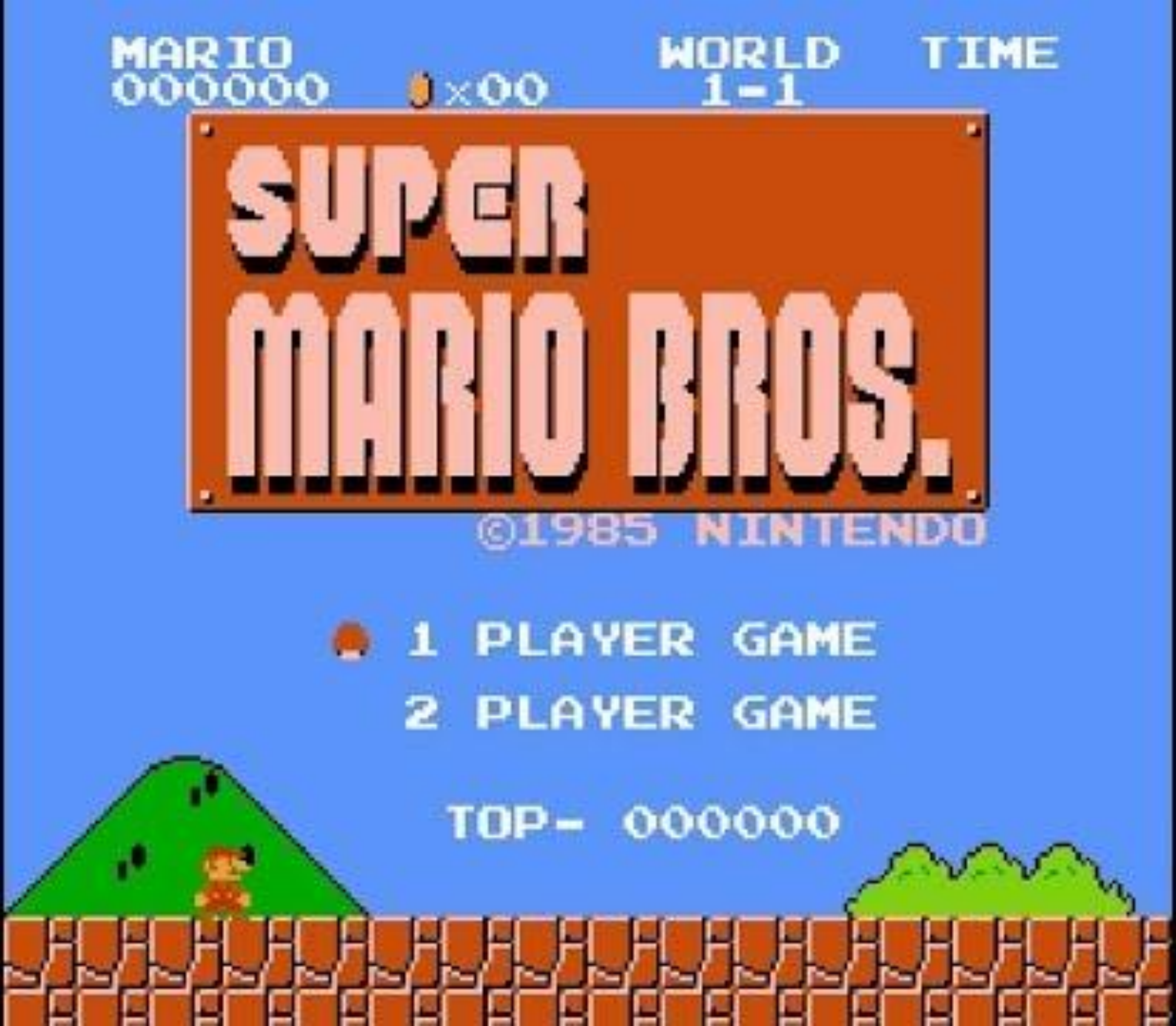
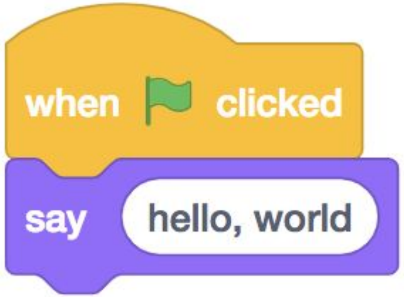
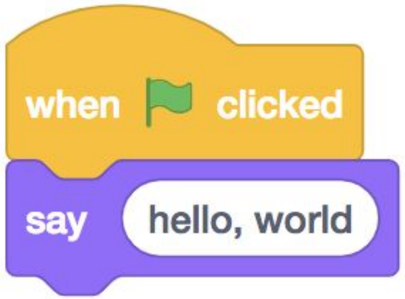
This is CS50


MARIO
000000
×00
WORLD
1-1
TIME
SUPER
MARIO BROS.
©1985 NINTENDO
1 PLAYER GAME
2 PLAYER GAME
TOP- 000000

This is CS50



```
when 🏳 clicked
say hello, world
```



```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

```c
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

```
01111111 01000101 01001100 01000110 00000010 00000001 00000001 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000010 00000000 00111110 00000000 00000001 00000000 00000000 00000000
10110000 00000101 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
11010000 00010011 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 01000000 00000000 00111000 00000000
00001001 00000000 01000000 00000000 00100100 00000000 00100001 00000000
00000110 00000000 00000000 00000000 00000101 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
00001000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000011 00000000 00000000 00000000 00000100 00000000 00000000 00000000
00111000 00000010 00000000 00000000 00000000 00000000 00000000 00000000
...
```

```
clang hello.c

./a.out
```

```
clang -o hello hello.c

./hello
```

```
make hello

./hello
```

```c
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string name = get_string("Name: ");
    printf("hello, %s\n", name);
}
```

```
clang hello.c

./a.out
```

```
clang hello.c -lcs50

./a.out
```

```
clang -o hello hello.c -lcs50

./hello
```

```
make hello

./hello
```

compiling

preprocessing

compiling

assembling

linking

preprocessing

compiling

assembling

linking

```
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("Name: ");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("Name: ");
    printf("hello, %s\n", name);
}
```

```
string get_string(string prompt);
#include <stdio.h>


int main(void)
{
    string name = get_string("Name: ");
    printf("hello, %s\n", name);
}
```

```
string get_string(string prompt);
#include <stdio.h>


int main(void)
{
    string name = get_string("Name: ");
    printf("hello, %s\n", name);
}
```

```
string get_string(string prompt);
int printf(const char *format, ...);


int main(void)
{
    string name = get_string("Name: ");
    printf("hello, %s\n", name);
}
```

```
...
string get_string(string prompt);
int printf(const char *format, ...);
...

int main(void)
{
    string name = get_string("Name: ");
    printf("hello, %s\n", name);
}
```

preprocessing

compiling

assembling

linking

```
...
string get_string(string prompt);
int printf(const char *format, ...);
...

int main(void)
{
    string name = get_string("Name: ");
    printf("hello, %s\n", name);
}
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

intel® inside

preprocessing

compiling

assembling

linking

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

```
01111111010001010100110001000110
00000010000000010000000100000000
00000000000000000000000000000000
00000000000000000000000000000000
00000001000000000011111000000000
00000001000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
10100000000000010000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000100000000000000
00001010000000000000000100000000
01010101010010001000100111100101
01001000100000111110110000010000
00110001110000001000100111000111
01001000101111110000000000000000
00000000000000000000000000000000
00000000000000001011100000000000
11101000000000000000000000000000
00000000010010001011111100000000
00000000000000000000000000000000
00000000000000000000000001001000
...
```

preprocessing

compiling

assembling

linking

hello.c

```
hello.c                  cs50.c
hello.c                    cs50.c                     stdio.c
```

hello.c cs50.c printf.c

```
01111111010001010100110001000110
00000010000000010000000100000000
00000000000000000000000000000000
00000000000000000000000000000000
00000001000000000011111000000000
00000001000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
10100000000000100000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000100000000000000
00001010000000000000000100000000
01010101010010001000100111100101
01001000100000111110110000010000
00110001110000001000100111000111
01001000101111100000000000000000
00000000000000000000000000000000
00000000000000001011000000000000
11101000000000000000000000000000
00000000010010001011111100000000
00000000000000000000000000000000
00000000000000000000000001001000
...
```

cs50.c

printf.c

