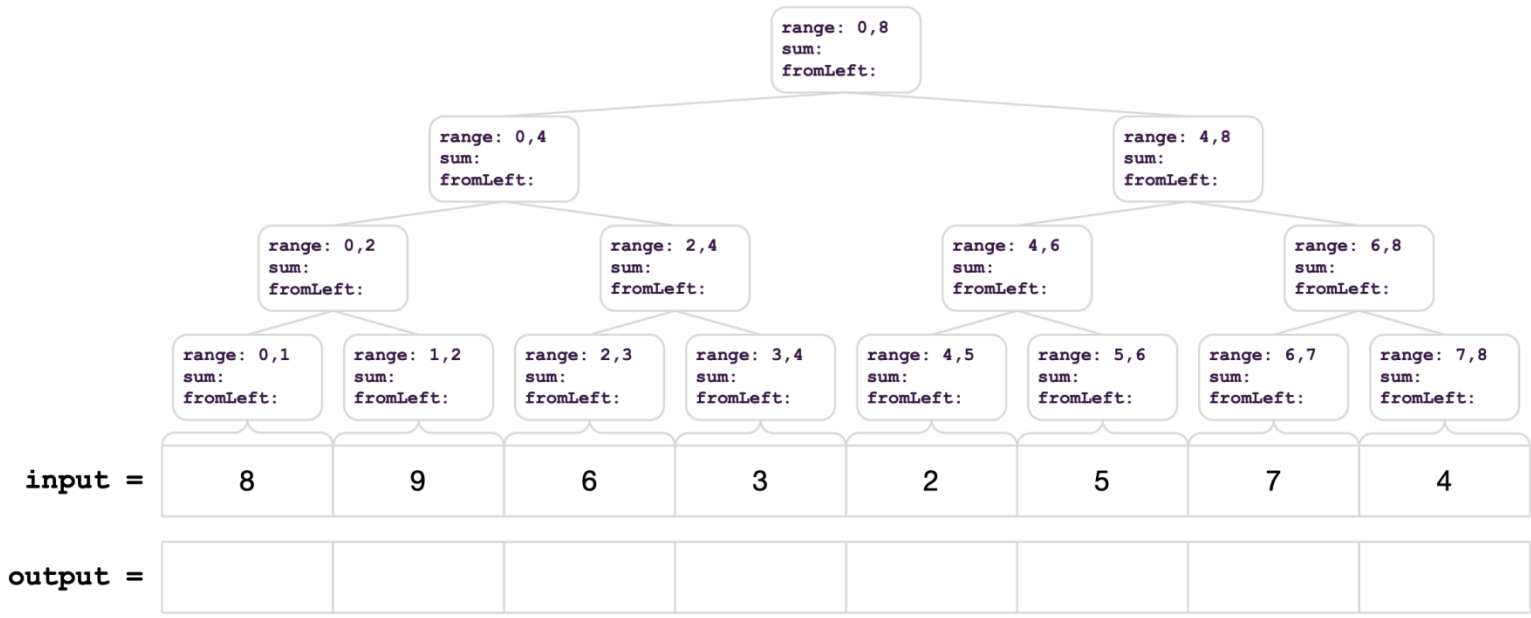
CSE 332: Data Structures and Parallelism

Section 8: Parallel Prefix

0a. Parallel Prefix Sum

Given input array `[8, 9, 6, 3, 2, 5, 7, 4]`, output an array such that each `output[i] = sum(array[0], array[1], ..., array[i])`.

