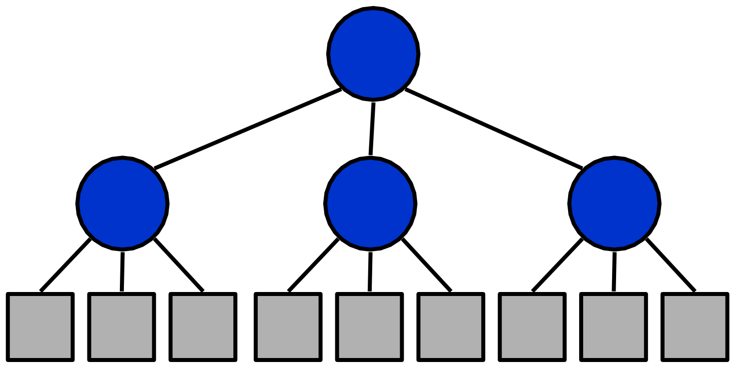
Divide-and-Conquer Technique:
Finding Maximum & Minimum

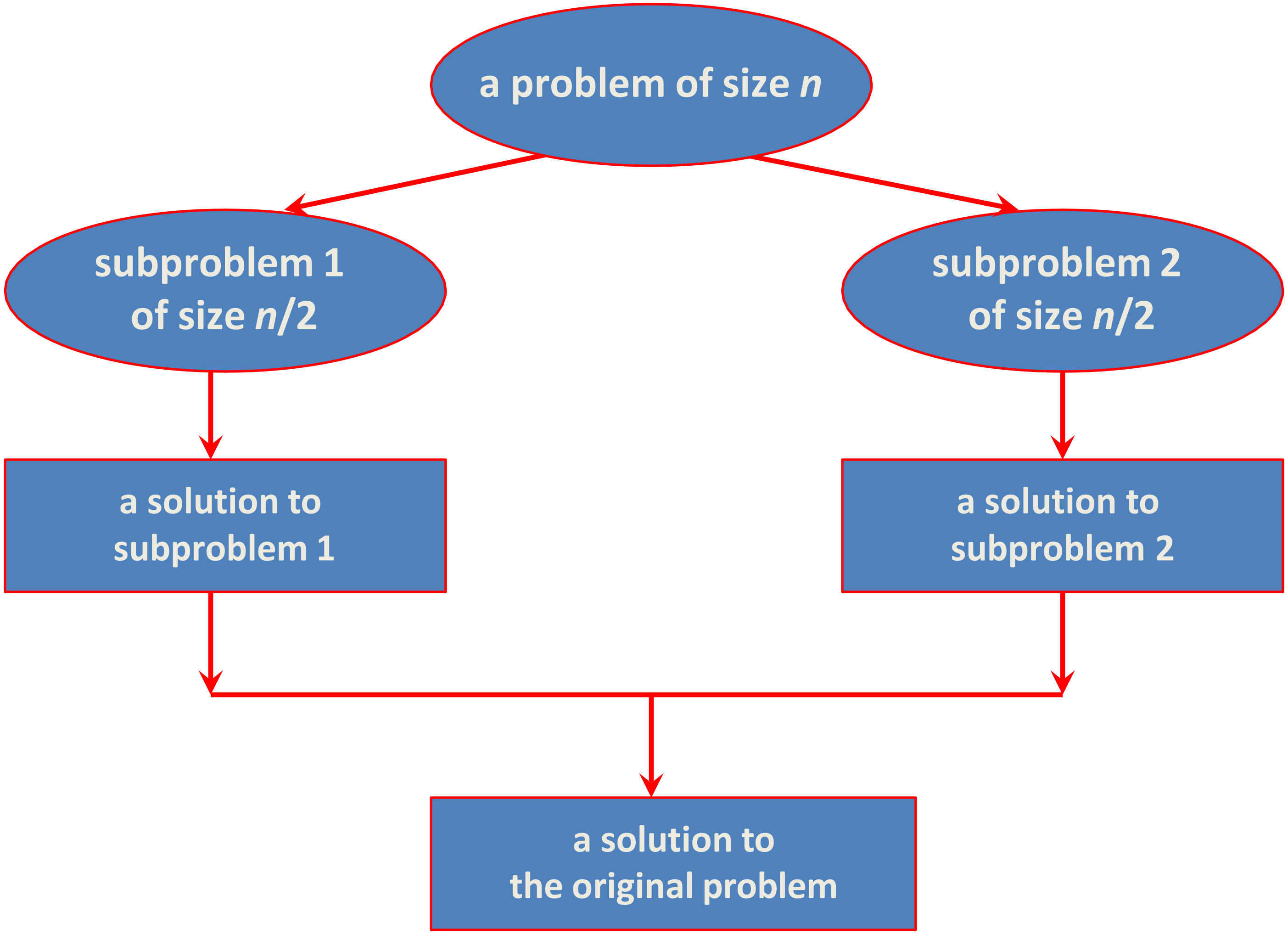
Lecturer Saifur Rahman, Dept. of CSE, United International University