```
01111111010001010100110001000110   01111111010001010100110001000110
00000010000000010000000100000000   00000010000000010000000100000000
00000000000000000000000000000000   00000000000000000000000000000000
00000000000000000000000000000000   00000000000000000000000000000000
00000001000000000011111000000000   00000011000000000011111000000000
00000001000000000000000000000000   00000001000000000000000000000000
00000000000000000000000000000000   11000000000011110000000000000000
00000000000000000000000000000000   00000000000000000000000000000000
00000000000000000000000000000000   01000000000000000000000000000000
00000000000000000000000000000000   00000000000000000000000000000000
10100000000000010000000000000000   00101000000110010000000000000000
00000000000000000000000000000000   00000000000000000000000000000000
00000000000000000000000000000000   00000000000000000000000000000000
01000000000000000000000000000000   01000000000000000001110000000000
00000000000000000100000000000000   00000111000000000100000000000000
00001010000000000000000100000000   00011100000000000001100100000000
01010101010010001000100111100101   00000001000000000000000000000000
01001000100000111110110000010000   00000101000000000000000000000000
00110001110000000100010011100011   00000000000000000000000000000000
01001000101111100000000000000000   00000000000000000000000000000000
00000000000000000000000000000000   00000000000000000000000000000000
00000000000000001011000000000000   00000000000000000000000000000000
11101000000000000000000000000000   00000000000000000000000000000000
00000000010010001011111100000000   00000000000000000000000000000000
00000000000000000000000000000000   01011100001001010000000000000000
00000000000000000000000001001000   00000000000000000000000000000000
...                                ...
```

printf.c

```
01111111010001010100110001000110   01111111010001010100110001000110   00101111011011000110100101100010
00000010000000010000000100000000   00000010000000010000000100000000   01100011001011100111001101101111
00000000000000000000000000000000   00000000000000000000000000000000   00101110001101100010000000101111
00000000000000000000000000000000   00000000000000000000000000000000   01110101011100110111001000101111
00000001000000000011111000000000   00000011000000000011111000000000   01101100011010010110001000101111
00000001000000000000000000000000   00000001000000000000000000000000   01111000001110000011011001011111
00000000000000000000000000000000   11000000000011110000000000000000   00110110001101000010110101101100
00000000000000000000000000000000   00000000000000000000000000000000   01101001011011100111010101111000
00000000000000000000000000000000   01000000000000000000000000000000   00101101011001110110111001110101
00000000000000000000000000000000   00000000000000000000000000000000   00101111011011000110100101100010
10100000000000001000000000000000   00101000001100100000000000000000   01100011010111110110111001101111
00000000000000000000000000000000   00000000000000000000000000000000   01101110011100110110100001100001
00000000000000000000000000000000   00000000000000000000000000000000   01110010011001010110010000101110
01000000000000000000000000000000   01000000000000000011100000000000   01100001001000000010000000100001
00000000000000000100000000000000   00001110000000000100000000000000   01010011010111110100111001000101
00001010000000000000000100000000   00011100000000000001100100000000   01000101010001000100010101000100
01010101010010001000100111100101   00000001000000000000000000000000   00100000001010000010000000101111
01001000100000111110110000010000   00000101000000000000000000000000   01101100011010010110001000101111
00110001110000001000100111000111   00000000000000000000000000000000   01111000001110000011011001011111
01001000101111100000000000000000   00000000000000000000000000000000   00110110001101000010110101101100
00000000000000000000000000000000   00000000000000000000000000000000   01101001011011100111010101111000
00000000000000001011000000000000   00000000000000000000000000000000   00101101011001110110111001110101
11101000000000000000000000000000   00000000000000000000000000000000   00101111011011000110010000101101
00000000010010001011111100000000   00000000000000000000000000000000   01101100011010010110111001110101
00000000000000000000000000000000   01011100001001010000000000000000   01111000001011010111100000111000
00000000000000000000000001001000   00000000000000000000000000000000   00110110001011010011011000110100
...                                ...                                ...
```

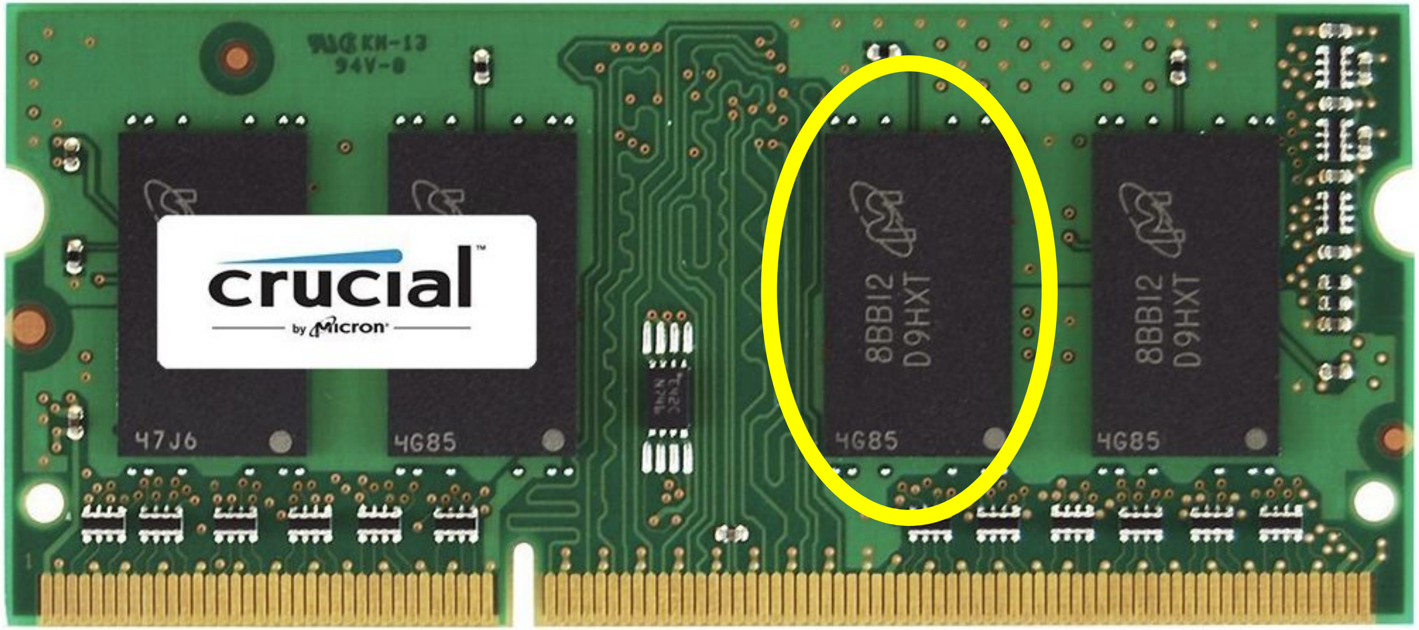
```
010111111101000101010011000100011000000010000000010000000100000000000000000000000000000000000000000000
000000000000000000000000000000000100000000001111100000000000000001000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000001010000000000010000000000000000000000000000000000000000000000000000000000000000000000000
000000000100000000000000000000000000000000000000000000000001000000000000000000101000000000000000010000000
000101010101001000100010011110010101001000100000111110110000010000011000111000000010001001110001110100
100010111110000000000000000000000000000000000000000000000000000000000000000000101100000000000011101000000
000000000000000000000000000000010010001011111100000000000000000000000000000000000000000000000000000000
0000000001001000...01111111010001010100110001000110000000100000000100000001000000000000000000000000000
000000000000000000000000000000000000000000000000000110000000000111110000000000000000100000000000000000
000000001100000000001111100000000000000000000000000000000000000000000000100000000000000000000000000000
000000000000000000000000000000000001010000011001000000000000000000000000000000000000000000000000000000
000000000000000000000000000010000000000000000111000000000000000011100000000010000000000000000011100000
000000000110010000000000000010000000000000000000000000000010100000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000001011100001001010000000000000
00000000000000000000000000000000000...0010111101101100011010010110001001100011001011100111001101101111
001011100011011000100000001011110111010101110011011100100010111101101100011010010110001000101111011110
000011100000110110010111110011011000110100001011010110110001101001011011100111010101111000001011010110
011101101110011101010010111101101100011010010110001001100011010111110110111001101111011011100111001101
101000011000010111001001100101011001000010111001100001001000000010000001000001010100110101111101001110
010001010100010101000100010001010100010000100000001010000010000000101111011011000110100101100010001011
110111100000111000001101100101111100110110001101000010110101101100011010010110111001110101011110000010
110101100111011011100111010100101111011011000110010000101101011011000110100101101110011101010111100000
101101011110000011100000110110001011010011011000110100...
```

Intel® Core™ i7 processor

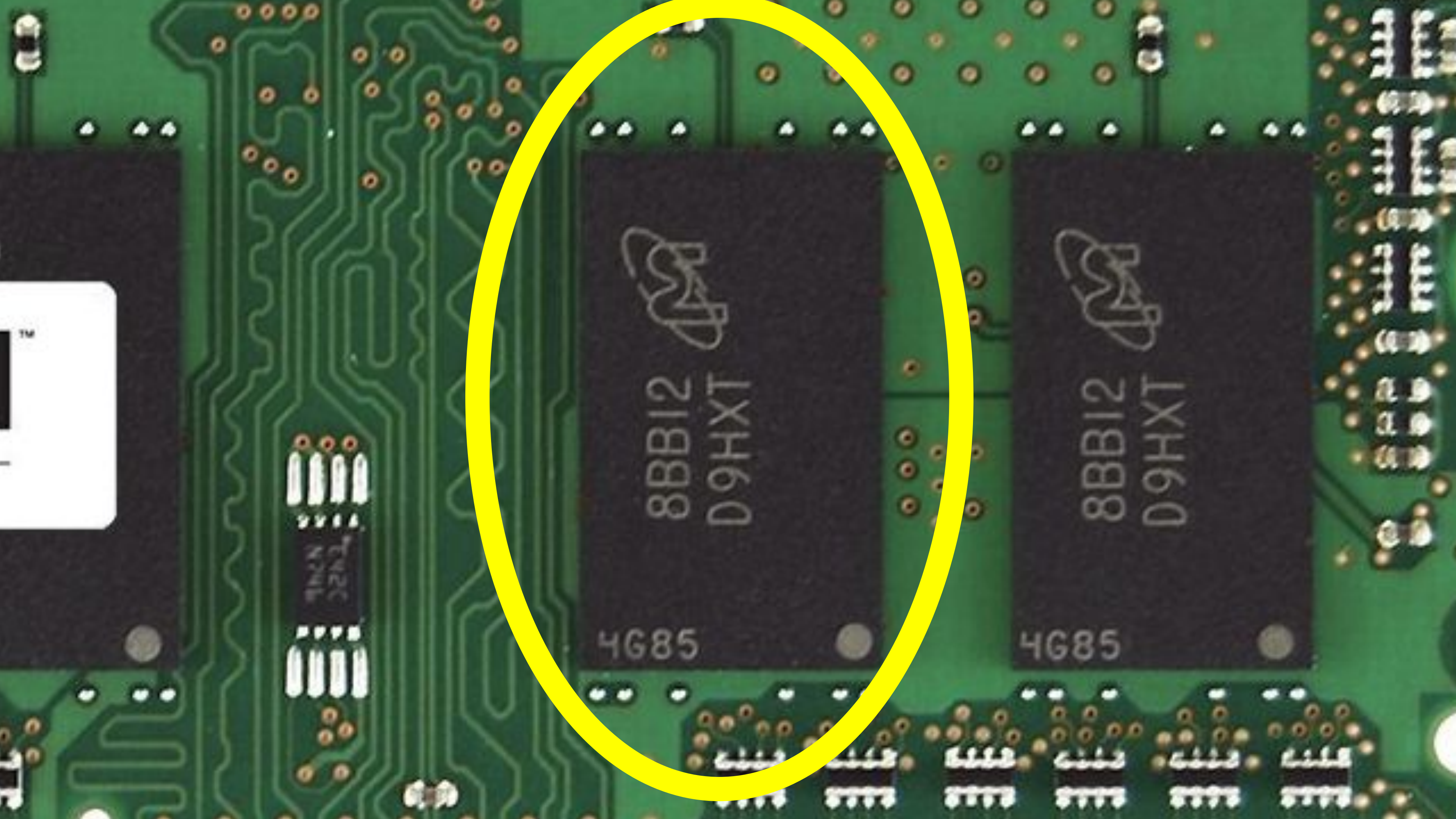
```
help50

