Week 10 Studio
Searching and Sorting for Arrays, Memoisation

CS1101S AY21/22 Semester 1
Studio 05E

18 Oct 2021

Yan Xiaozhi (David)
@david_eom
yan_xiaozhi@u.nus.edu

Admin

- Contact tracing (QR code + class photo)
- Reading Assessment 2 ON THIS FRIDAY!!!
  - Get and do past year papers from LumiNUS
  - Easier than RA1 (at least imo)
  - Environment model will be heavily tested upon, practice until it becomes second nature
- PID stop mission
- My workload getting super heavy these days… 😩

Recap

Searching and Sorting for Arrays

What Is It?

- Already gone through the algos in studio 07 a few weeks ago
- Left as a challenge: implement all searching and sorting with arrays instead of lists!
  - “Ideal” searching and sorting
  - Data are usually presented in array form
  - Who uses linked lists? (actually a lot of use cases)

Searching and Sorting for Arrays

Let’s Try to Implement!

- Searching
  - Linear search
  - Binary search (recursion & loop)
- Sorting
  - Selection sort
  - Insertion sort (shifting left & shifting right)
  - Merge sort

Merge Sort for Arrays

General Idea

- ```
  function merge_sort(A) {
      merge_sort_helper(A, 0, array_length(A) - 1);
  }
  ```

- ```
  function merge_sort_helper(A, low, high) {
      if (low < high) {
          const mid = math_floor((low + high) / 2);
          merge_sort_helper(A, low, mid);
          merge_sort_helper(A, mid + 1, high);
          merge(A, low, mid, high);
      }
  }
  ```

Merge Sort for Arrays

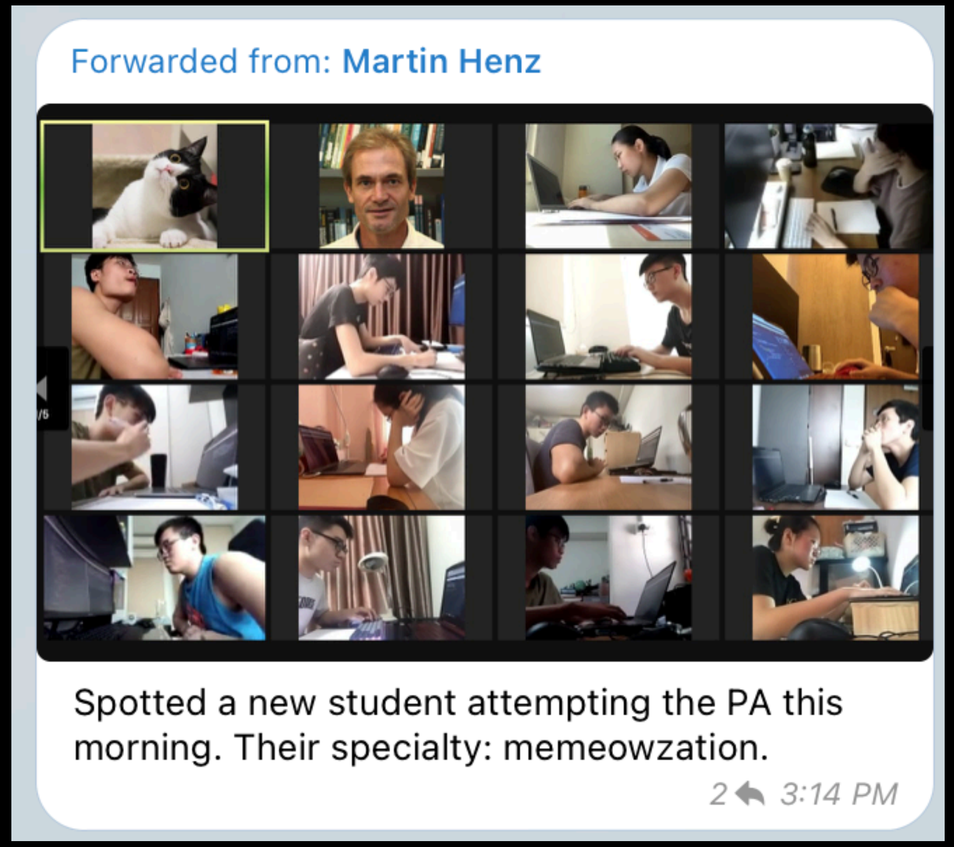
`merge` Function

```
function merge(A, low, mid, high) {
  const B = [];
  let left = low;
  let right = mid + 1;
  let Bidx = 0;

  while (left <= mid && right <= high) {
    if (A[left] <= A[right]) {
      B[Bidx] = A[left];
      left = left + 1;
    } else {
      B[Bidx] = A[right];
      right = right + 1;
    }
    Bidx = Bidx + 1;
  }

  while (left <= mid) {
    B[Bidx] = A[left];
    Bidx = Bidx + 1;
    left = left + 1;
  }

  while (right <= high) {
    B[Bidx] = A[right];
    Bidx = Bidx + 1;
    right = right + 1;
  }

  for (let k = 0; k < high - low + 1; k = k + 1)
    A[low + k] = B[k];
  }
}
```

Memoisation

What Is It?

