International study centre

Searching, Sorting, Test Driven Development

- ❑Name : John Alamina
- ❑Email : john.alamina@hud.ac.uk

Contents

❑ Searching Algorithm

❑ Simple Sort

❑ Selection Sort

❑ Bubble sort

❑ Test Driven development

Searching algorithm

- For an array it is simply a matter of using a for loop to access each member until a match is found. Then return the index of the matching number or -1 if no match was found.

Searching algorithm

```
int find(int needle, int num[], int len) {
      for (n = 0; n < len; n++)
      {
            if(num[n]==needle)return n;
      }
      return -1;
}
```

Sorting algorithm

- Sorting is a little more subtle. We study the following
  a. Simple sorting strategy
  b. Insertion sort algorithm
  c. Bubble sort algorithm

Simple sort strategy

- Given an array of 5 elements let’s visually walk through a simple sort strategy.
- Whenever we visually walk through a strategy to solve a problem it is called a “dry run”.

Simple sort strategy code

```
void ssort(int arr[], int n)
{
        int sorted_arr[n];
        int idx=findMin(arr,n);
        int sai=0;
        sorted_arr[sai++]=arr[idx];
        int tmp_array[n];
        for (int i = n; i > 0; --i) {
                int a=0;
                for (int j = 0; j < i; j++) {
                        if (j == idx)continue;
                        tmp_array[a++]=arr[j];
                }
                for(int k=0;k<i-1;k++)arr[k]=tmp_array[k];
                idx = findMin(arr,i-1);
                sorted_arr[sai++] = arr[idx];
        }
        printArray(sorted_arr,n);
}
```

Simple sort strategy

- Given the array below

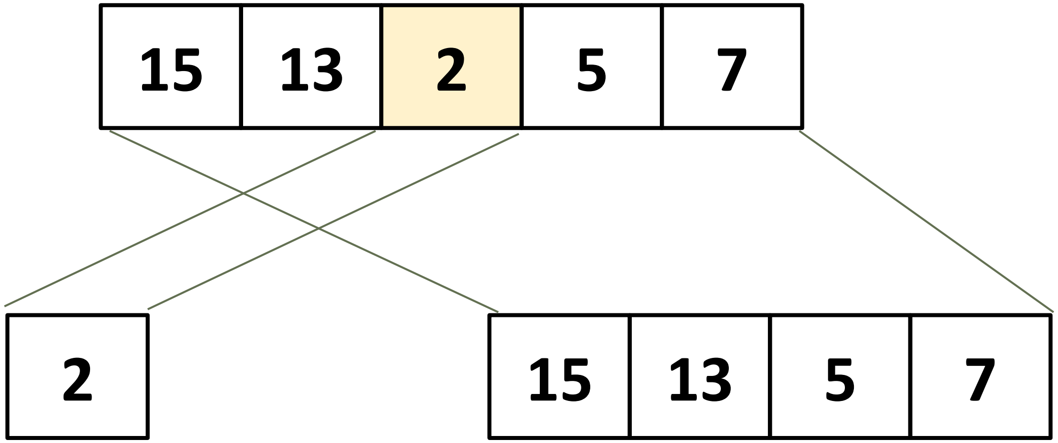
15	**13**	**2**	**5**	**7**

- Step 1 (find the minimum)

