Chocolate distribution problem :→ (Sliding Window Problem)

int[] arr = {7, 3, 2, 4, 9, 12, 56}; m = 3 (students)

[0/p: 2]

Step: 1 Sort → [chocolates] 2, 3, 4, 7, 9, 12, 56 (mx, mn)

0 1 2 3 4 5 6

min = Large

for(i=0; i i+m-1 < arr.length ; i++)

(0+3-1) 2
1+3-1 3
2+3-1 4

int diff = a(i+m-1) - a(i)

min = min(diff, min)

	Mx	Mn
0	4	2
	7	3
	9	4
	12	7
	56	9

= 2
= 4
= 5
= 5
= 47

ClassName obj = new Classname(); — P S V M

ClassName: class; new: instance/object/reference ⇒ Heap Memory; Classname(): class, Constructor

{Static Memory} — var

Heap

Therefore, the main method is static so that when the Java Application runs, we don't need an object to call/run the "main method".

The more no of objects the less efficient is the code.

Dangling pointer

int a = 10 → value

1000 addr → &a l-p

folder

Storage Classes

name	scope	lifetime
static	global	file
extern	global	folder
*register	RAM	CPU
auto	Local	{ }

int * ptr = &a;

1000

* No pointers allowed !

C + C++ = Java

C + SIMULA = C++

Rust

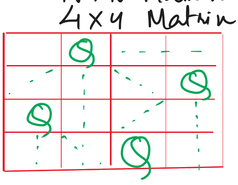
Application of Recursion :→ (N Queens)*(Backtracking)

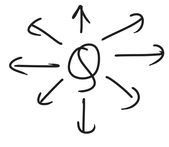
(Rat In A Maze)

N×N Matrix N=4

4×4 Matrix

(Chessboard)





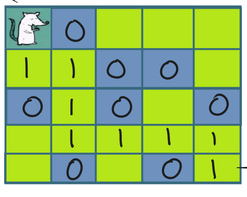
Rat In A Maze :→ (n×n)

green = safe (1)

Dark = walls (0)

Two directions:

(0,0)

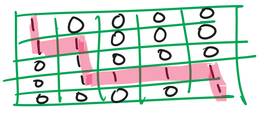


(n-1, n-1)

choose → Currpos (x,y){

Forward (x+1, y)

Downward (x, y+1)



all zeros.

}