Divide-and-Conquer

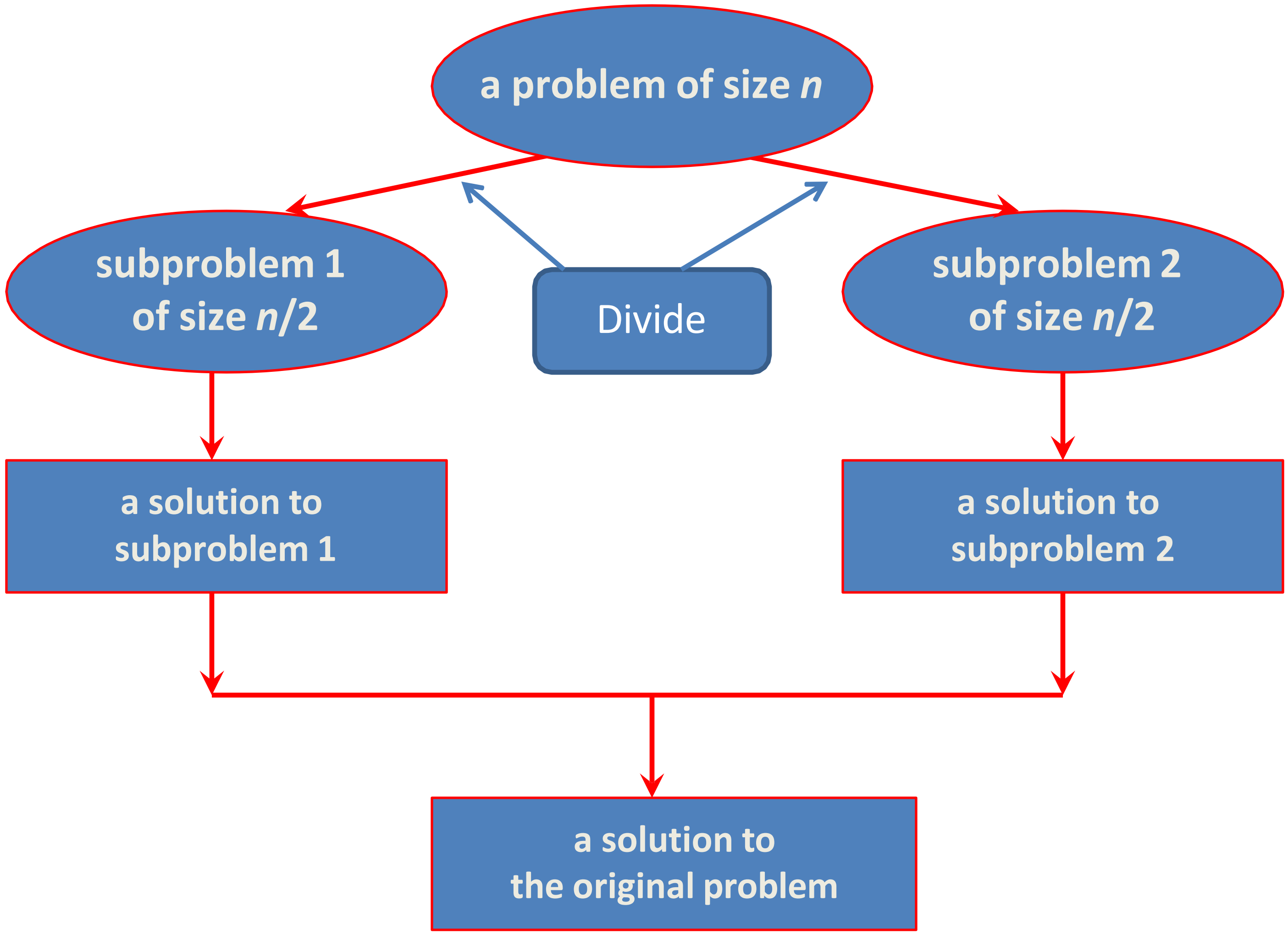
- Divide-and-Conquer is a general algorithm design paradigm:
  - Divide the problem into a number of subproblems that are smaller instances of the same problem
  - Conquer the subproblems by solving them recursively
  - Combine the solutions to the subproblems into the solution for the original problem
- The base case for the recursion are subproblems of constant size
- Analysis can be done using **recurrence equations**