15	**13**	**2**	**5**	**7**

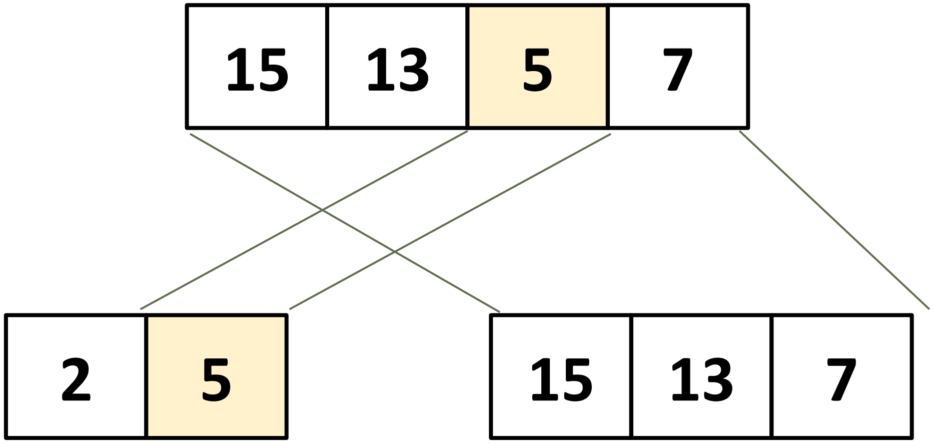
Simple sort strategy

- Create two more arrays like so



Simple sort strategy

- Repeat the process again



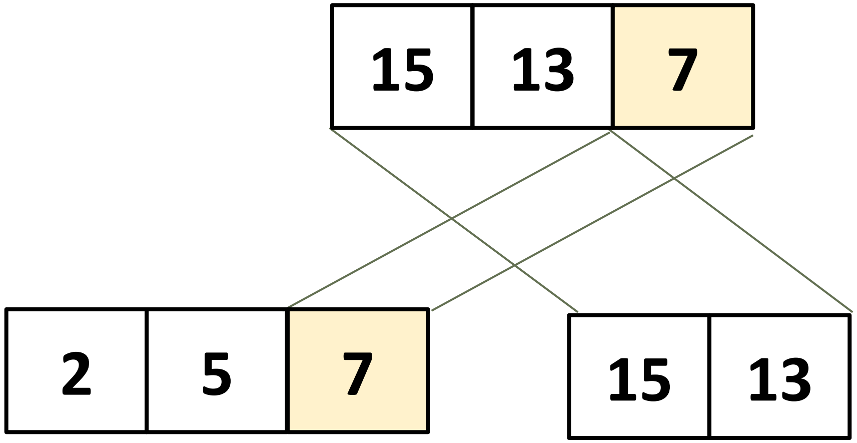
15 13 5 7

2 5

15 13 7

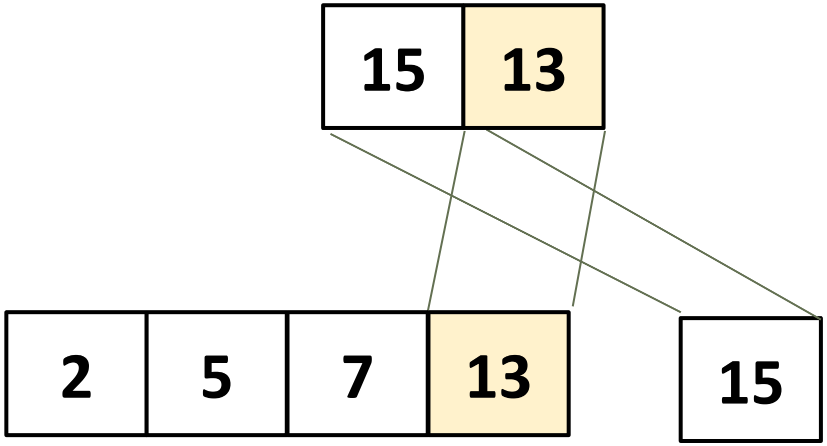
Simple sort strategy

- and again



Simple sort strategy

- Until finally



Simple sort strategy

- There’s a single sorted array

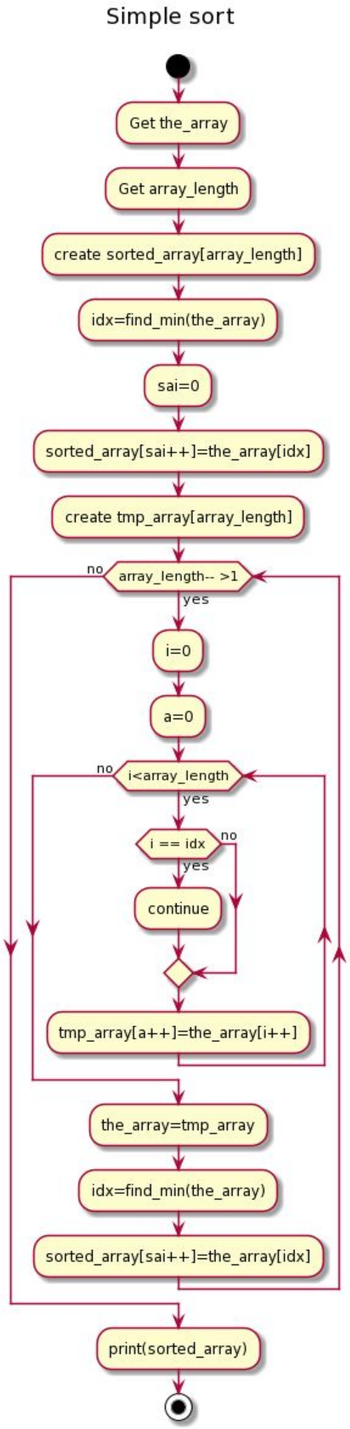
2	**5**	**7**	**13**	**15**

Simple sort strategy - dry run

- That’s only half the story
- A dry run is not usually as visually appealing
- First you need an algorithm/flowchart
- Then you need a table of variables
- Then you need to run through the flowchart
- And make changes to the values accordingly