printf

style50
```

crucial
by Micron
KN-13
94V-0
47J6
4G85
8BB12
D9HXT
4G85
8BB12
D9HXT
4G85
4G85

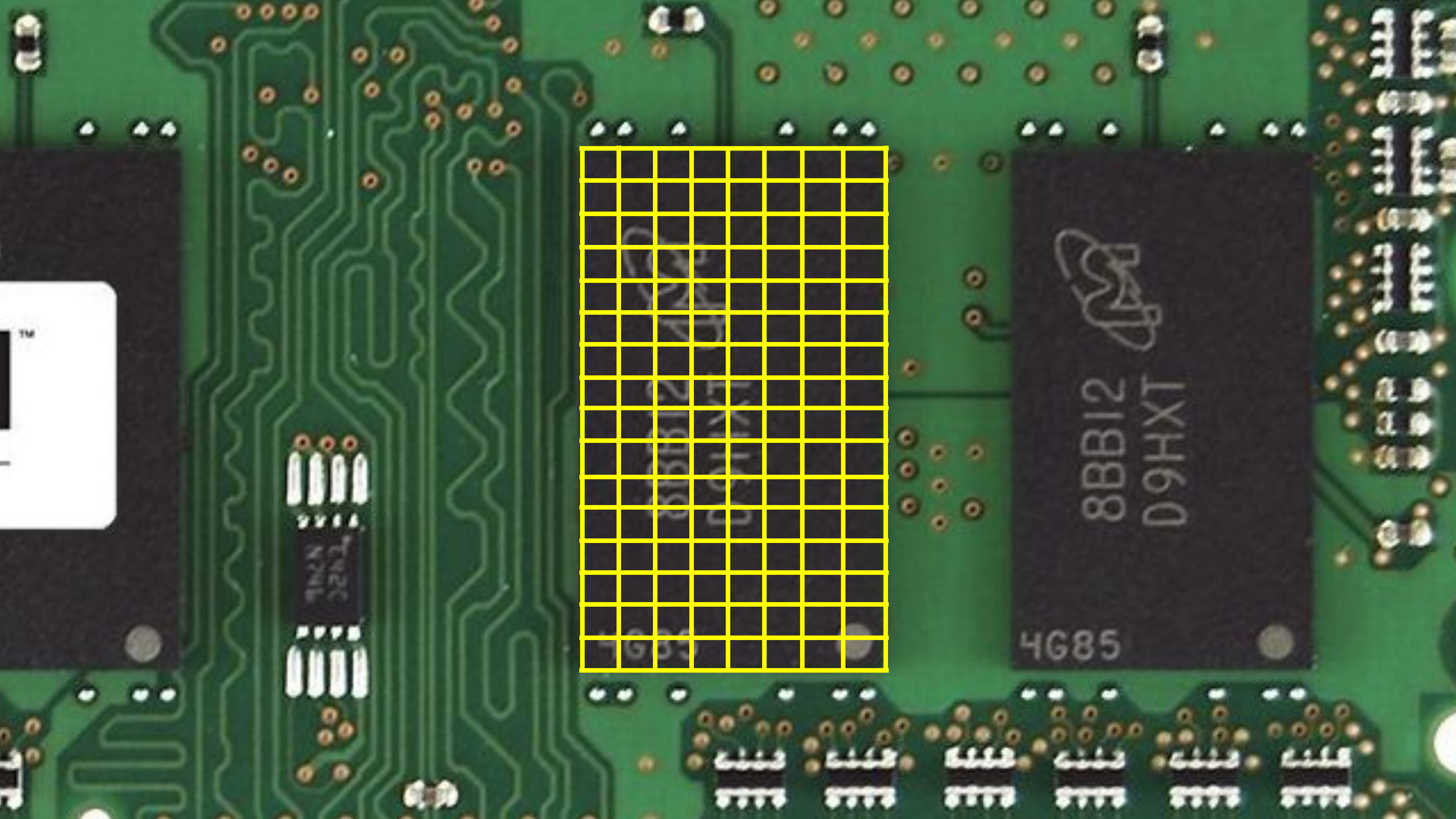
amazon.com

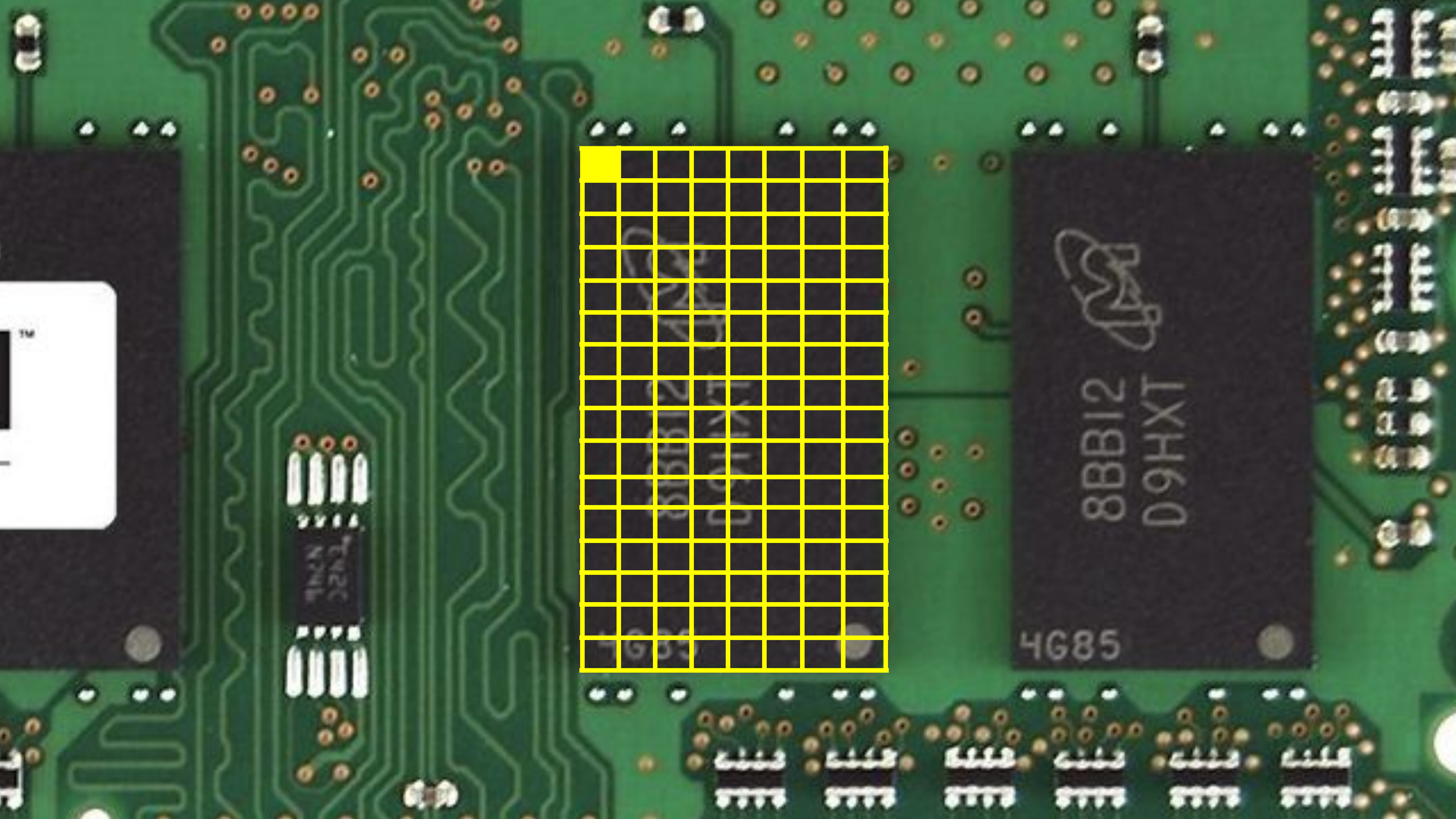


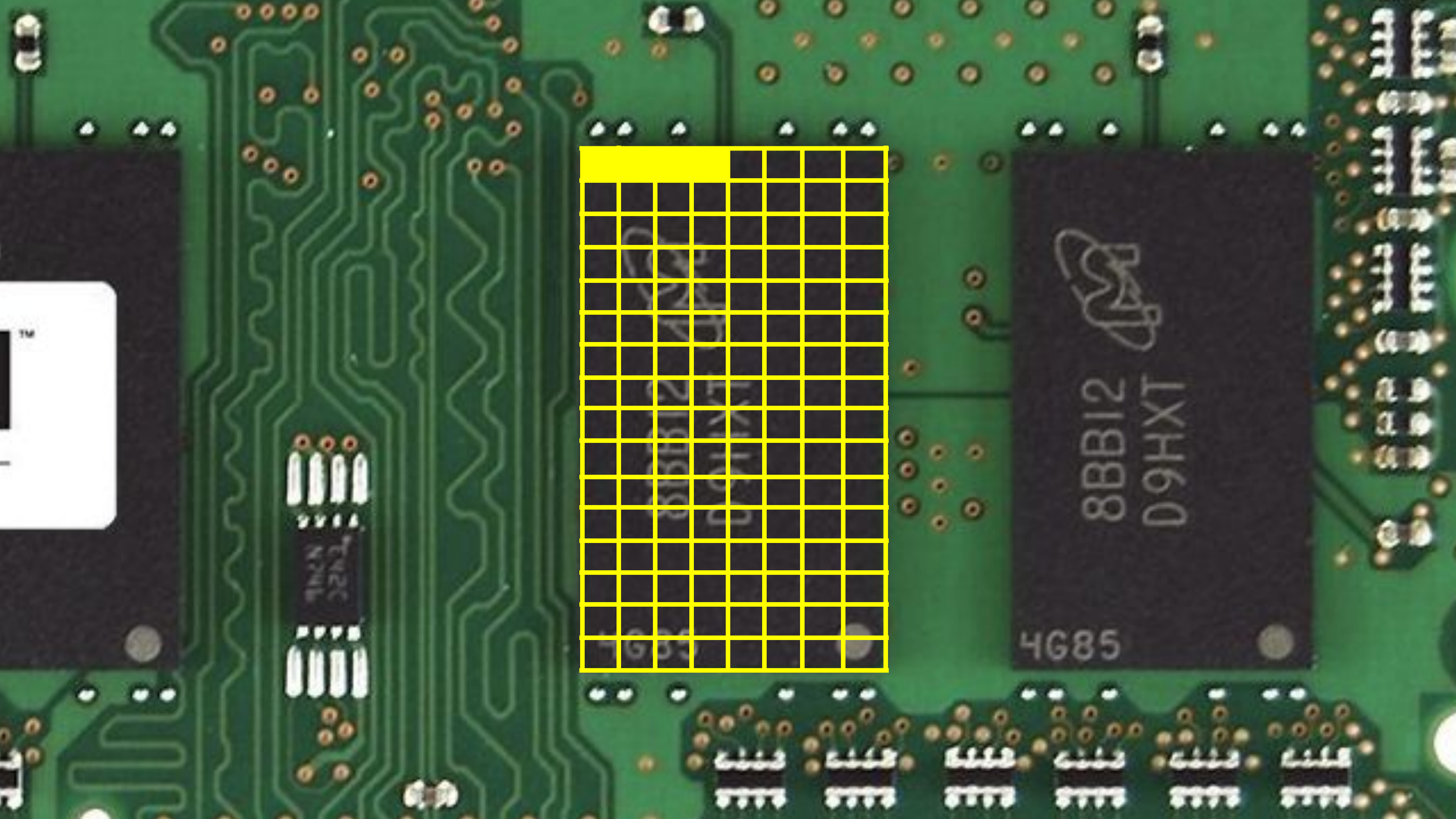
amazon.com

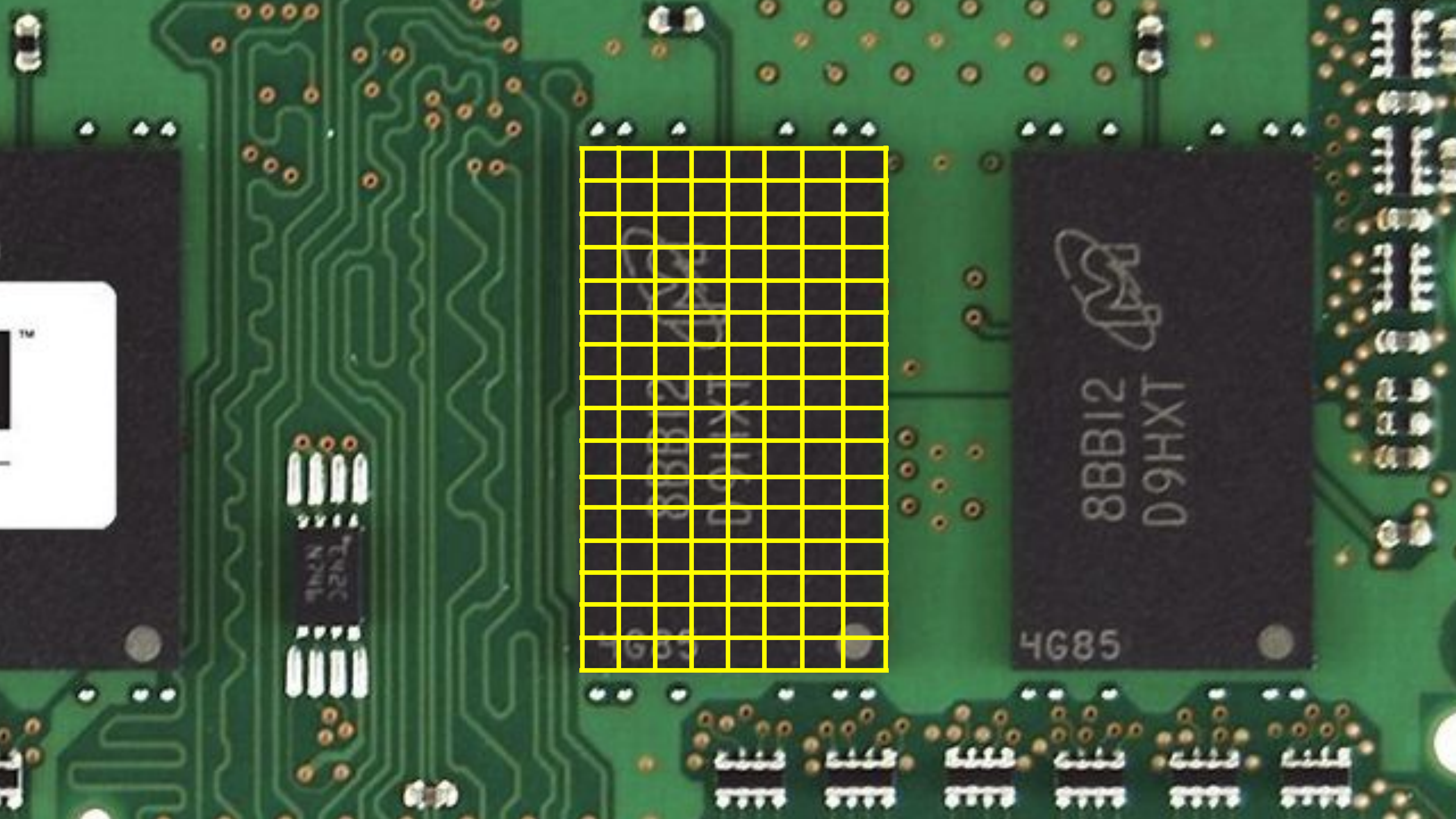

8BB12
D9HXT
4G85
8BB12
D9HXT
4G85

8BB12
D9HXT
4G85
8BB12
D9HXT
4G85

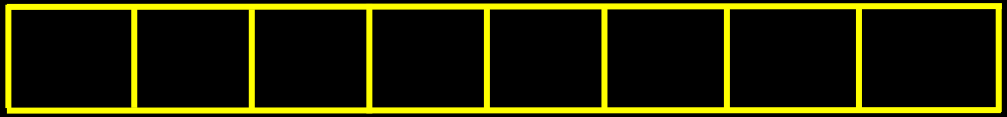