Divide-and-Conquer



Divide-and-Conquer



Finding Maximum and Minimum

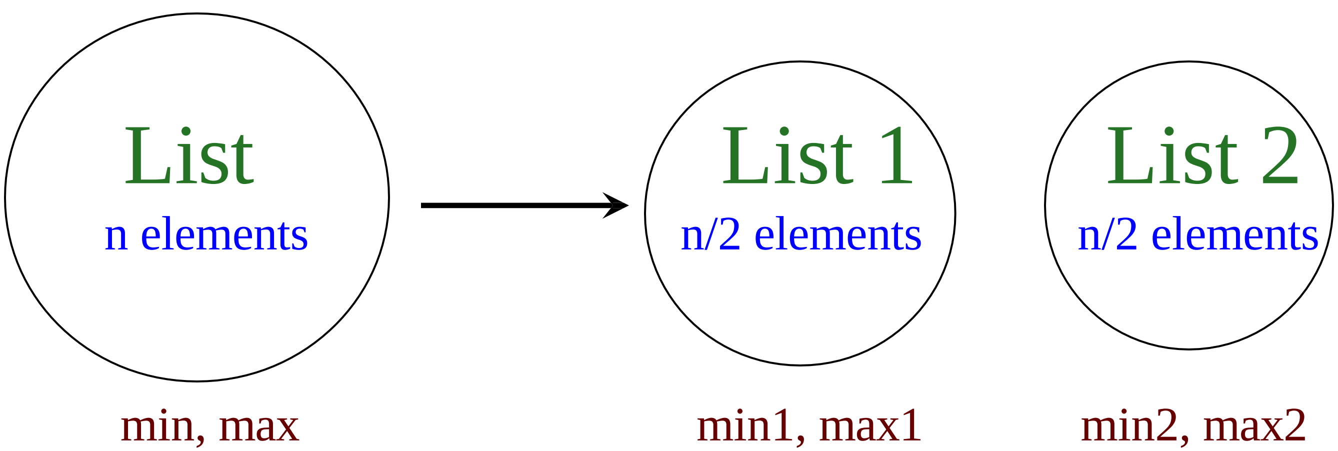
- *Input:* an array $A[1..n]$ of n numbers
- *Output:* the maximum and minimum value

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	**13**	**-13**	**-25**	**20**	**-3**	**-16**	**-23**	**18**	**20**	**-7**	**12**	**-5**	**-22**	**15**	**-4**	**7**

Finding Maximum and Minimum

- *Input:* an array $A[1..n]$ of n numbers
- *Output:* the maximum and minimum value