Here’s the algorithm

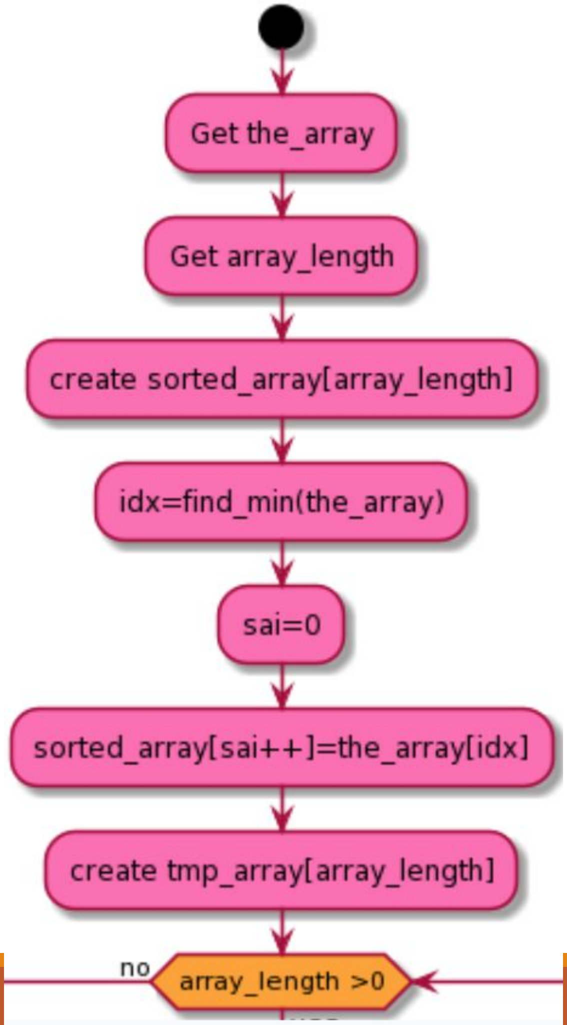
```
start
:Get the_array;
:Get array_length;
:create sorted_array[array_length];
:idx=find_min(the_array);
:sai=0;
:sorted_array[sai++]=the_array[idx];
:create tmp_array[array_length];
while (array_length>0) is (yes)
   :i=0;
        :a=0;
    while(i<array_length) is (yes)
    if(i == idx) is (yes) then
      :continue;
    else (no)
    endif
    :tmp_array[a++]=the_array[i++];
    end while (no)
    :array_length--;
    :the_array=tmp_array;
    :idx=find_min(the_array);
    :sorted_array[sai++]=the_array[idx];
end while (no)
:print(sorted_array);
stop
```



Here’s table of variables and conditional expressions

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	-
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	0→1
i<arr_length	

Variable	Value
tmp_array[0]	
tmp_array[1]	
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	5
idx	4
find_min()	4
arr_length >0	
i++	
i++ == idx	
a++	



Get the_array

Get array_length

create sorted_array[array_length]

idx=find_min(the_array)

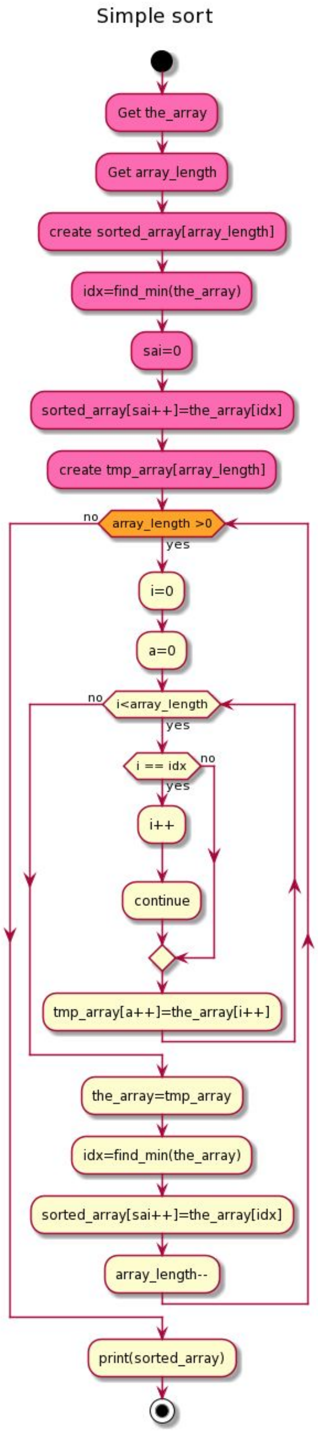
sai=0

sorted_array[sai++]=the_array[idx]

create tmp_array[array_length]

no

array_length >0



Simple sort

Get the_array

Get array_length

create sorted_array[array_length]

idx=find_min(the_array)

sai=0

sorted_array[sai++]=the_array[idx]

create tmp_array[array_length]

no

array_length >0

yes

i=0

a=0

no

i<array_length

yes

i == idx

no

yes

i++

continue

tmp_array[a++]=the_array[i++]

the_array=tmp_array

idx=find_min(the_array)

sorted_array[sai++]=the_array[idx]

