



CS1010

Evan Tay | *evantay@comp.nus.edu.sg*

https://github.com/DigiPie/cs1010_tut_c09





Today's plan

- Unit 25: Tower of Hanoi
- Unit 26: Permutations
- Practical Exam 1
 - *Q2: Newton*
 - *Q5: Square*



TOWER OF HANOI

Unit 25

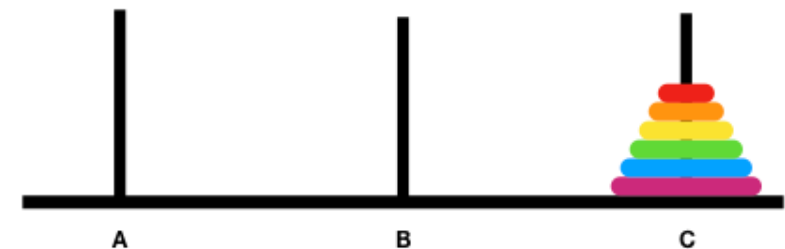
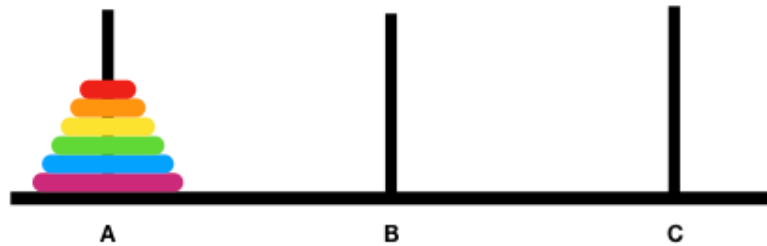


Tower of Hanoi

There are **3** pegs and **n** disks of various size that we can slide into any of the pegs. There are a few rules that we have to obey:

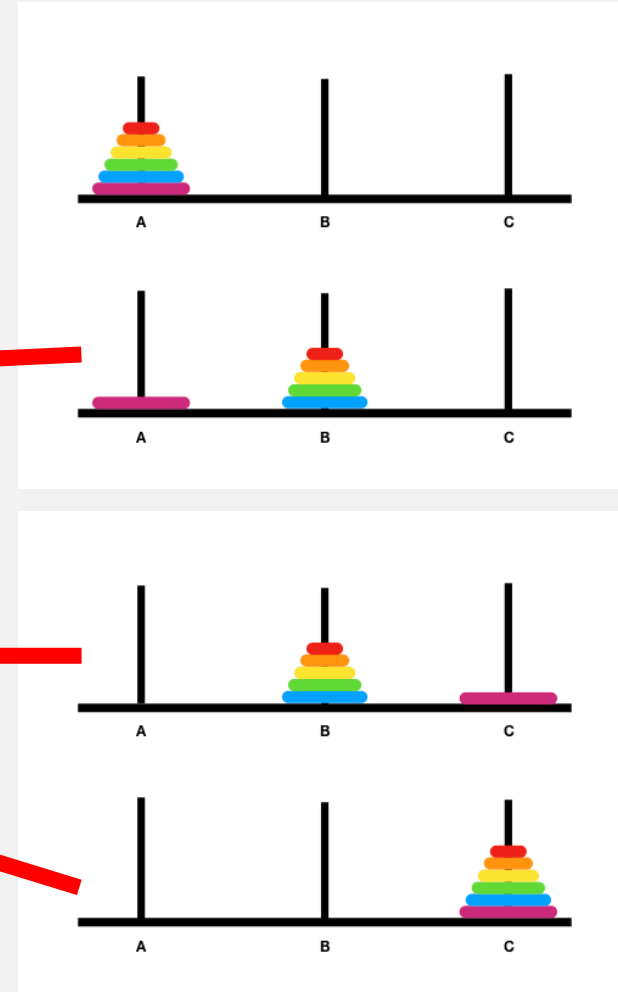
- we can only move one disk at a time;
- we can only move the topmost disk from one peg and place the disk on another peg;
- no disk can be placed on top of a smaller disk.