8BB12
D9HXT
4G85
8BB12
D9HXT
4G85


8BB12
D9HXT
4G85


8BB12
D9HXT
4G85

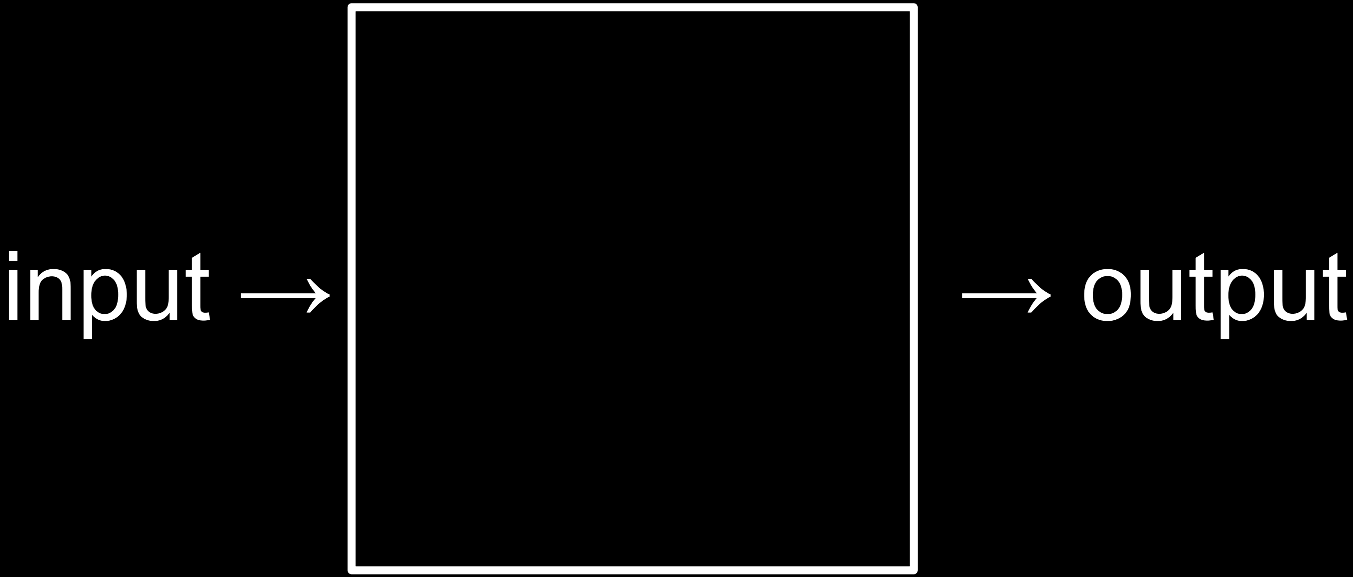

8BB12
D9HXT
4G85
TM

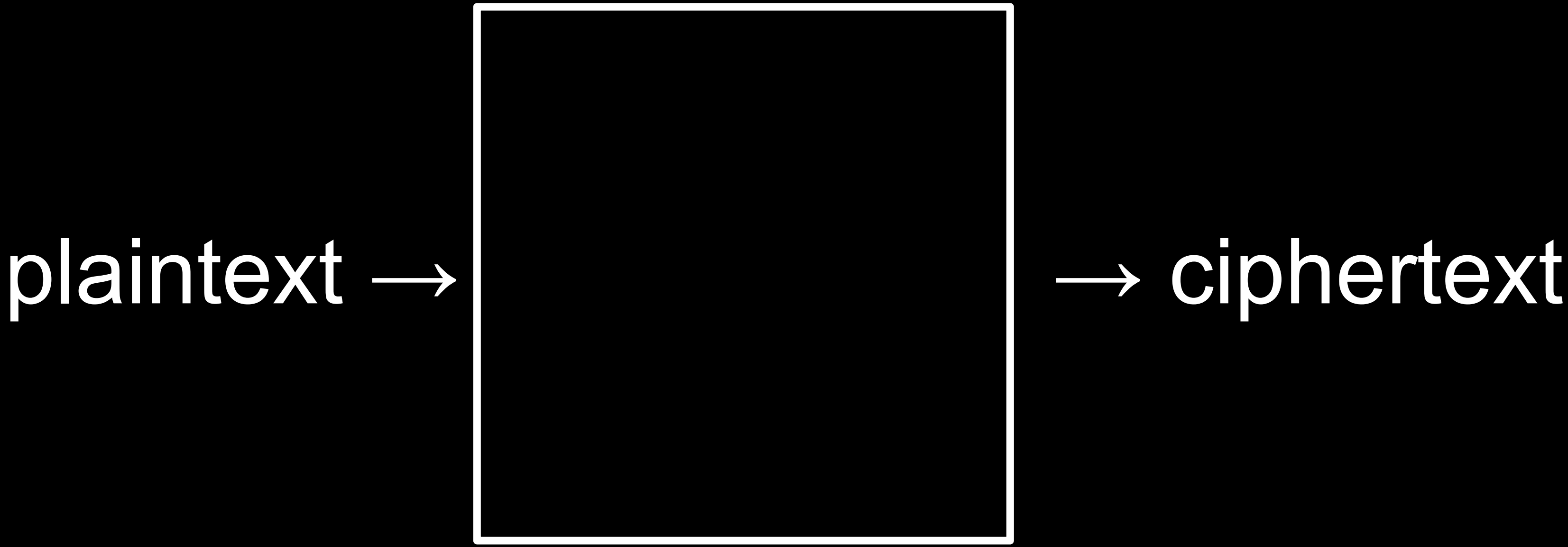
[illegible]



array

string


input →
→ output


plaintext →
→ ciphertext

H I !

72 73 33

CLASS OF SERVICE DESIRED	
Fast Day Message	✓
Day Letter	
Night Message	
Night Letter	

Patrons should mark an X opposite the class of service desired; OTHERWISE THE TELEGRAM WILL BE TRANSMITTED AS A FAST DAY MESSAGE.

CHARGE 8587

WESTERN UNION TELEGRAM

NEWCOMB CARLTON, PRESIDENT

MC

Receiver's No. 2/5

Check 5500

Time Filed

716 Od German Emb

Send the following telegram, subject to the terms on back hereof, which are hereby agreed to

via Galveston

Via Galveston

JAN 19 1917

862.20212/82A

GERMAN LEGATION

MEXICO CITY