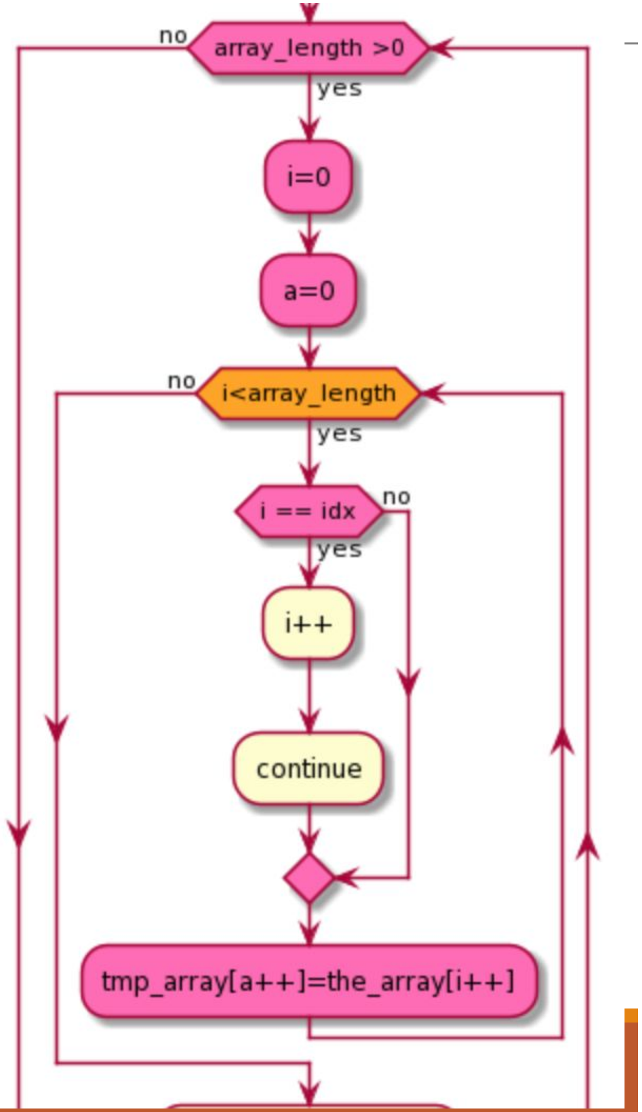
array_length--

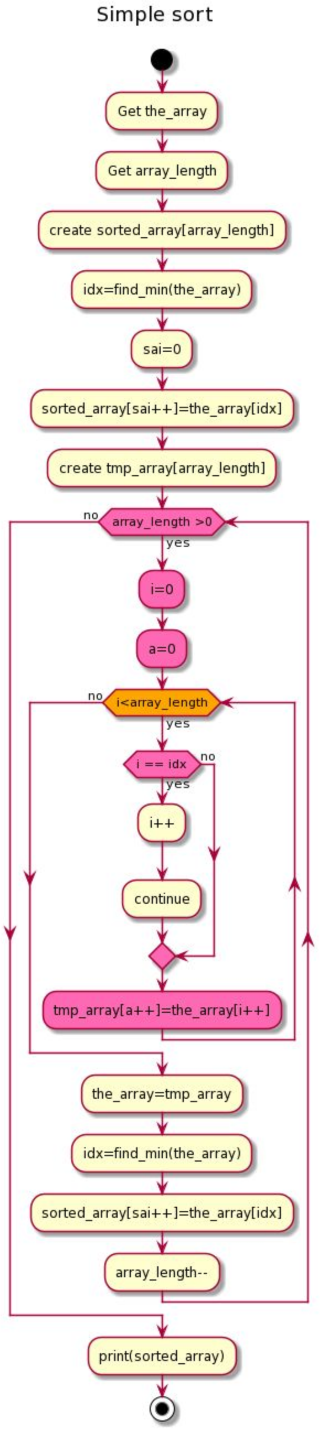
print(sorted_array)

Copying(5) into tmp_array..0

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	-
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	0→1
i<arr_length	(0<5) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	5
idx	4
find_min()	4
arr_length >0	(5>0) Y
i++	0 → 1
i == idx	(0==4) N
a++	0 → 1