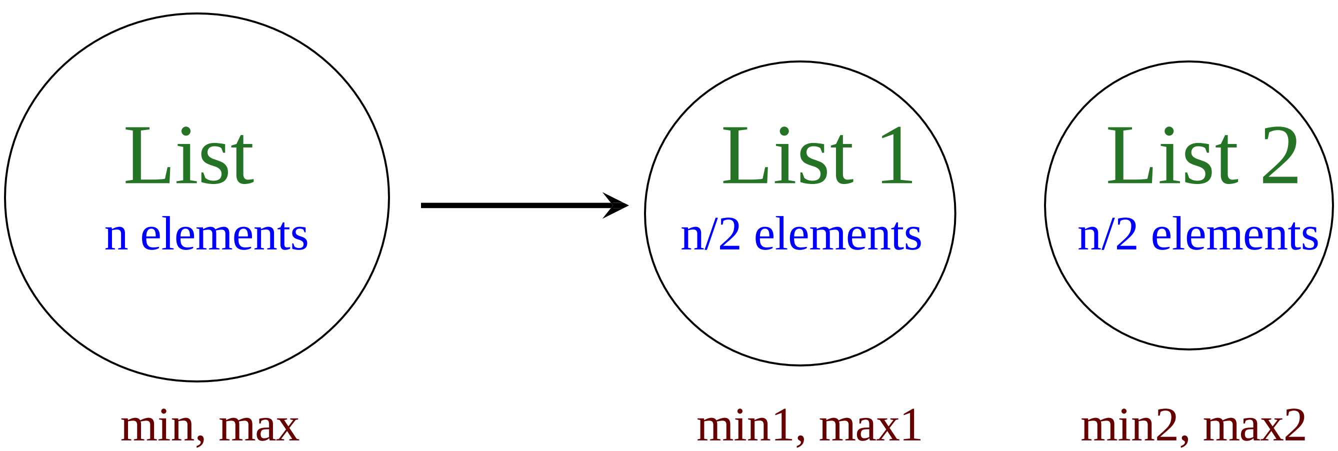
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	**13**	**-13**	**-25**	**20**	**-3**	**-16**	**-23**	**18**	**20**	**-7**	**12**	**-5**	**-22**	**15**	**-4**	**7**



Finding Maximum and Minimum

- *Input:* an array $A[1..n]$ of n numbers
- *Output:* the maximum and minimum value

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	**13**	**-13**	**-25**	**20**	**-3**	**-16**	**-23**	**18**	**20**	**-7**	**12**	**-5**	**-22**	**15**	**-4**	**7**



min = MIN (min1, min2)

max = MAX (max1, max2)

Finding Maximum and Minimum

The straightforward algorithm:

max ← min ← A (1);
for i ← 2 to n do
    if (A (i) > max) then max ← A (i);
    if (A (i) < min) then min ← A (i);

No. of comparisons: $2(n - 1)$

Finding Maximum and Minimum

The Divide-and-Conquer algorithm:

```
procedure Rmaxmin (i, j, fmax, fmin);        // i, j are index #, fmax,
    begin                                    // fmin are output parameters
      case:

      end
    end;
```

Finding Maximum and Minimum

The Divide-and-Conquer algorithm:

procedure Rmaxmin (i, j, $fmax$, $fmin$); // i, j are index #, $fmax$,
begin // $fmin$ are output parameters
case:
$i = j$: $fmax \leftarrow fmin \leftarrow A[i]$;

end
end;