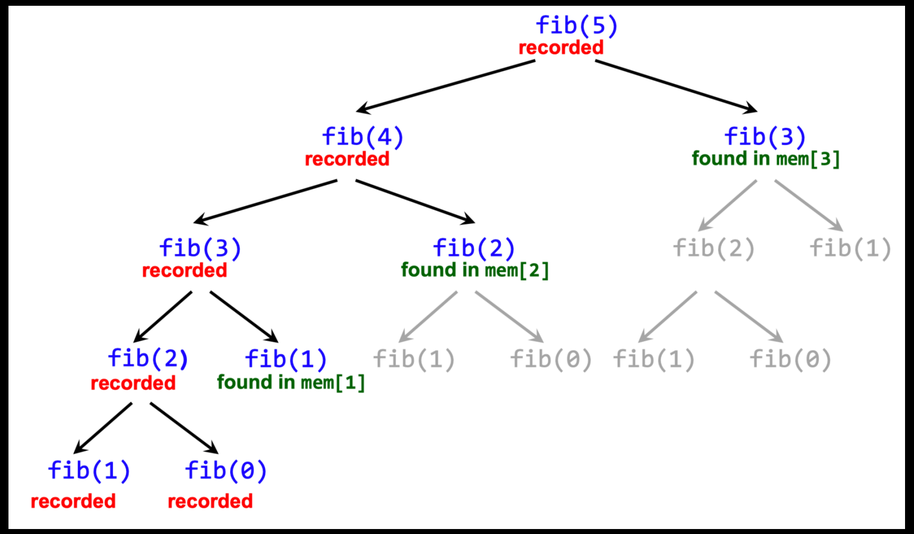
- This is NOT a typo…
- Storing results of expensive function calls and returning the cached result when the same inputs occur again
- Also known as “tabling”
- Common strategy for dynamic programming! (CS2040S nightmares)



Memoisation

How Does It Work?

- Arrays can be accessed in constant time
  - (Although lists work for memoisation as well)
- When function is called:
  - Called and stored before: return stored value
  - Not called before: calculate and store the value
- Classic examples: Fibonacci, n choose k



Memoisation

Abstraction for Memoisation

```
function memoize(f) {
    const mem = [];
    function mf(x) {
        if (mem[x] !== undefined) {
            return mem[x];
        } else {
            const result = f(x);
            mem[x] = result;
            return result;
        }
    }
    return mf;
}
```

Memoisation

What to Memoise?

- Choose only the parameters that are useful for calculation!
  - 1 parameter: O(n) space
  - 2 parameters: O(n^2) space
  - k parameters: O(n^k) space!!! 👎👎

	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
$n=0$					
$n=1$	5	4			
$n=2$	10	6	3		
$n=3$	15	11	7	2	
$n=4$		16	12	8	1
$n=5$			17	13	9
$n=6$				18	14
$n=7$					19

Memoisation

What to Memoise?

- ```
  function f(x, y, z) {
    return y === 0
      ? x
      : y + z + f(x, y - 1, z) + f(x, y - 1, z + 1);
  }
  ```

- ```
  function f(x) {
    return x === 0
      ? 1
      : x * f(x - 1);
  }
  ```

Environment Model

Empty Frames

- No empty frames!!!
  - No frame for application of nullary functions
  - No frames for blocks (function or if/else) if it does not directly declare names
  - Global environment is the same as program environment if the program itself does not directly declare any names

Environment Model

Primitive and Predeclared Functions

- Global environment as their environment (eyeball on the right)
- If program involves predeclared functions, bodies of functions will be given during exam
- Applications of primitive functions immediately return the result without any need for an environment

Environment Model

Compound values

- Arrays drawn like pairs but with multiple cells
- Empty array as a very thin box
- Values in pairs/arrays
  - Primitive values written inside the frames
  - `null`: slash instead of word
  - Direction “`/ || \`” does not matter

Any Questions?