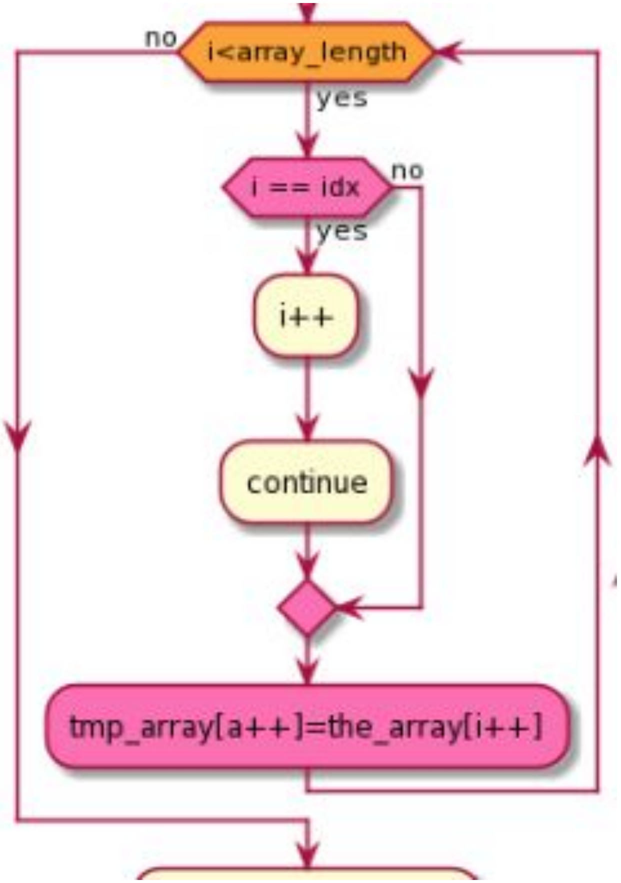




Copying(5) into tmp_array..1

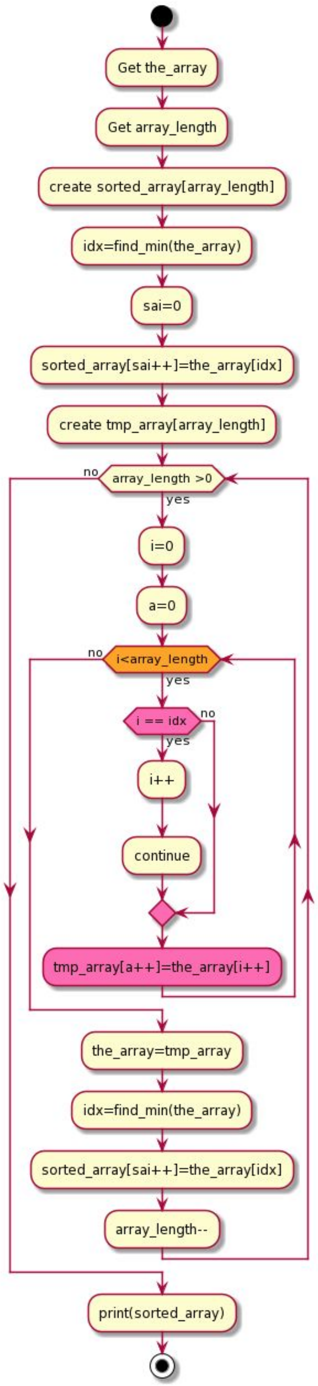
Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	-
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	0→1
i<arr_length	(1<5) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	6
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	5
idx	4
find_min()	4
arr_length >0	(5>0) Y
i++	1 →2
i == idx	(1==4) N
a++	1 → 2



no
i<array_length
yes
i == idx
no
yes
i++
continue
tmp_array[a++]=the_array[i++]

Simple sort

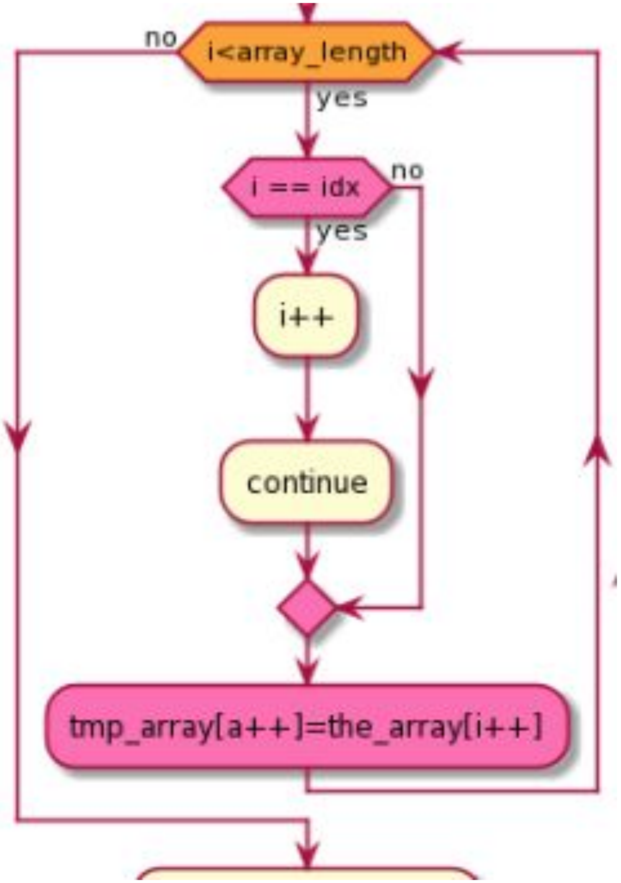


Get the_array
Get array_length
create sorted_array[array_length]
idx=find_min(the_array)
sai=0
sorted_array[sai++]=the_array[idx]
create tmp_array[array_length]
no
array_length >0
yes
i=0
a=0
no
i<array_length
yes
i == idx
no
yes
i++
continue
tmp_array[a++]=the_array[i++]
the_array=tmp_array
idx=find_min(the_array)
sorted_array[sai++]=the_array[idx]
array_length--
print(sorted_array)

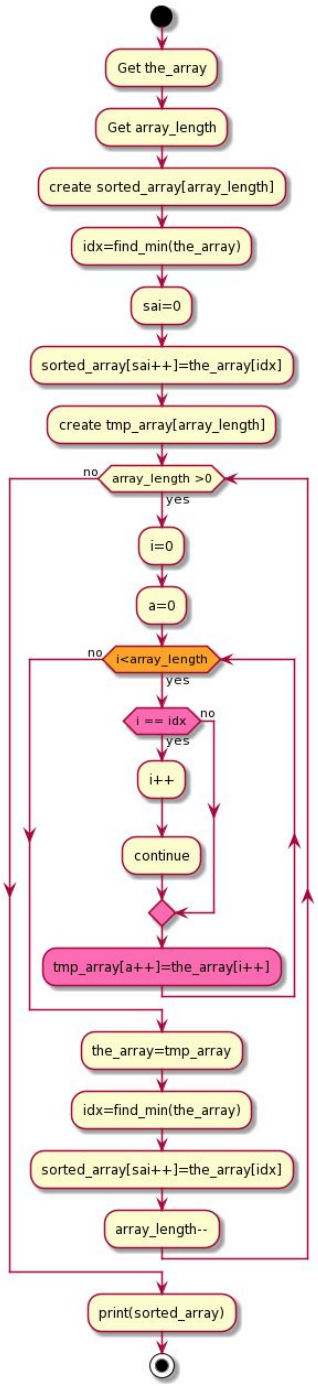
Copying(5) into tmp array..2

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	-
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	0→1
i<arr_length	(2<5) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	6
tmp_array[2]	10
tmp_array[3]	
tmp_array[4]	
arr_length	5
idx	4
find_min()	4
arr_length >0	(5 > 0)Y
i++	2 →3
i == idx	(2==4) N
a++	2→ 3



