

National University of Singapore
School of Computing
CS1101S: Programming Methodology
Semester I, 2021/2022

S10 Searching and Sorting II; Memoization

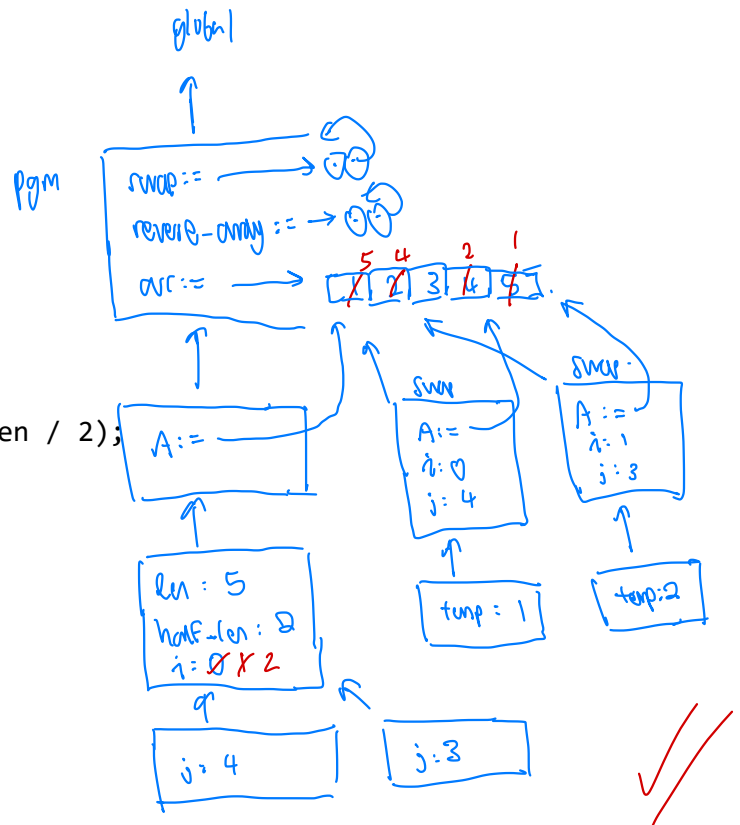
Problems:

1. Consider the following [Source program](#):

```
function swap(A, i, j) {
  let temp = A[i];
  A[i] = A[j];
  A[j] = temp;
}

function reverse_array(A) {
  const len = array_length(A);
  const half_len = math_floor(len / 2);
  let i = 0;
  while (i < half_len) {
    const j = len - 1 - i;
    swap(A, i, j);
    i = i + 1;
  }
}

const arr = [1, 2, 3, 4, 5];
reverse_array(arr);
arr;
```



Draw the diagram to show the environment during the evaluation of the program. Show all the frames that are created during the program evaluation. Show the final value of each binding. Note that calls of primitive functions, such as `array_length` and `math_floor`, do not create any frames.

2. The following function, `bubblesort_array`, is an implementation of the **Bubble Sort** algorithm to sort an array of numbers into ascending order:

```
function bubblesort_array(A) {
  const len = array_length(A);
  for (let i = len - 1; i >= 1; i = i - 1) {
    for (let j = 0; j < i; j = j + 1) {
      if (A[j] > A[j + 1]) {
        const temp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = temp;
      }
    }
  }
}
```

Handwritten notes and diagrams:

- len-1 to 1 → n times.*
- 0 to i*
- decreasing linearly.*
- not division.*
- Complexity analysis:*

$$(n-1) + (n-2) + \dots + 1 = (1+2+\dots+(n-1)) + n = \frac{n(n-1)}{2} + n = \frac{n^2 - n + 2n}{2} = \frac{n^2 + n}{2} = O(n^2)$$

- (a) What is the order of growth of its runtime for an input array of n elements?

$O(n^2)$

- (b) Write the function, `bubblesort_list`, that takes as argument a **list** of numbers and uses the bubble sort algorithm to sort the list into ascending order. Your function **must not create any new pair or array**, and **must not use the function `set_tail`**. Its runtime must have the same order of growth as that of `bubblesort_array`.

```
function bubblesort_list(L) {
  // ???
}
```

Example use:

```
const LL = list(3, 5, 2, 4, 1);
bubblesort_list(LL);
LL; // should show [1, [2, [3, [4, [5, null]]]]
```

Handwritten notes and diagrams:

- swap whenever it is bigger than any elem in the list.*
- inefficient + $O(n^2)$.*
- pointer reset to head of list.*
- Complexity analysis:*

$$(n-1) + (n-2) + \dots + 1 = \frac{n(n-1)}{2} = O(n^2)$$

3. Consider the cc (coin change) function presented in Lecture L3:

```
function cc(amount, kinds_of_coins) {
  return amount === 0
    ? 1
    : amount < 0 || kinds_of_coins === 0
    ? 0
    : cc(amount, kinds_of_coins - 1)
      +
      cc(amount - first_denomination(kinds_of_coins),
          kinds_of_coins);
}

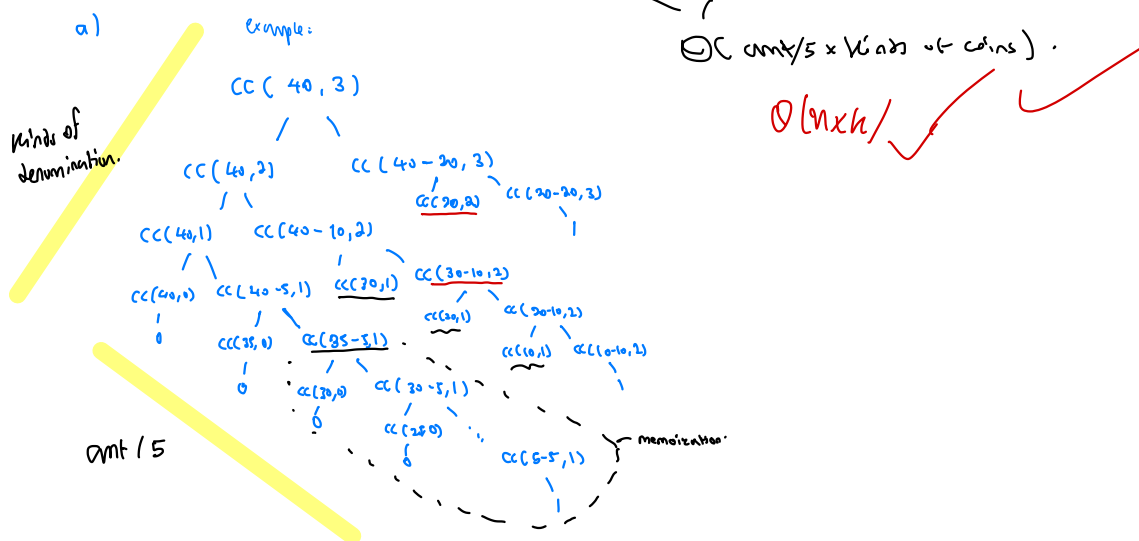
function first_denomination(kinds_of_coins) {
  return kinds_of_coins === 1 ? 5 :
    kinds_of_coins === 2 ? 10 :
    kinds_of_coins === 3 ? 20 :
    kinds_of_coins === 4 ? 50 :
    kinds_of_coins === 5 ? 100 : 0;
}
```

(a) Is function `cc` a good candidate for memoization? Support your answer with an example.

Yes. Ceiling of some parameters.

(b) If memoization is suitable for function cc, provide the **implementation**.

(c) What are the orders of growth in *time* and *space* of the memoized version?



6) let arr = [];

```

function cc = memm ( amb, kinds-of-vals ) {
    if ( amb [ kinds-of-vals ] ) {
        return amb [ amb ] [ kinds-of-vals ];
    } else {
        amb [ amb ] = [];
        if ( amb == 0 ) {
            amb [ amb ] [ kinds-of-vals ] = 1;
            return 1;
        } else if ( amb < 0 || kinds-of-vals == 0 ) {
            amb [ amb ] [ kinds-of-vals ] = 0;
            return 0;
        }
        cc = memm ( amb [ kinds-of-vals ] =
            + cc [ amb, kinds-of-vals - 1 ] )
            + cc [ amb - 1, kinds-of-vals ];
    }
}

```

Some
body.

3.

3.

3

if (! isUnder (overCount [currentIndex - 1]))

will override.

if (isUnder (overCount []))

overCount = 0

check
if
undefined.

$(0,0)$ $(0,3)$ $(3,0)$
 \swarrow \swarrow \swarrow
 $1, 2, 3, 4$ $4, 3, 2, 1$ $(0,0)$ at $(0,0)$ $(0,0)$ at $(0,0)$.
 $5, 6, 7, 8$ $8, 7, 6, 5$
 $9, 10, 11, 12 \Rightarrow 12, 11, 10, 9.$

$13, 14, 15, 16.$ $16, 15, 14, 13.$
 \swarrow \swarrow \swarrow \swarrow \swarrow \swarrow
 $(3,0)$ $(2,3)$ $(3,3)$ at $(0,3)$ at $(3,3)$.
 $(0,3)$

\searrow
 $(0,0)$ $1, 5, 9, 13$ $(0,3)$
 $2, 6, 10, 14$ Sum row and col.
 $3, 7, 11, 15$
 $4, 8, 12, 16.$
 $(3,0)$ $(3,3)$