```
130    13042  13401  8501   115    3528   416    17214  6491   11310
18147  18222  21560  10247  11518  23677  13605  3494   14936
98092  5905   11311  10392  10371  0302   21290  5161   39695
23571  17504  11269  18276  18101  0317   0228   17694  4473
22284  22200  19452  21589  67893  5569   13918  8958   12137
1333   4725   4458   5905   17166  13851  4458   17149  14471  6706
13850  12224  6929   14991  7382   15857  67893  14218  36477
5870   17553  67893  5870   5454   16102  15217  22801  17138
21001  17388  7446   23638  18222  6719   14331  15021  23845
3156   23552  22096  21604  4797   9497   22464  20855  4377
23610  18140  22260  5905   13347  20420  39689  13732  20667
6929   5275   18507  52262  1340   22049  13339  11265  22295
10439  14814  4178   6992   8784   7632   7357   6926   52262  11267
21100  21272  9346   9559   22464  15874  18502  18500  15857
2188   5376   7381   98092  16127  13486  9350   9220   76036  14219
5144   2831   17920  11347  17142  11264  7667   7762   15099  9110
10482  97556  3569   3670
```

BERNSTORFF.

INSPECTED

Charge German Embassy.

4458	gemeinsam
17149.	Friedensschluß.
14471	⊙
6706	reichlich
13850	finanziell
12224	unterstützung
6929	und
14991	Einverständnis
7382	unsererseits.
158(5)7	daß
67893	Mexico.
14218	in
36477	Texas
5870	⊙
17553	neu
67893	Mexico.
5870	⊙
5454	AR
16102	IZ
15217	ON
22801	A

like a pall on sculpture, till another man took the burden from him and went up to the house with his dead.

When Mr. Raleigh entered the house again, it was at break of dawn. Some one opened the library-door and beckoned him in. Marguerite sprang into his arms.

"What if she had died?" said Mrs. Purcell, with her swift satiric breath, and folding a web of muslin over her arm. "See! I had got out the shroud. As it is, we drink *skål* and say grace at breakfast. The funeral baked-meats shall coldly furnish forth the marriage-feast. You men are all alike. *Le Roi est mort! Vive la Reine!*"

PAUL REVERE'S RIDE.

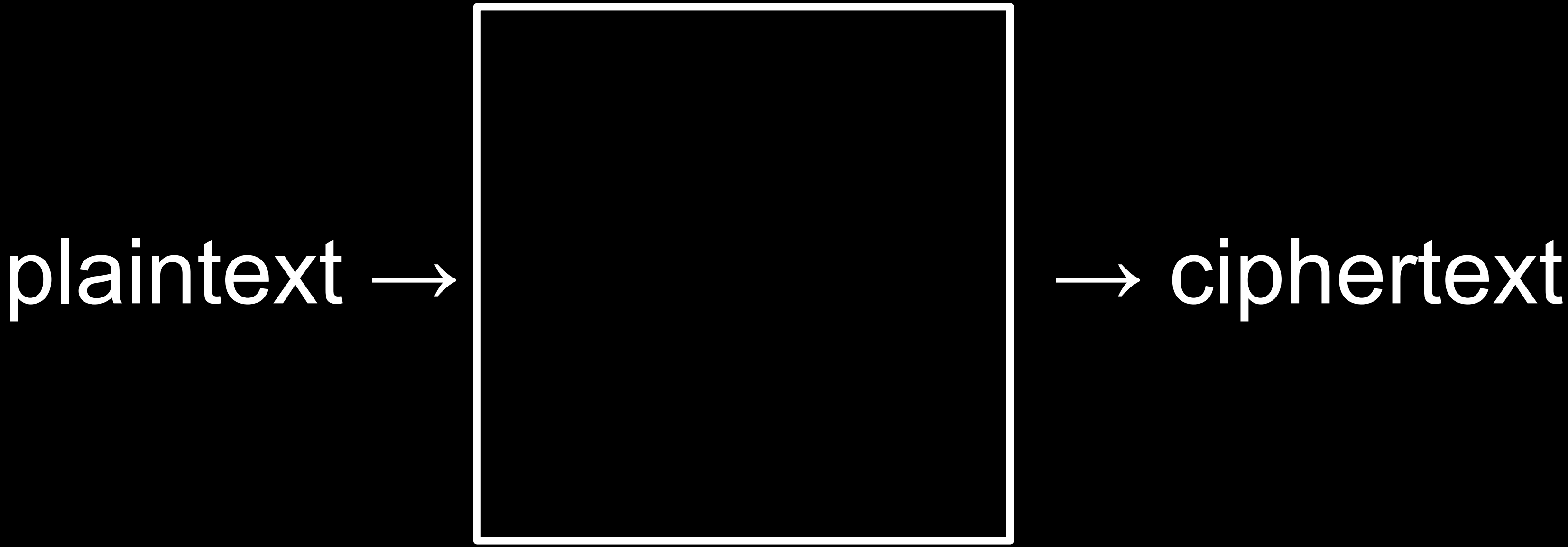
LISTEN, my children, and you shall hear
Of the midnight ride of Paul Revere,
On the eighteenth of April, in Seventy-Five:
Hardly a man is now alive
Who remembers that famous day and year.

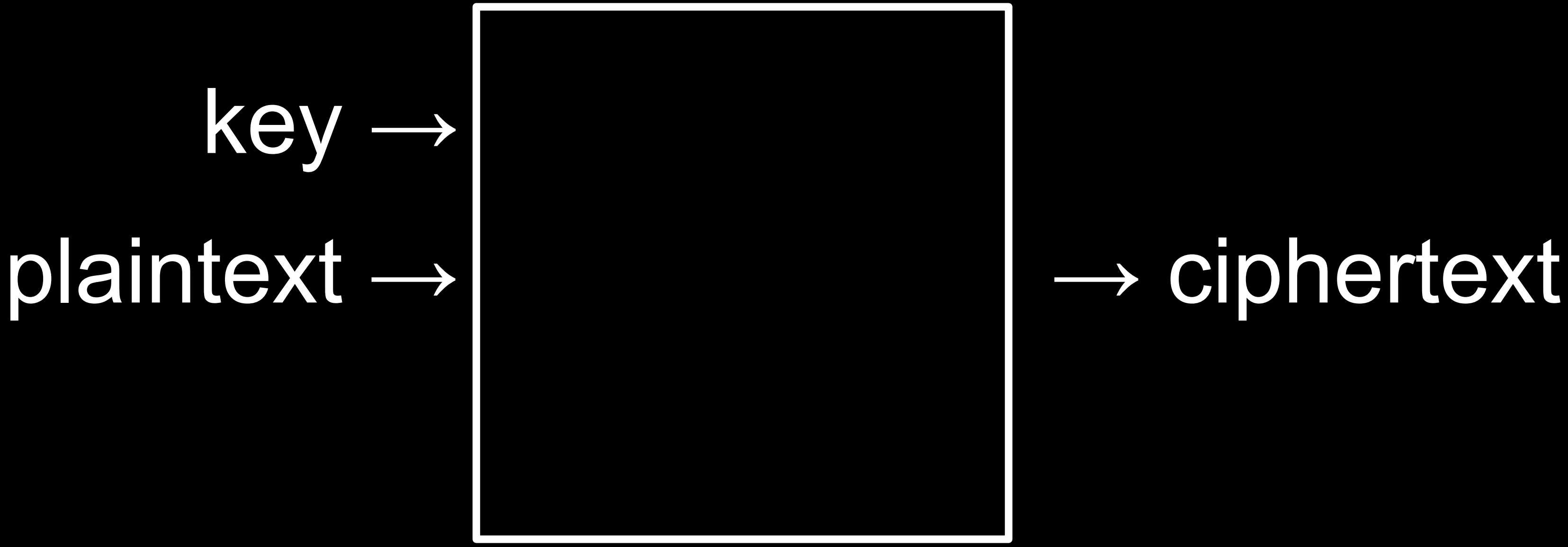
He said to his friend, — "If the British march
By land or sea from the town to-night,
Hang a lantern aloft in the belfry-arch
Of the North-Church-tower, as a signal-light, —
One if by land, and two if by sea;
And I on the opposite shore will be,
Ready to ride and spread the alarm
Through every Middlesex village and farm,
For the country-folk to be up and to arm."

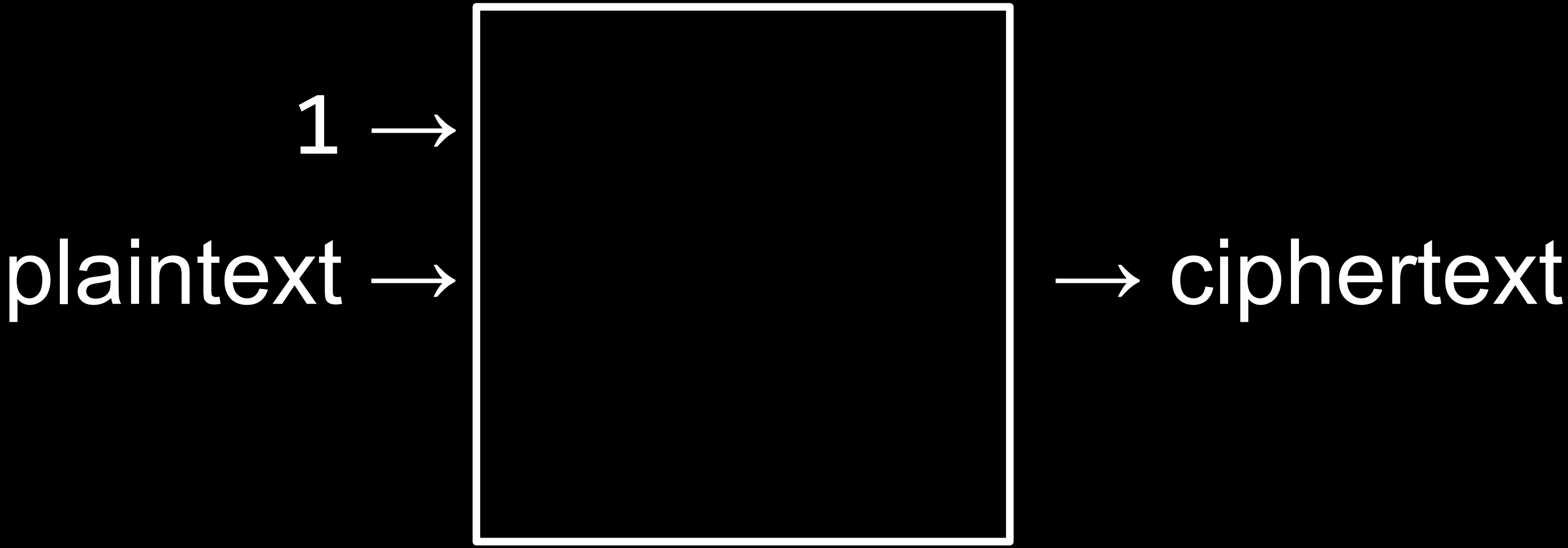
Then he said good-night, and with muffled oar
Silently rowed to the Charlestown shore,
Just as the moon rose over the bay,
Where swinging wide at her moorings lay
The Somersett, British man-of-war:
A phantom ship, with each mast and spar
Across the moon, like a prison-bar,
And a huge, black hulk, that was magnified
By its own reflection in the tide.

Meanwhile, his friend, through alley and street
Wanders and watches with eager ears,
Till in the silence around him he hears
The muster of men at the barrack-door,
The sound of arms, and the tramp of feet,
And the measured tread of the grenadiers
Marching down to their boats on the shore.

Then he climbed to the tower of the church,
Up the wooden stairs, with stealthy tread,


plaintext →
→ ciphertext


key →
plaintext →
→ ciphertext


1 →
plaintext →
→ ciphertext

I L O V E Y O U

73 L O V E Y O U

73 76 O V E Y O U

73 76 79 V E Y O U

73 76 79 86 E Y O U

73 76 79 86 69 Y O U

73 76 79 86 69 89 0 U

73 76 79 86 69 89 79 U

73   76 79 86 69   89 79 85

74 76 79 86 69 89 79 85

74 77 79 86 69 89 79 85

74 77 80 86 69 89 79 85