Tower of Hanoi



Tower of Hanoi

```
void solve(long k, long source, long dest, long placeholder) {  
    if (k == 1) {  
        print(k, source, dest);  
    }  
    else {  
        solve(k - 1, source, placeholder, dest);  
        print(k, source, dest);  
        solve(k - 1, placeholder, dest, source);  
    }  
}
```



Tower of Hanoi

```
void solve(long k, long source, long dest, long placeholder) {  
    if (k == 1) {  
        print(k, source, dest);  
    }  
    else {  
        solve(k - 1, source, placeholder, dest);  
        print(k, source, dest);  
        solve(k - 1, placeholder, dest, source);  
    }  
}
```

$$T(k) = 2T(k - 1) + 1 = 4T(k - 2) + 2 + 1 = 8T(k - 2) + 4 + 2 + 1 = \dots$$

$$T(k) = 2^k - 1 + 2^{k-1} - 1 + \dots + 4 + 2 + 1 = 2^k - 1$$

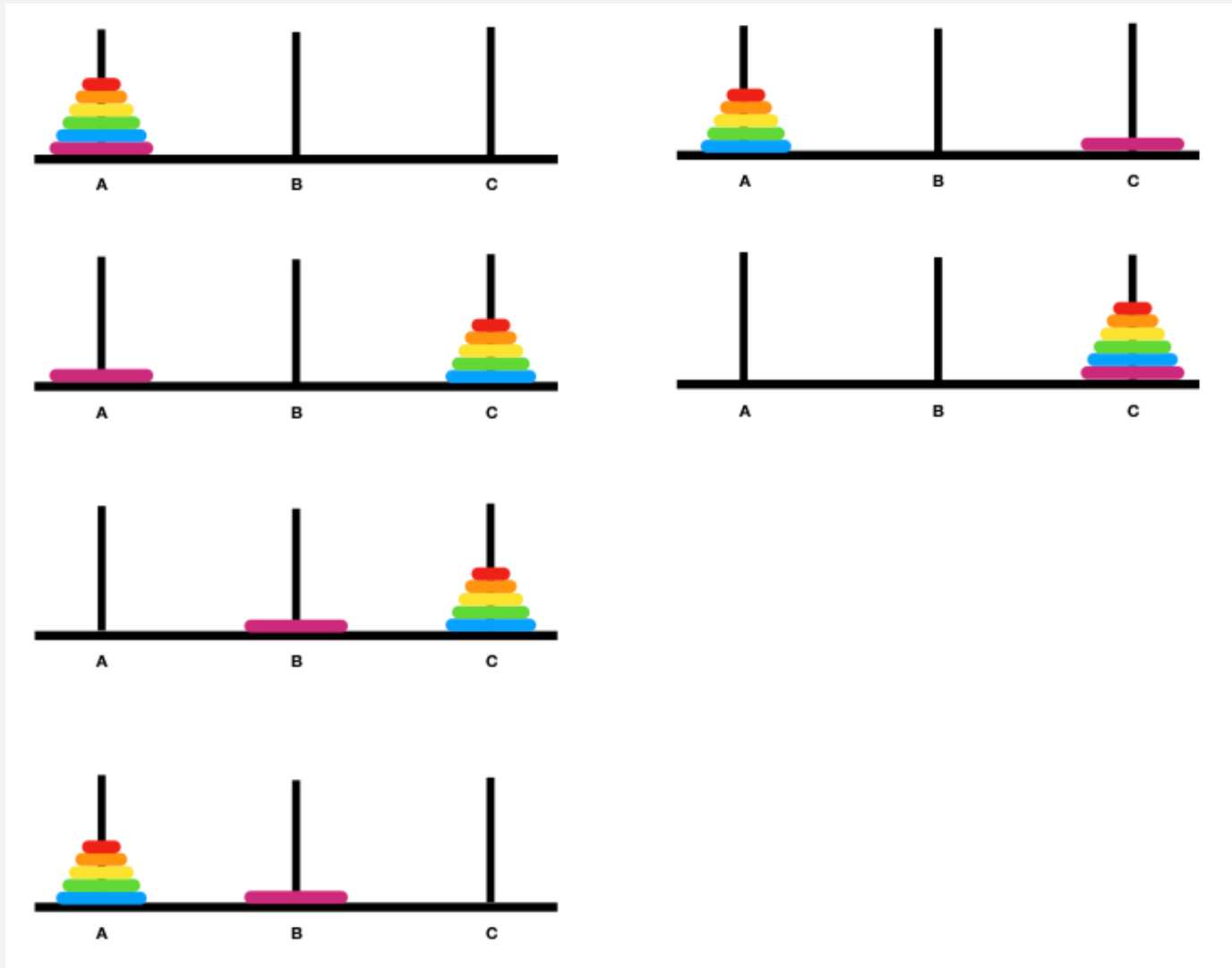
Problem Set 25.1

