CMPT 125

Introduction to Computing Science and Programming II

November 3, 2021

A comment on recursion

Comment on recursion

- How is recursion really implemented inside?
  - What happens when we make a recursive call?

- Wrong answer: my laptop delegates the subtask to other laptops.

- Correct answer: the subtask is added on execution stack:
  - The order in which the functions are called,
  - Where to return after the function ends
  - All the local variables of the function

- In particular, any recursive function can be implemented *non-recursively* using stack.

Comment on recursion

- Thumb rules for writing recursion:
  - Base case: if the problem minimal/not decomposable, solve the problem directly. Checking it should be (typically) the first line of the function.
  - *Examples*:
    empty array / n=0 / index out of bounds
  - Induction case: decompose the problem into one or more similar, STRICTLY smaller sub-problems.
  - *Examples*:
    apply induction on first half of the array, then on the second half of the array
    apply induction on n-1…
  - **BAD IDEAS**: do not use static/global variables in recursion.
  - These are bad practice, and very hard to follow
  - Often fails if the function is invoked several times.

  While we are on the subject

While we are on the subject

Do not use **global** variables.

- Only exceptions: global constants

Do not use **static** variables.

- Only exceptions: when we need a shared state for objects/functions

In particular, do not use strtok()

- Ever!
- strtok() uses static variable inside.
- Why is the first call *strtok(str, s)* and then *strtok(NULL, s)*?
- Looks very suspicious.
- It uses static variable to remember the previous call.

- *In general, only use library functions you can write yourself.*

Comment on recursion

Any recursive function can be implemented *non-recursively* using stack.

Example:

Quick sort(A[0… N-1])

    pivot_ind = rearrange(A, N)

    Quick sort(A[0…pivot_ind-1])

    Quick sort(A[pivot_ind+1… N-1))

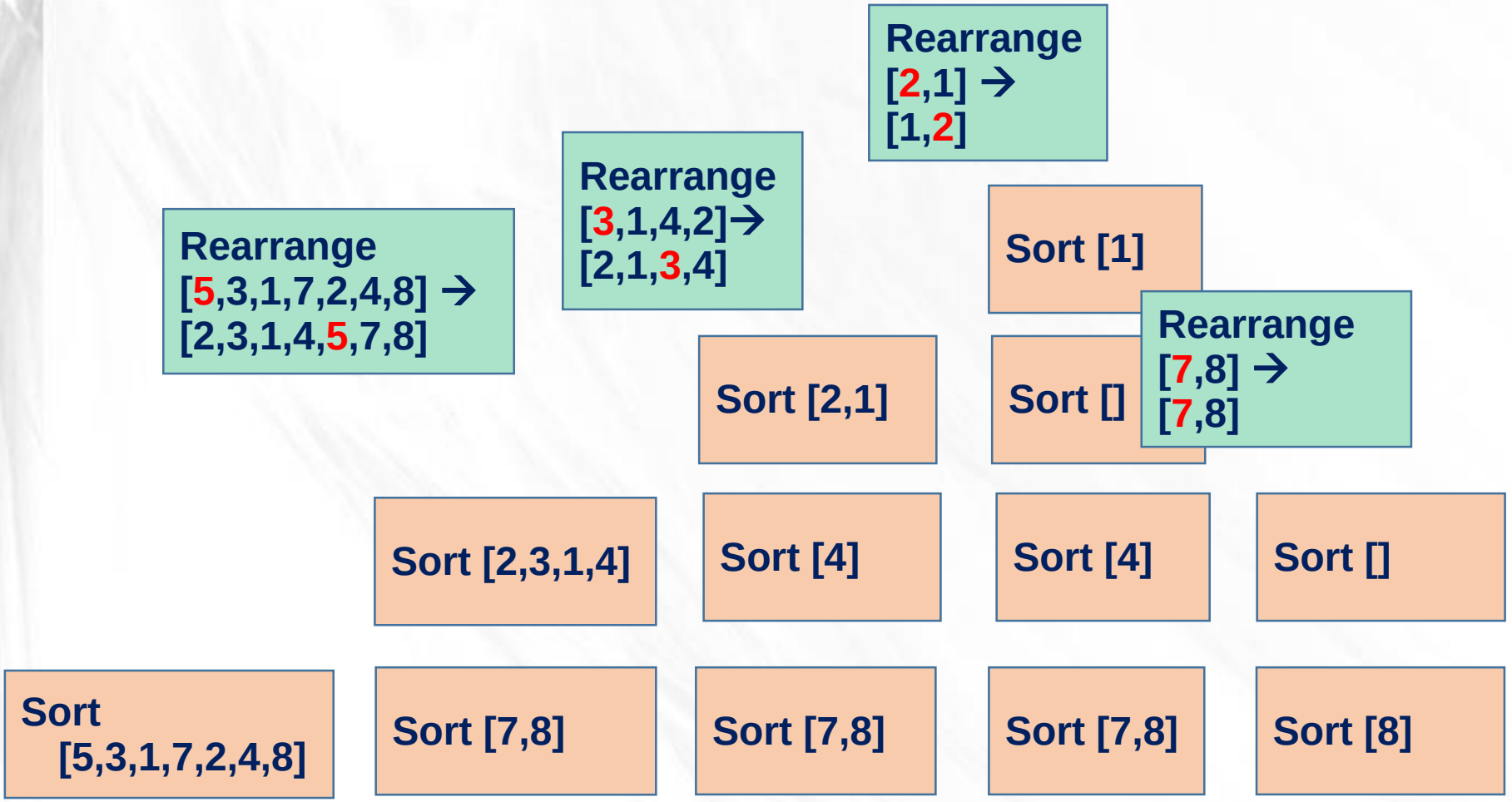
Implement this without using recursion.

Quick sort without using recursion

```
Quick sort( A[0… N-1] )  {

    // s will contain the indices of subarrays that we need to be sorted
    s = create_stack();  // stack of pairs of indices

    stack_push(s, (pair){0 , N-1});

    while (s is not empty)  {

        ( i , j ) = stack_pop(s);

        pivot_ind = rearrange( A[i…j], pivot_ind); //

        if (pivot_ind+1 < j)
            stack_push(s, (pair){pivot_ind+1 , j});

        if (i < pivot_ind-1)
            stack_push(s,  (pair){i , pivot_ind-1} );

    }

}
```

Homework: run this algorithm on several examples of arrays
In each step follow the state of the array and the state of the stack

Quick sort without using recursion



Questions?

Comments?