```
74  77 80 87 69  89 79 85
```

74 77 80 87 70 89 79 85

74 77 80 87 70 90 79 85

```
74  77 80 87 70  90 80 85
```

74 77 80 87 70 90 80 86

```
J   77 80 87 70  90 80 86
```

```
J   M  80 87 70  90 80 86
```

```
J   M  P  87 70  90 80 86
```

J   M   P   W   70   90 80 86

J   M P W F   90 80 86

J   M P W F   Z 80 86

J M P W F Z P 86

J M P W F Z P V

```
#include <stdio.h>

int main(void)
{
    ...
}
```

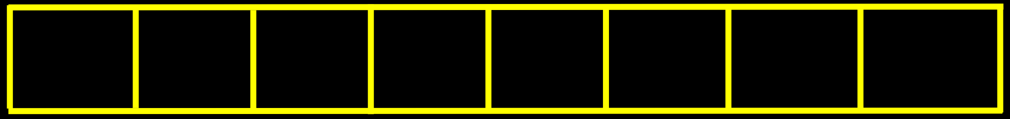
```
#include <stdio.h>

int main(void)
{
    ...
}
```

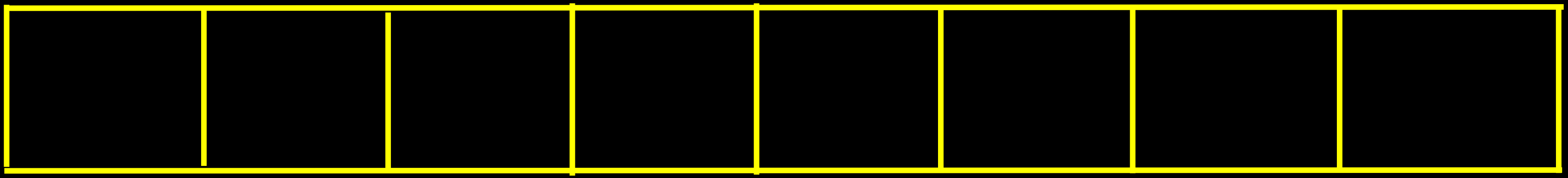
```
#include <stdio.h>

int main(int argc, string argv[])
{
    ...
}
```

```
#include <stdio.h>

int main(int argc, string argv[])
{
    ...
}
```

```
#include <stdio.h>

int main(int argc, string argv[])
{
    ...
}
```

```
#include <stdio.h>

int main(void)
{
    ...
}
```





Bubble Sort

```
repeat until no swaps
    for i from 0 to n-2
        if i'th and i+1'th elements out of order
            swap them
```

Selection Sort