- Suppose that we add a new restriction to the Tower of Hanoi puzzle. Let's say that the disks are on Peg A (or Peg 1) to begin with, and we want to move the disk to Peg C (or Peg 3). We are only allowed to move a disk either to Peg B from another peg or from Peg B to another peg. In other words, we **cannot move the disks between Peg A and Peg C directly.**

Problem Set 25.1

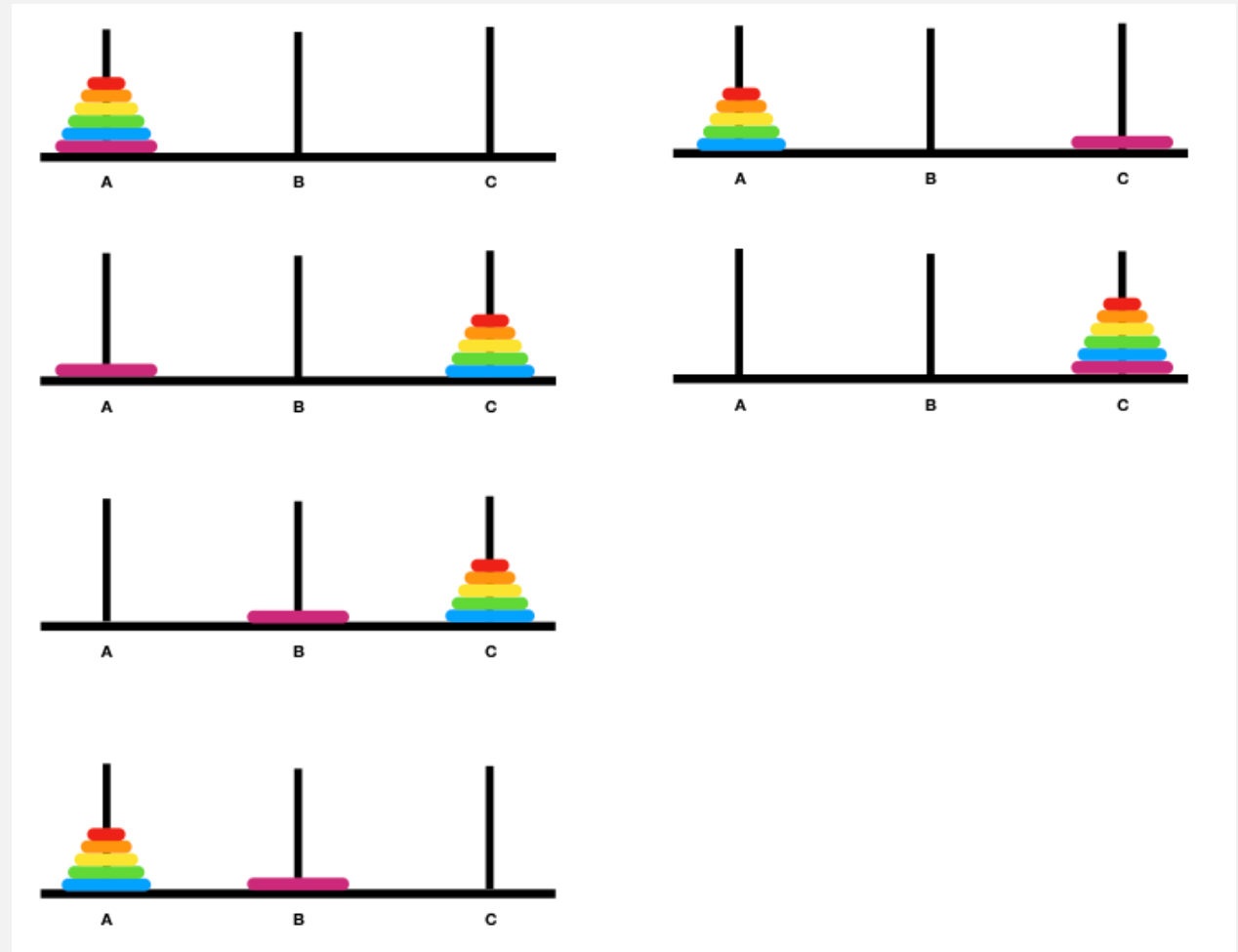
- Suppose that we add a new restriction to the Tower of Hanoi puzzle. Let's say that the disks are on Peg A (or Peg 1) to begin with, and we want to move the disk to Peg C (or Peg 3). We are only allowed to move a disk either to Peg B from another peg or from Peg B to another peg. In other words, we **cannot move the disks between Peg A and Peg C directly.**
 - *Can only move pegs between Peg A and B, and Peg B and C*