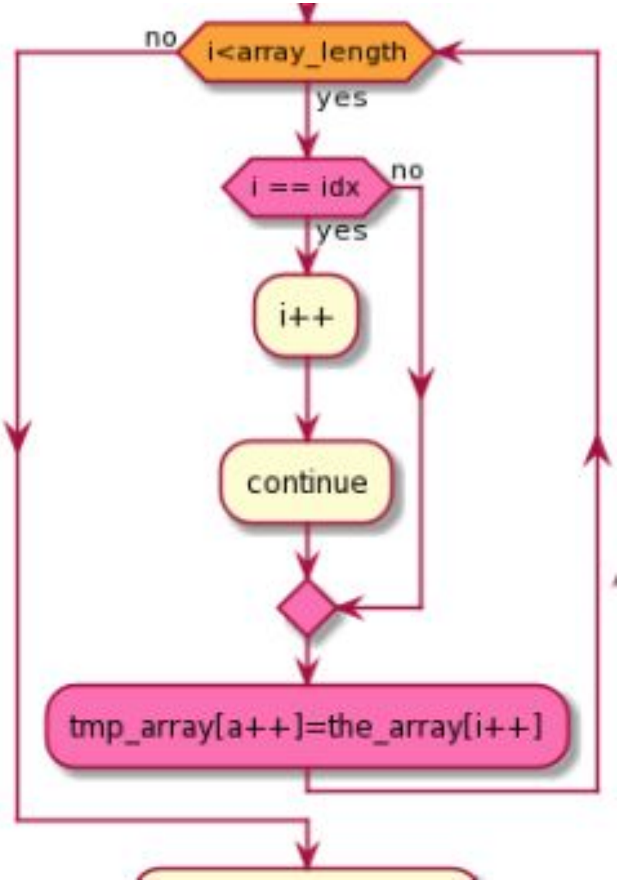
Simple sort

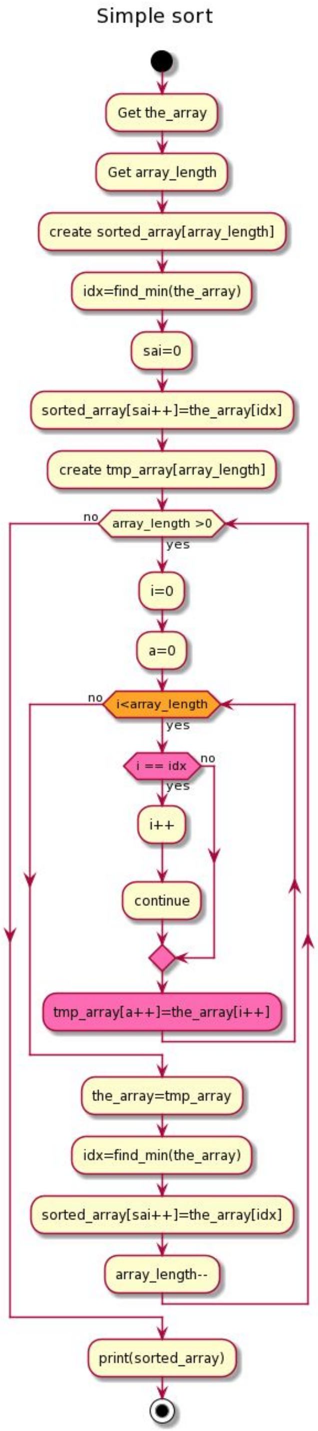


Copying(5) into tmp_array..3

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	-
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	0→1
i<arr_length	(3<5) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	6
tmp_array[2]	10
tmp_array[3]	5
tmp_array[4]	
arr_length	5
idx	4
find_min()	4
arr_length >0	(5 > 0) Y
i++	3 → 4
i == idx	(3==4) N
a++	3 → 4