```
for i from 0 to n-1
    find smallest element between i'th and n-1'th
    swap smallest with i'th element
```

$(n - 1)$

$(n - 1) + (n - 2)$

$(n - 1) + (n - 2) + \dots + 1$

$(n - 1) + (n - 2) + ... + 1$

$n(n - 1)/2$

$(n - 1) + (n - 2) + ... + 1$

$n(n - 1)/2$

$(n^2 - \mathrm{n})/2$

$(n - 1) + (n - 2) + ... + 1$

$n(n - 1)/2$

$(n^2 - \mathrm{n})/2$

$n^2/2 - n/2$

$(n - 1) + (n - 2) + ... + 1$

$n(n - 1)/2$

$(n^2 - \mathrm{n})/2$

$n^2/2 - n/2$

$O(n^2)$

$n^2/2 - n/2$

$n^2/2 - n/2$

$1{,}000{,}000^2/2 - 1{,}000{,}000/2$

$n^2/2 - n/2$

$1{,}000{,}000^2/2 - 1{,}000{,}000/2$

$500{,}000{,}000{,}000 - 500{,}000$

$n^2/2 - n/2$

$1{,}000{,}000^2/2 - 1{,}000{,}000/2$

$500{,}000{,}000{,}000 - 500{,}000$

$499{,}999{,}500{,}000$

$n^2/2 - n/2$

$1{,}000{,}000^2/2 - 1{,}000{,}000/2$

$500{,}000{,}000{,}000 - 500{,}000$

$499{,}999{,}500{,}000$

$O(n^2)$

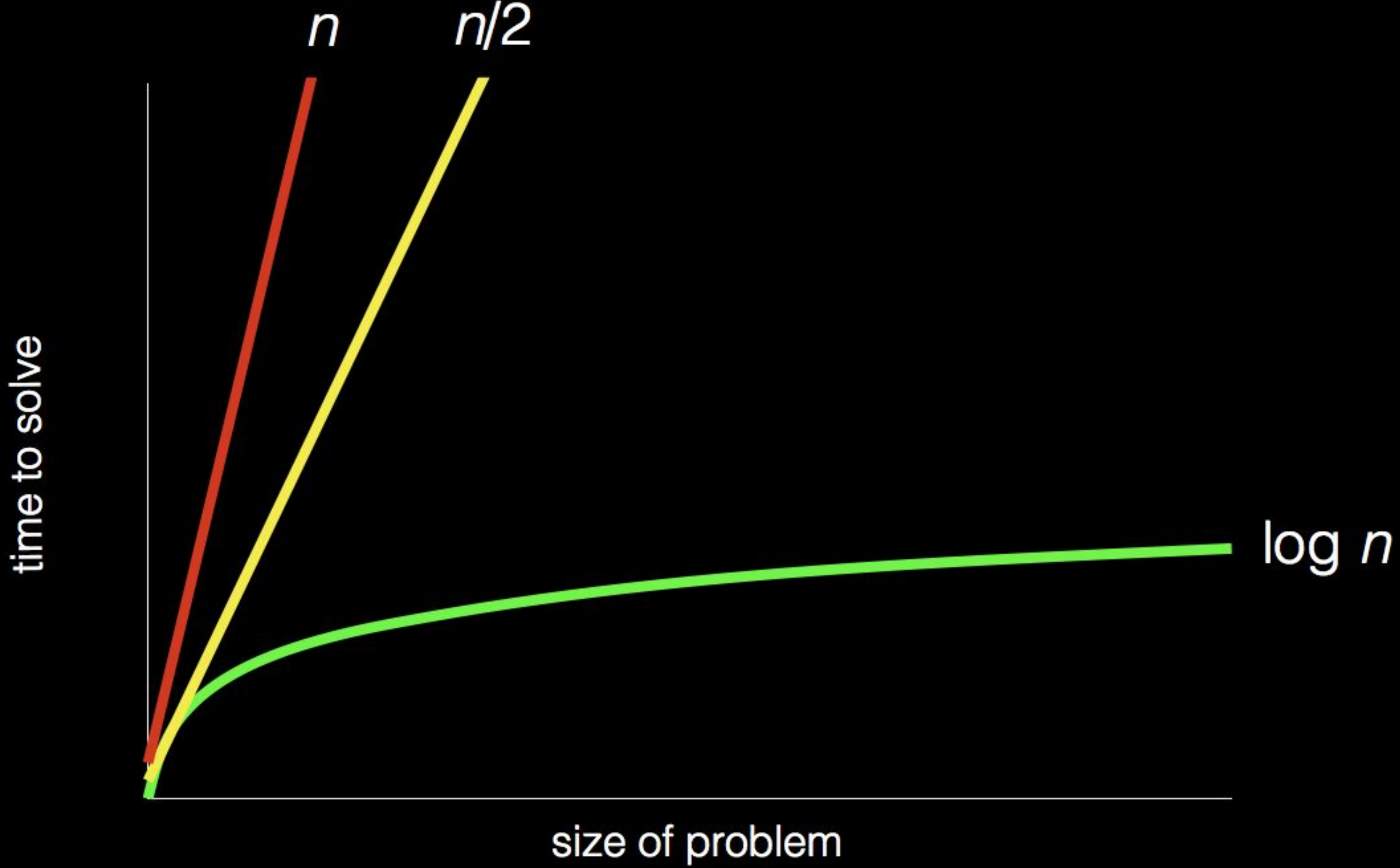
$O(n^2)$

$O(n \log n)$

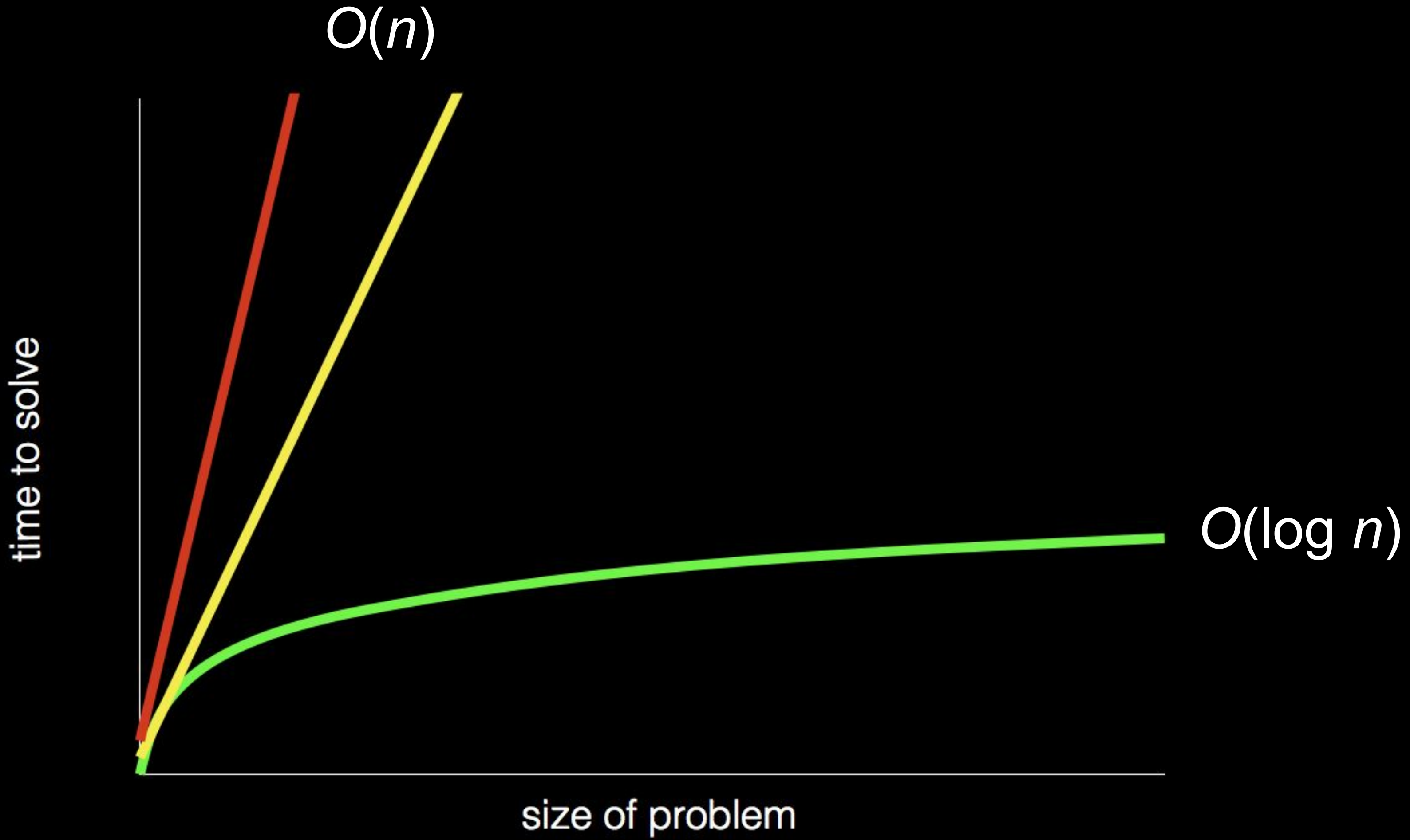
$O(n)$

$O(\log n)$

$O(1)$



n
n/2
log n
time to solve
size of problem


$O(n)$
$O(\log n)$
time to solve
size of problem

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

$\Theta(n^2)$

$\Theta(n \log n)$

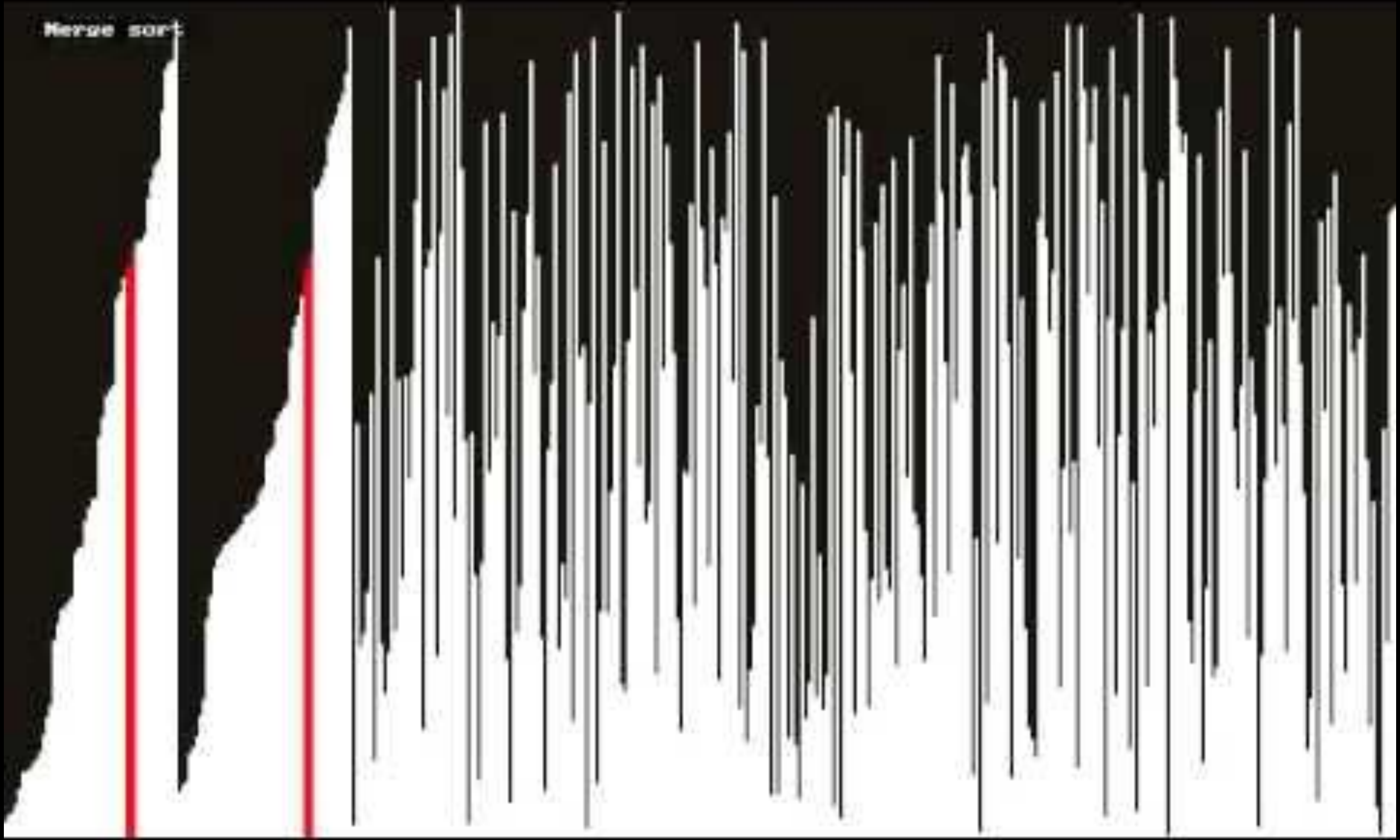
$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$

Merge Sort

```
on input of n elements
    if n < 2
        return
    else
        sort left half of elements
        sort right half of elements
        merge sorted halves
```


Merge sort

This is CS50