Use the Parallel Prefix Sum algorithm from lecture. Show the intermediate steps. Draw the input and output arrays, and for each step, show the tree of the recursive task objects that would be created (where a node's child is for two problems of half the size) and the fields each node needs. Do not use a sequential cut-off.



leaves[i].sum = _____;

parent.sum = _____;

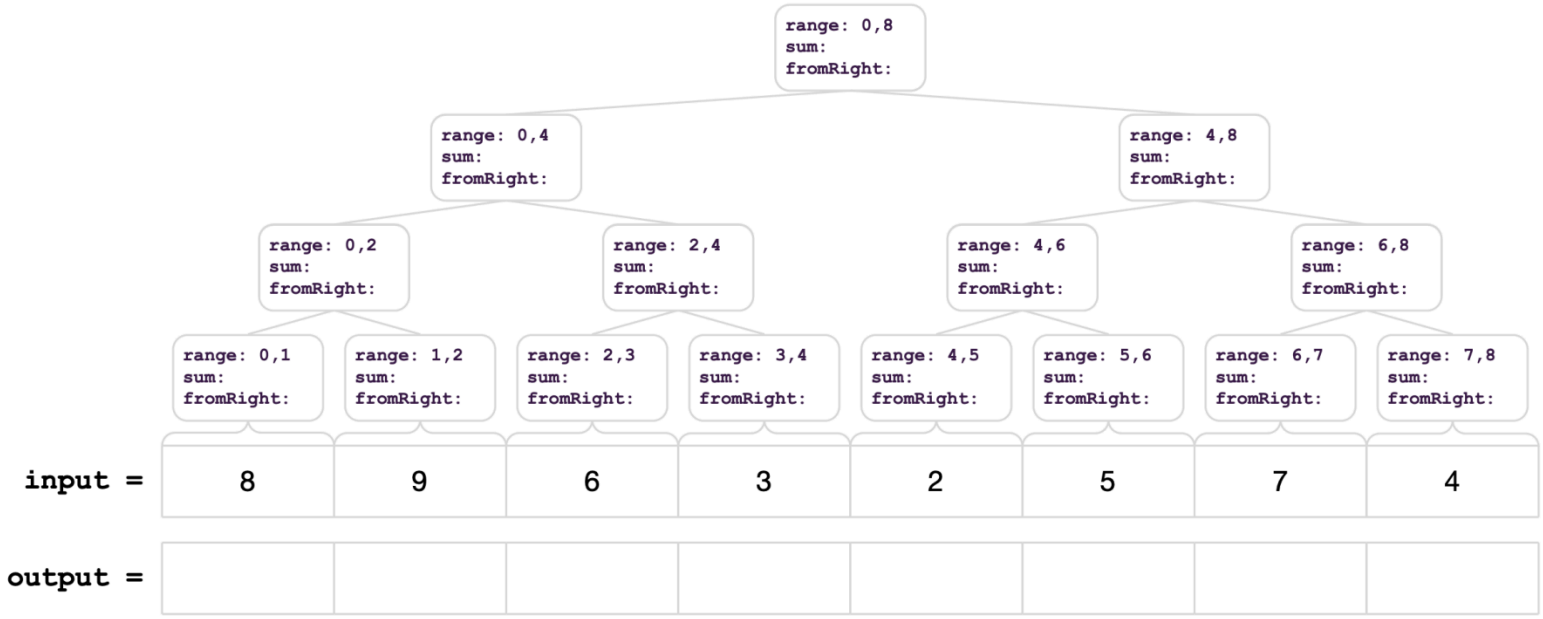
left.fromLeft = _____;
right.fromLeft = _____;

output[i] = _____;

0b. Parallel Suffix Sum

Given input array `[8, 9, 6, 3, 2, 5, 7, 4]`, output an array such that each `output[i] = sum(array[0], array[1], ..., array[i])`.