Finding Maximum and Minimum

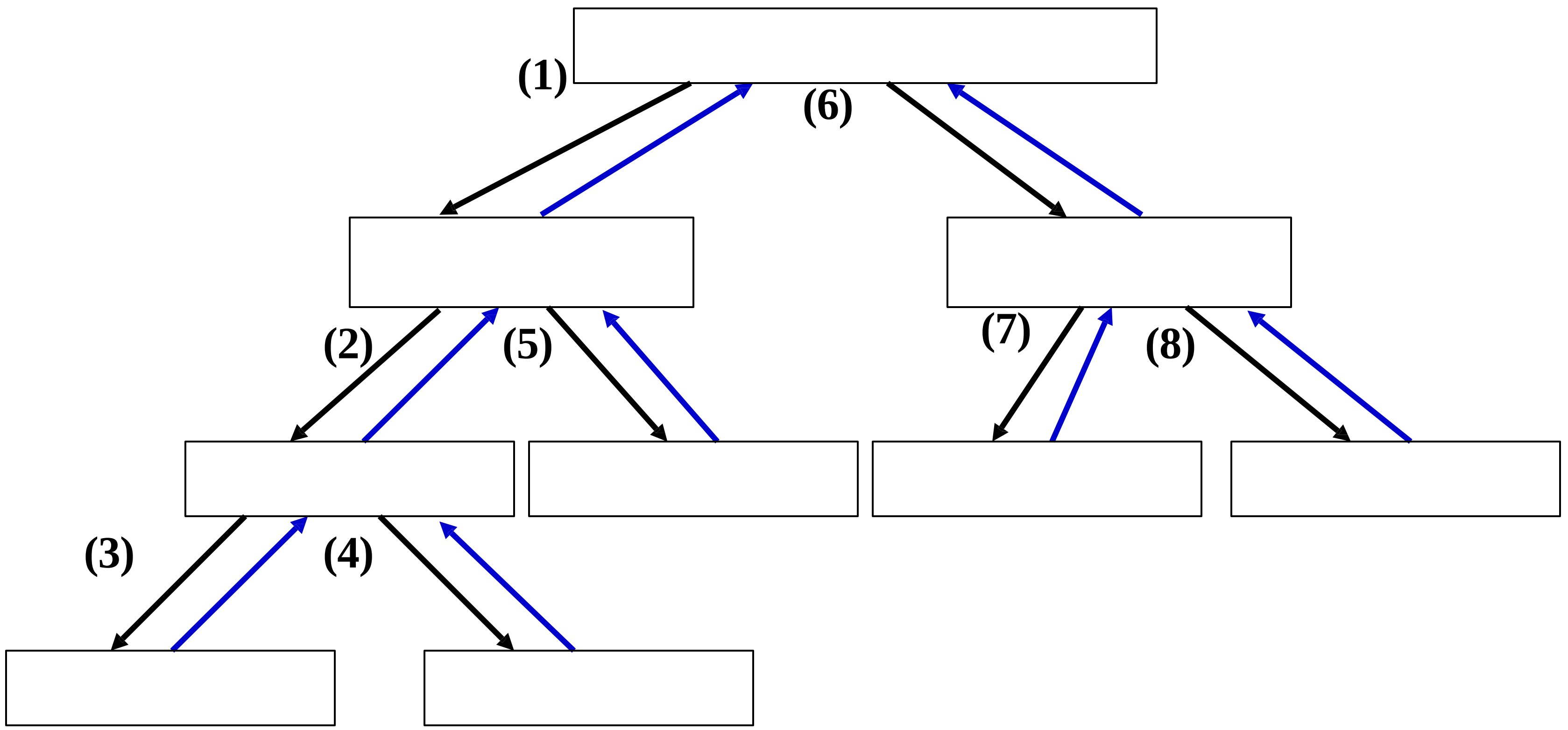
The Divide-and-Conquer algorithm:

```
procedure Rmaxmin (i, j, fmax, fmin);        // i, j are index #, fmax,
    begin                                     // fmin are output parameters
      case:
          i = j:        fmax ← fmin ← A[i];

          else:         mid ← (i + j)/2;
                        call Rmaxmin (i, mid, gmax, gmin);
                        call Rmaxmin (mid+1, j, hmax, hmin);
                        fmax ← MAX (gmax, hmax);
                        fmin ← MIN (gmin, hmin);
      end
    end;
```