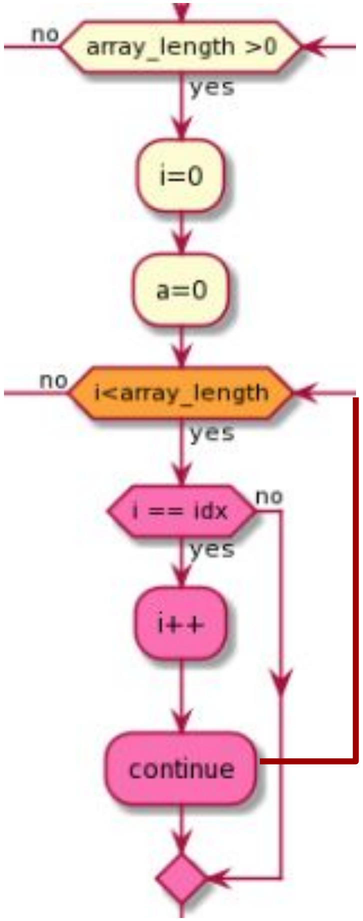




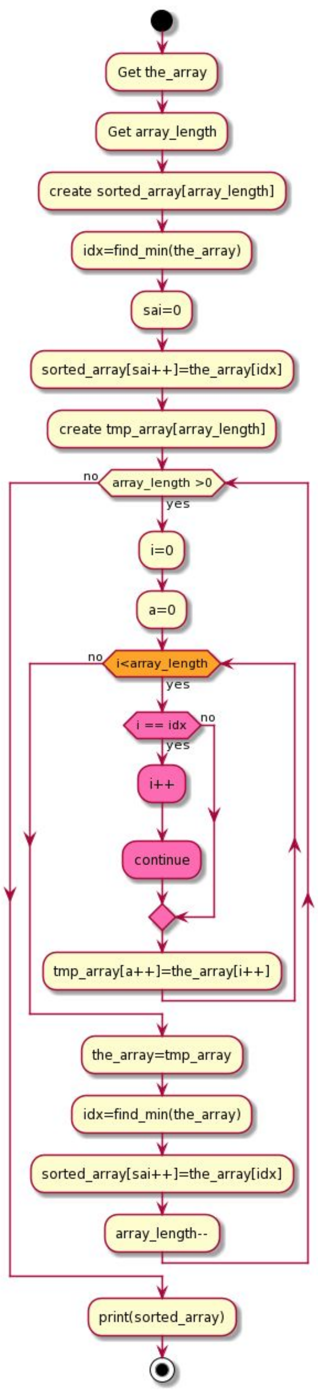
Copying(5) into tmp_array..4

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	-
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	0→1
i<arr_length	(4<5) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	6
tmp_array[2]	10
tmp_array[3]	5
tmp_array[4]	
arr_length	5
idx	4
find_min()	4
arr_length >0	(5 > 0) Y
i++	4 → 5
i == idx	(4==4) Y
a++	4



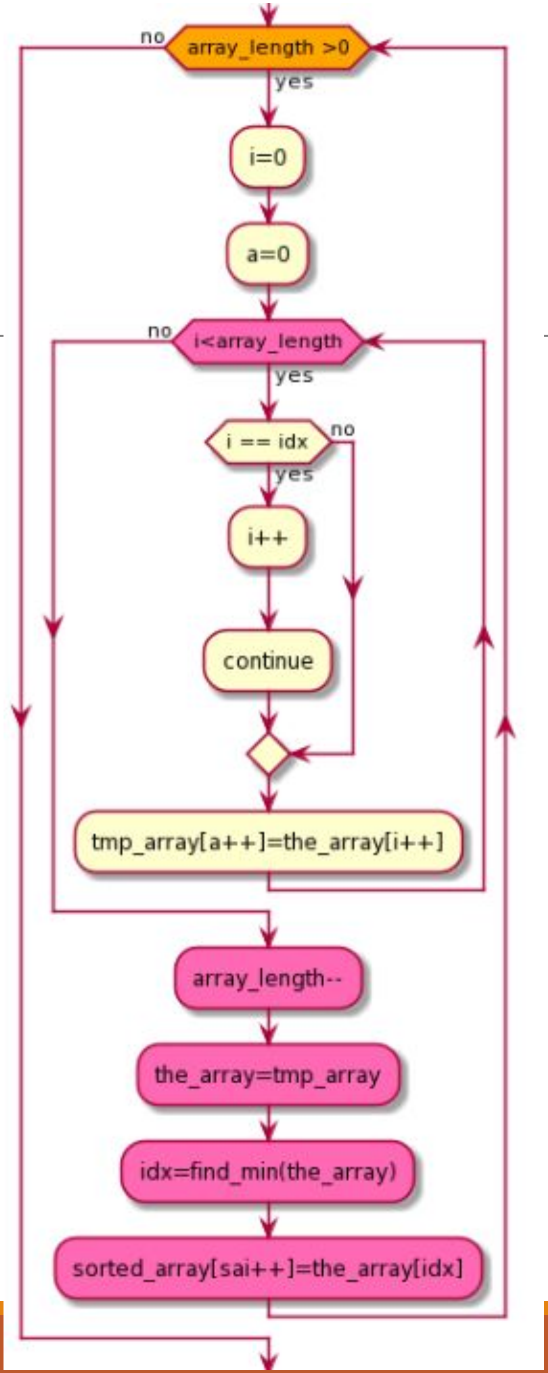
Simple sort



Update outer-loop

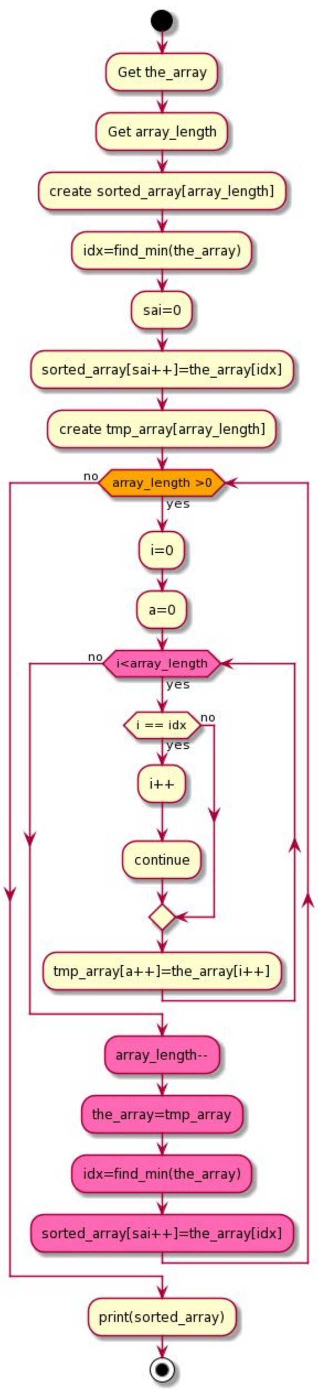
Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	-
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	1→2
i<arr_length	(5<5) N