Use the Parallel Suffix algorithm. Show the intermediate steps. Draw the input and output arrays, and for each step, show the tree of the recursive task objects that would be created (where a node's child is for two problems of half the size) and the fields each node needs. Do not use a sequential cut-off.



leaves[i].sum = _____;

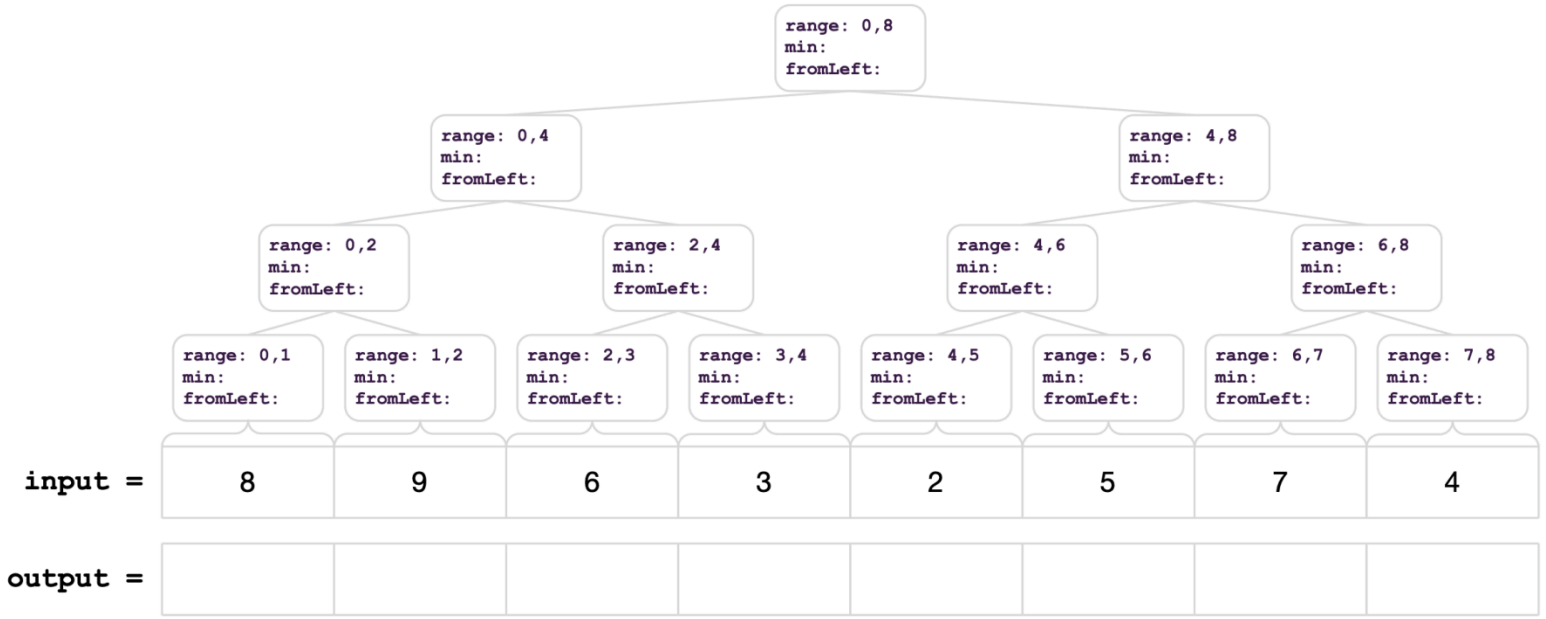
parent.sum = _____;

left.fromRight= _____;
right.fromRight= _____;

output[i] = _____;

1. Parallel Prefix FindMin

Given input array `[8, 9, 6, 3, 2, 5, 7, 4]`, output an array such that each `output[i] = min(array[0], array[1], ..., array[i])`. Show all steps, as above.



leaves[i].min = _____;

parent.min = _____;