Problem Set 25.1



Problem Set 25.1

```
void solve(long k, long source, long dest, long placeholder) {  
    if (k == 1) {  
        move(k, source, placeholder);  
        move(k, placeholder, dest);  
        return;  
    }  
    solve(k - 1, source, dest, placeholder);  
    move(k, source, placeholder);  
    solve(k - 1, dest, source, placeholder);  
    move(k, placeholder, dest);  
    solve(k - 1, source, dest, placeholder);  
}
```



Problem Set 25.1

```
void solve(long k, long source, long dest, long placeholder) {  
    if (k == 1) {  
        move(k, source, placeholder);  
        move(k, placeholder, dest);  
        return;  
    }  
    solve(k - 1, source, dest, placeholder);  
    move(k, source, placeholder);  
    solve(k - 1, dest, source, placeholder);  
    move(k, placeholder, dest);  
    solve(k - 1, source, dest, placeholder);  
}
```

How many steps (use big O notation) are needed now?

From $O(2^k)$ to $O(3^k)$

$$T(k) = 3T(k-1) + 2 = 9T(k-2) + 6 + 2 + \dots = O(3^k)$$

Problem Set 25.1

```
void solve(long k, long source, long dest, long placeholder) {  
    if (k == 1) {  
        move(k, source, placeholder);  
        move(k, placeholder, dest);  
        return;  
    }  
    solve(k - 1, source, dest, placeholder);  
    move(k, source, placeholder);  
    solve(k - 1, dest, source, placeholder);  
    move(k, placeholder, dest);  
    solve(k - 1, source, dest, placeholder);  
}
```

How many steps (use big O notation) are needed now?

From $O(2^k)$ to $O(3^k)$

$$T(k) = 3T(k-1) + 2 = 9T(k-2) + 6 + 2 + \dots = O(3^k)$$

Tower of Hanoi

- Original Tower of Hanoi code:

https://github.com/DigiPie/cs1010_tut_c09/blob/master/Tutorial_10/tower_of_hanoi.c

- Problem Set 25.1 solution:

https://github.com/DigiPie/cs1010_tut_c09/blob/master/Tutorial_10/problem25_1.c



PERMUTATIONS

Unit 26



Permutations

- Consider a string length 3, **abc**. We start with **a** as the first character, and generate all the permutations of the string **bc**. We get two permutations **abc** and **acb**. The next character is **b**. We generate all permutations of the string **ac**. We get **bac** and **bca**. Similarly, we get the permutations **cab** and **cba** by considering **c** as the first character and permutating **ba**.

