

COMSM1201 : Data Structures & Algorithms

Dr. Neill Campbell
Neill.Campbell@bristol.ac.uk

University of Bristol

October 20, 2021



Table of Contents

N : Recursion

Simple Recursion

- When a function calls itself, this is known as recursion.
- This is an important theme in Computer Science that crops up time & time again.
- Can sometimes lead to very simple and elegant programs.
- Let's look at some toy examples to begin with.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  #define SWAP(A,B) {char temp; temp=A;A=B;B=temp;}
5
6  void strrev(char *s, int n);
7
8  int main(void)
9  {
10     char str[] = "Hello World!";
11     strrev(str, strlen(str));
12     printf("%s\n", str);
13     return 0;
14 }
15
16 /* Iterative Inplace String Reverse */
17 void strrev(char *s, int n)
18 {
19     for(int i=0, j=n-1; i<j; i++, j--){
20         SWAP(s[i], s[j]);
21     }
22 }
```

Execution :

!dlroW olleH

Recursion for *strrev()*

```
1  #include <stdio.h>
2  #include <string.h>
3
4  #define SWAP(A,B) {char temp; temp=A;A=B;B=temp;}
5
6  void strrev(char *s, int start, int end);
7
8  int main(void)
9  {
10     char str[] = "Hello World!";
11     strrev(str, 0, strlen(str)-1);
12     printf("%s\n", str);
13     return 0;
14 }
15
16 /* Recursive : Inplace String Reverse */
17 void strrev(char *s, int start, int end)
18 {
19     if(start >= end){
20         return;
21     }
22     SWAP(s[start], s[end]);
23     strrev(s, start+1, end-1);
24 }
```

Execution :

!dlroW olleH

- We need to change the function prototype.
- This allows us to track both the start and the end of the string.

The Fibonacci Sequence

A well known example of a recursive function is the Fibonacci sequence. The first term is 1, the second term is 1 and each successive term is defined to be the sum of the two previous terms, i.e. :

$\text{fib}(1)$ is 1

$\text{fib}(2)$ is 1

$\text{fib}(n)$ is $\text{fib}(n-1) + \text{fib}(n-2)$

1, 1, 2, 3, 5, 8, 13, 21, ...

Iterative & Recursive Fibonacci

```
1  #include <stdio.h>
2
3  #define MAXFIB 25
4
5  int fibonacci(int n);
6
7  int main(void)
8  {
9
10     for(int i=1; i<MAXFIB; i++){
11         printf("%d = %d\n", i, fibonacci(i));
12     }
13
14     return 0;
15 }
16
17 int fibonacci(int n)
18 {
19     if(n <= 2){
20         return 1;
21     }
22     int a = 1;
23     int b = 1;
24     int next;
25     for(int i=3; i<=n; i++){
26         next = a + b;
27         a = b;
28         b = next;
29     }
30     return b;
31 }
32 }
```

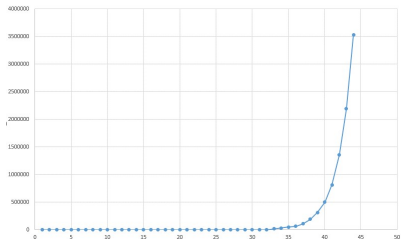
Execution :

```
1 = 1
2 = 1
3 = 2
4 = 3
5 = 5
6 = 8
7 = 13
8 = 21
9 = 34
10 = 55
11 = 89
12 = 144
13 = 233
14 = 377
15 = 610
16 = 987
17 = 1597
18 = 2584
19 = 4181
20 = 6765
21 = 10946
22 = 17711
23 = 28657
24 = 46368
```

Iterative & Recursive Fibonacci

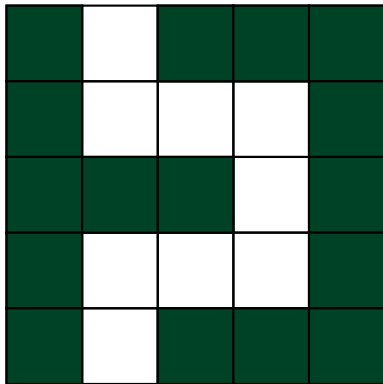
```
1  #include <stdio.h>
2
3  #define MAXFIB 25
4
5  int fibonacci(int n);
6
7  int main(void)
8  {
9
10     for(int i=1; i<MAXFIB; i++){
11         printf("%d = %d\n", i, fibonacci(i));
12     }
13
14     return 0;
15 }
16
17
18 int fibonacci(int n)
19 {
20     if(n == 1) return 1;
21     if(n == 2) return 1;
22     return( fibonacci(n-1)+fibonacci(n-2));
23 }
```

It's interesting to see how run-time increases as the length of the sequence is raised.



Maze Escape

The correct route through a maze can be obtained via recursive, rather than iterative, methods.



```
bool explore(int x, int y, char mz[YS][XS])
{
    if mz[y][x] is exit return true;

    Mark mz[y][x] so we don't return here

    if we can go up :
        if(explore(x, y+1, mz)) return true

    if we can go right :
        if(explore(x+1, y, mz)) return true

    Do left & down in a similar manner

    return false; // Failed to find route
}
```


Permuting

- Here we consider the ways to permute a string (or more generally an array)
- Permutations are all possible ways of rearranging the positions of the characters. Execution :

ABC
ACB
BAC
BCA
CBA
CAB

```
1 // From e.g. http://www.geeksforgeeks.org
2 #include <stdio.h>
3 #include <string.h>
4
5 #define SWAP(A,B) {char temp = *A; *A = *B; *B = temp;}
6
7 void permute(char *a, int s, int e);
8
9 int main()
10 {
11     char str[] = "ABC";
12     int n = strlen(str);
13     permute(str, 0, n-1);
14     return 0;
15 }
16
17 void permute(char *a, int s, int e)
18 {
19     if (s == e){
20         printf("%s\n", a);
21         return;
22     }
23     for (int i = s; i <= e; i++){
24         SWAP((a+s), (a+i)); // Bring one char to the front
25         permute(a, s+1, e);
26         SWAP((a+s), (a+i)); // Backtrack
27     }
28 }
```

Self-test : Power

- Raising a number to a power $n = 2^5$ is the same as multiple multiplications
 $n = 2*2*2*2*2$.
- Or, thinking recursively, $n = 2 * (2^4)$.

```
1  /* Try to write power(a,b) to computer a^b
2  without using any maths functions other than
3  multiplication :
4  Try (1) iterative then (2) recursive
5  (3) Trick that for  $n\%2==0$ ,  $x^n = x^{(n/2)} * x^{(n/2)}$ 
6
7  */
8
9  #include <stdio.h>
10
11 int power(unsigned int a, unsigned int b);
12
13 int main(void)
14 {
15
16     int x = 2;
17     int y = 16;
18
19     printf("%d^%d = %d\n", x, y, power(x,y));
20
21 }
22
23 int power(unsigned int a, unsigned int b)
24 {
25 }
```