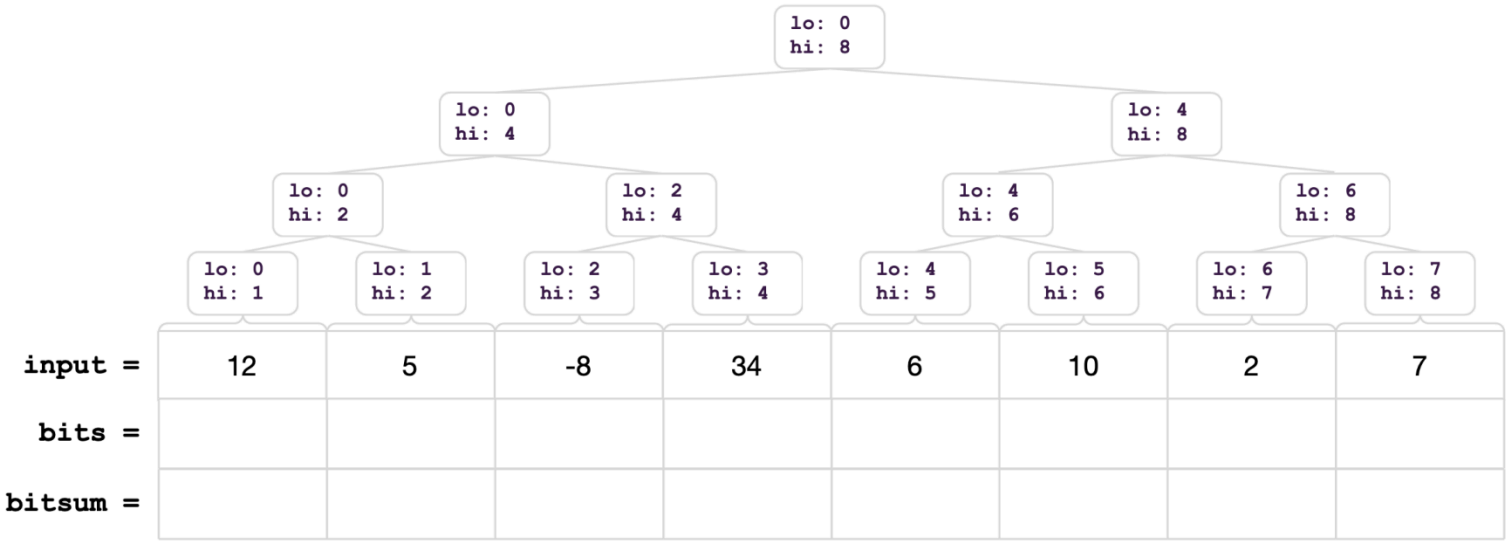
left.fromLeft = _____;
right.fromLeft = _____;

output[i] = _____;

2. Parallel Pack

Given input array `[12, 5, -8, 34, 6, 10, 2, 7]`, output an array that contains only the elements that are less than 10.

Use the Parallel Pack algorithm from lecture. Show the intermediate steps. Draw the input and output arrays, and for each step, show the tree of the recursive task objects that would be created (where a node's child is for two problems of half the size) and the fields each node needs. Do not use a sequential cut-off.



1) Compute bits:

2) Compute bitsum:

3) Produce output:

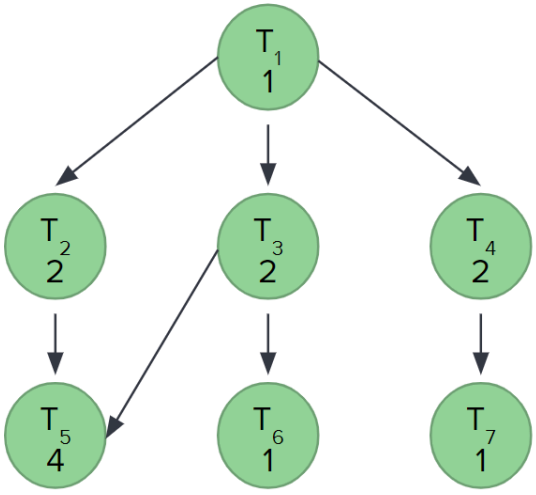
3. Work it Out [the Span]

a) Define work and span.

b) How do we calculate work and span?

c) Does adding more processors affect the work or span?

d) What is the total work and span of this task graph?



e) Suppose a program needs to do 20% of its work sequentially (the rest can be parallelized), what is the maximum speed up with 4 processors?