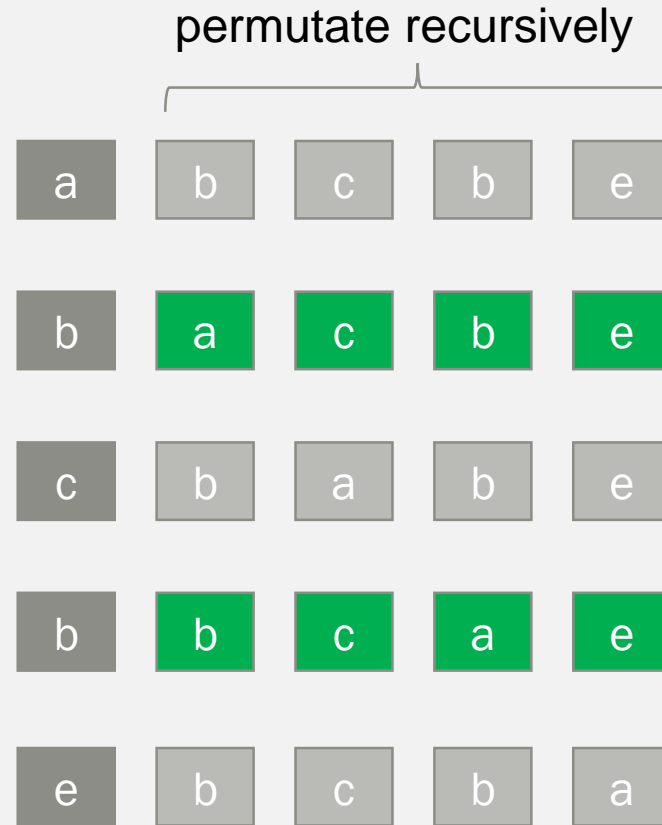
Permutations

```
void permute(char a[], long len, long curr) {  
    // permute characters a[curr]..a[len-1] and print out a for each permutation.  
    if (curr == len-1) {  
        cs1010_println_string(a);  
        return;  
    }  
  
    permute(a, len, curr + 1);  
    for (long i = curr + 1; i < len; i += 1) {  
        swap(a, curr, i);  
        permute(a, len, curr + 1);  
        swap(a, i, curr);  
    }  
}
```

Problem Set 26.1

- Write a boolean function that we can call in Line A to check if we should continue to permute the rest of the string, and therefore avoid generating duplicate permutations when the input string contains duplicate characters.
 - *For instance, if the input is aaa, the existing code would print aaa six times. Print one time only instead.*

Problem Set 26.1



Duplicate work!