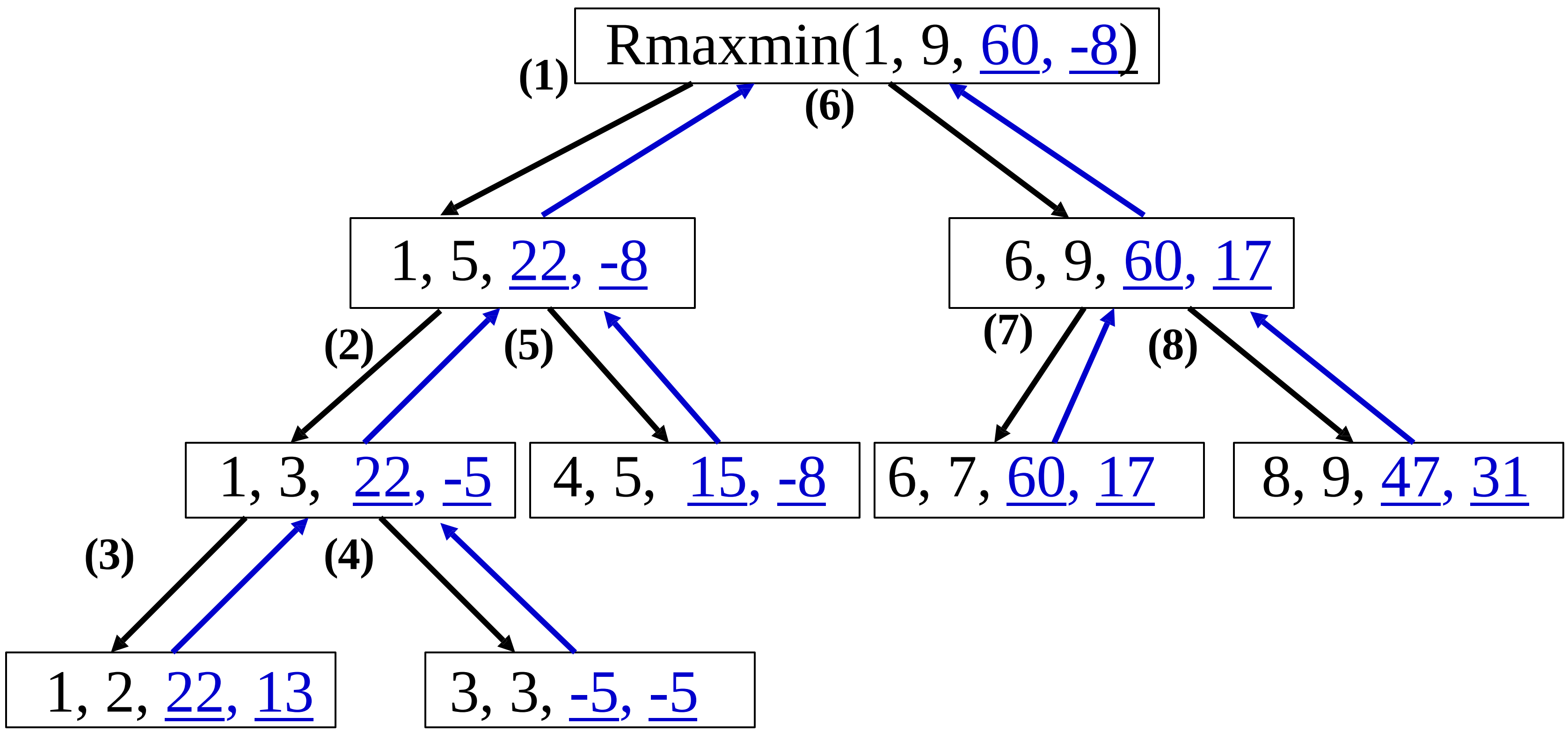
Finding Maximum and Minimum

Index:	1	2	3	4	5	6	7	8	9
Array:	22	13	-5	-8	15	60	17	31	47



Finding Maximum and Minimum

Index:	1	2	3	4	5	6	7	8	9
Array:	22	13	-5	-8	15	60	17	31	47



Finding Maximum and Minimum

The recurrence for the worst-case running time $T(n)$ is

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2\mathrm{T}(n/2) + \Theta(1) & \text{if } n > 1 \end{cases}$$

equivalently

$$T(n) = \begin{cases} b & \text{if } n = 1 \\ 2T(n/2) + b & \text{if } n > 1 \end{cases}$$

By solving the recurrence, we get

$T(n)$ is $O(n)$