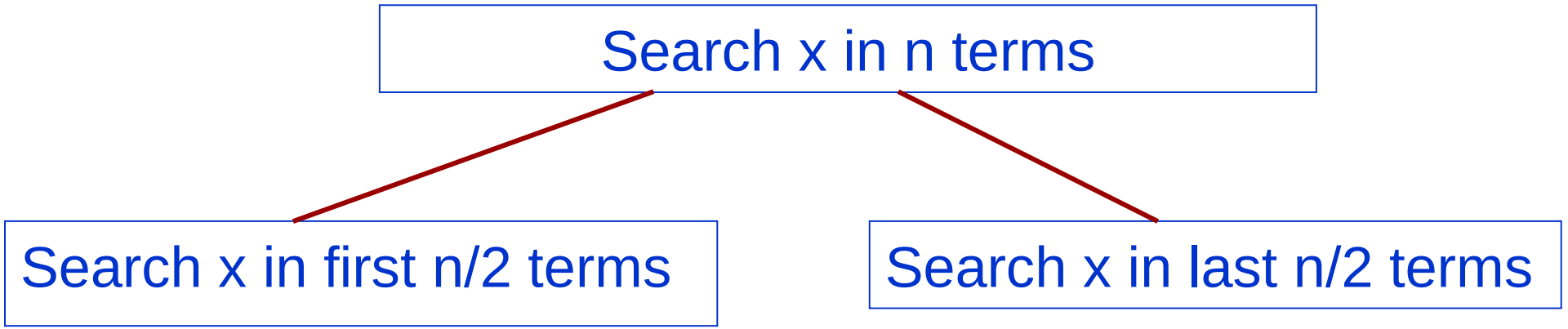
- Given an *ordered sequence* of N elements, find whether there exists an element 'x' in the sequence or not

Broadly speaking

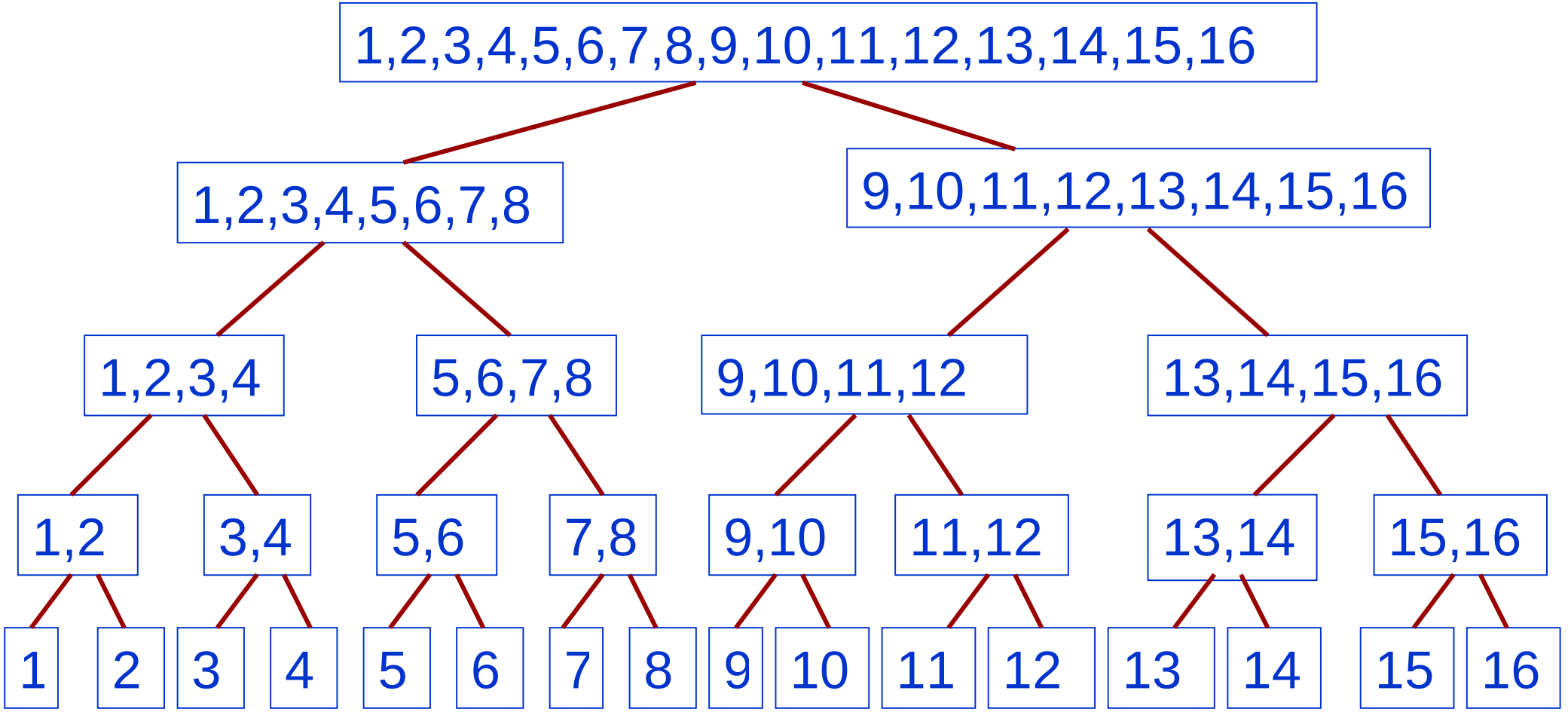


Binary Search - design

Broadly speaking



Binary Search - example



Binary Search - Design

Broadly speaking