Problem Set 26.1

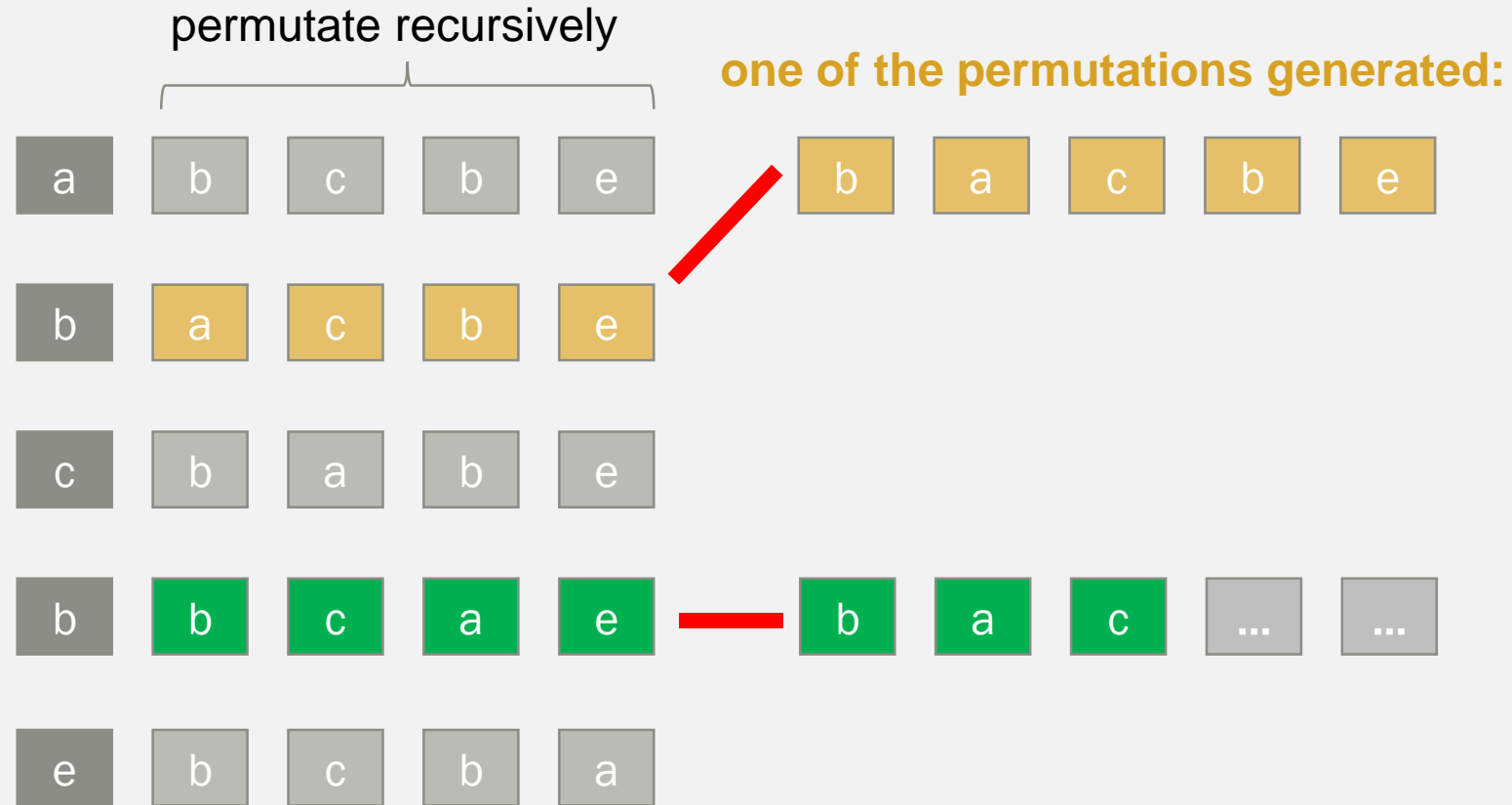
```
void permute(char a[], long len, long curr) {  
    if (curr == len-1) {  
        cs1010_println_string(a);  
        return;  
    }  
  
    permute(a, len, curr + 1);  
    for (long i = curr + 1; i < len; i += 1) {  
        if (...) { // Line A  
            swap(a, curr, i);  
            permute(a, len, curr + 1);  
            swap(a, i, curr);  
        }  
    }  
}
```

Problem Set 26.1

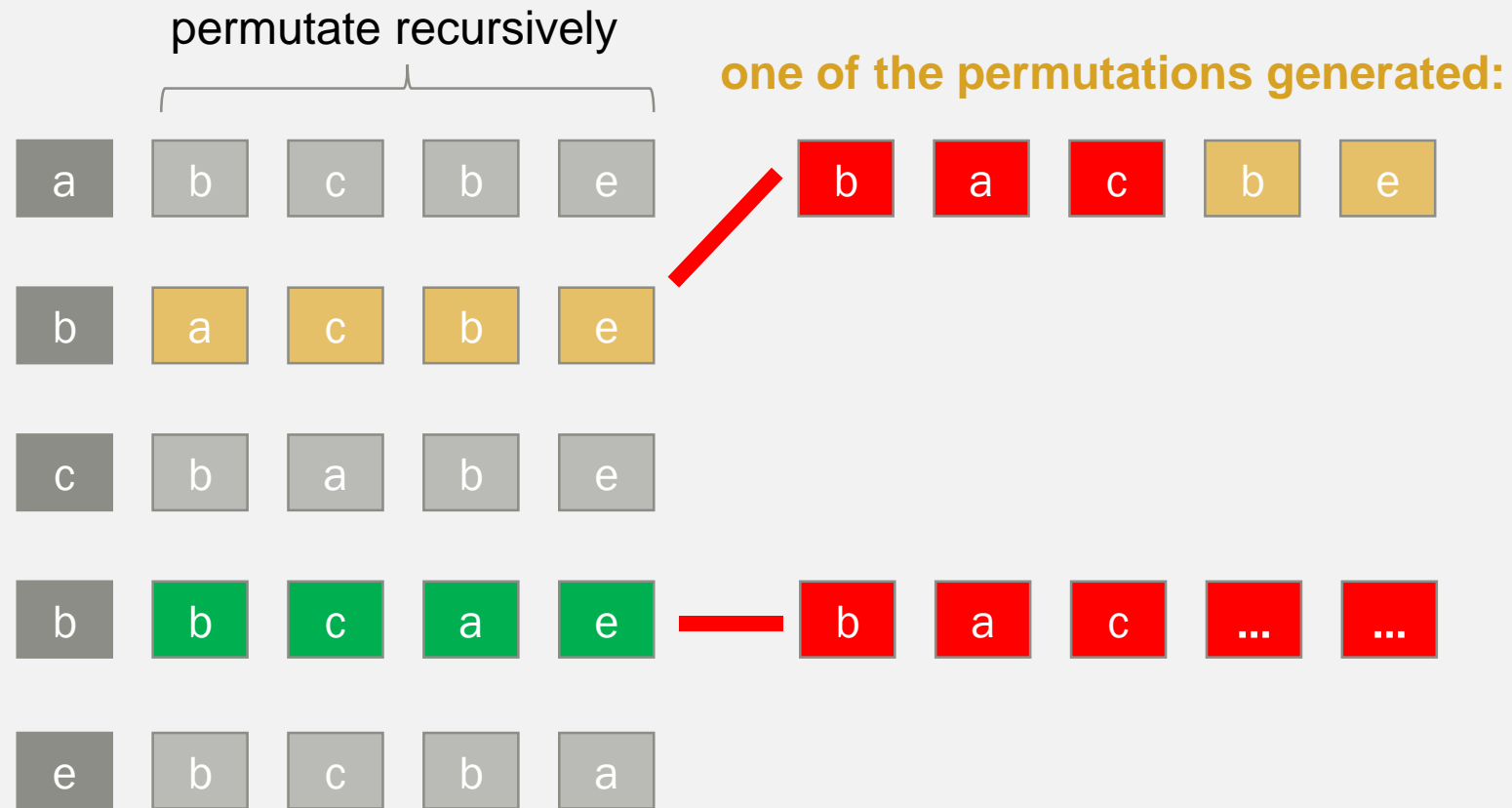
```
void permute(char a[], long len, long curr) {
    if (curr == len-1) {
        cs1010_println_string(a);
        return;
    }

    permute(a, len, curr + 1);
    for (long i = curr + 1; i < len; i += 1) {
        if (!appear_before(a, curr, i)) { // If appeared before, skip further permutation
            swap(a, curr, i);
            permute(a, len, curr + 1);
            swap(a, i, curr);
        }
    }
}
```

Problem Set 26.1



Problem Set 26.1



Problem Set 26.1



```
bool appear_before(char a[], long k, long i) {  
    for (int j = k; j < i; j += 1) {  
        if (a[j] == a[i]) {  
            return true;  
        }  
    }  
    return false;  
}
```


Permutations

- Original Permutations code:

https://github.com/DigiPie/cs1010_tut_c09/blob/master/Tutorial_10/permutations.c

- Problem Set 26.1 solution:

https://github.com/DigiPie/cs1010_tut_c09/blob/master/Tutorial_10/problem26_1.c



PRACTICAL EXAM 1

Q2 Newton. Q5 Square





THE END

Evan Tay | evantay@comp.nus.edu.sg

https://github.com/DigiPie/cs1010_tut_c09