Variable	Value
tmp_array[0]	12
tmp_array[1]	6
tmp_array[2]	10
tmp_array[3]	5
tmp_array[4]	-
arr_length--	5 → 4
idx	3
find_min()	3
arr_length >0	(5 > 0) Y
i	5
i == idx	(4==4) Y
a++	3 → 4



no
array_length >0
yes
i=0
a=0
no
i<array_length
yes
i == idx
no
yes
i++
continue
tmp_array[a++]=the_array[i++]
array_length--
the_array=tmp_array
idx=find_min(the_array)
sorted_array[sai++]=the_array[idx]

Simple sort

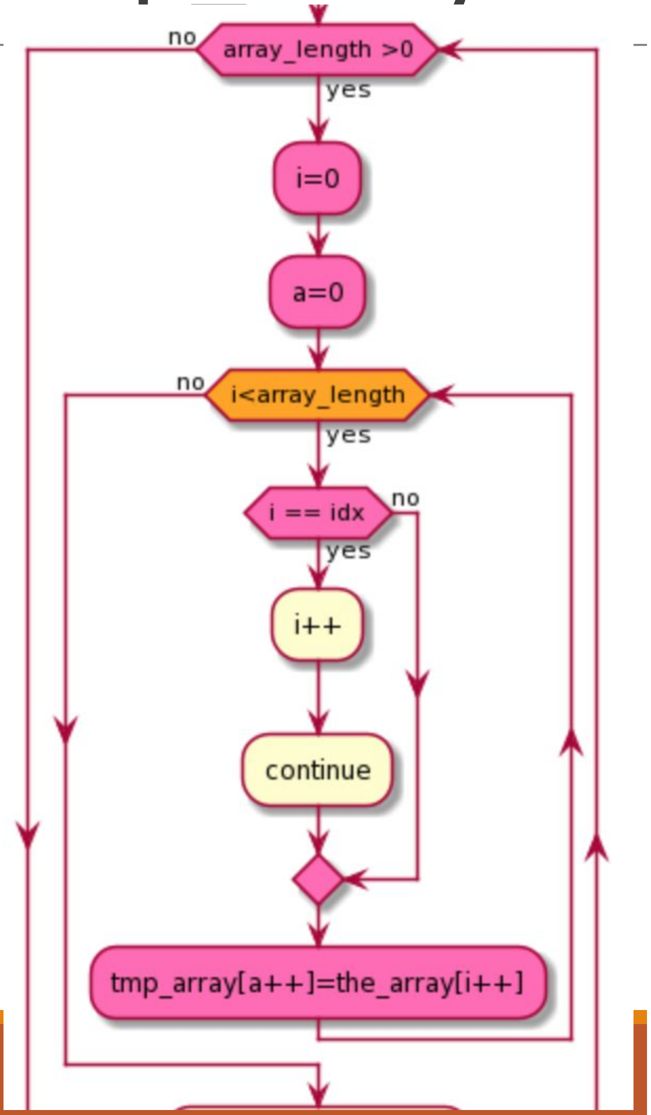


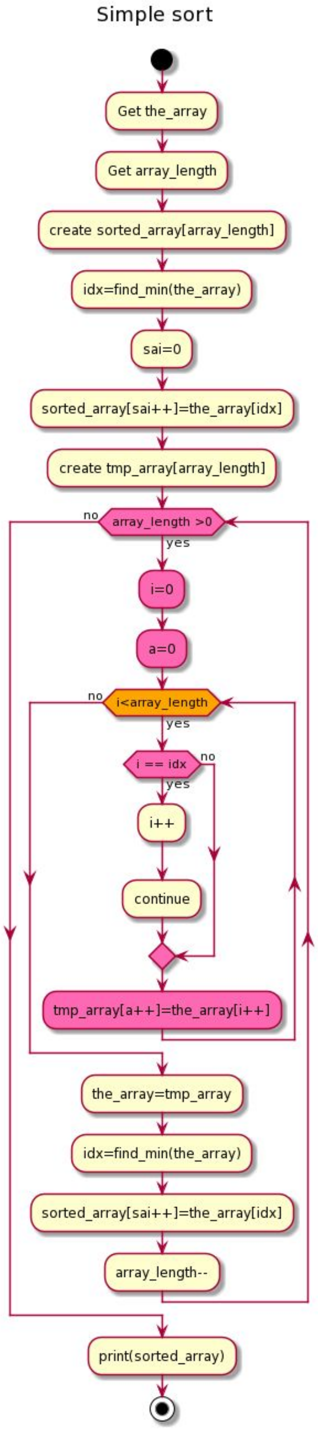
Get the_array
Get array_length
create sorted_array[array_length]
idx=find_min(the_array)
sai=0
sorted_array[sai++]=the_array[idx]
create tmp_array[array_length]
no
array_length >0
yes
i=0
a=0
no
i<array_length
yes
i == idx
no
yes
i++
continue
tmp_array[a++]=the_array[i++]
array_length--
the_array=tmp_array
idx=find_min(the_array)
sorted_array[sai++]=the_array[idx]
print(sorted_array)

Copying(4) into tmp_array..0

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	1→2
i<arr_length	(0<4) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	4
idx	3
find_min()	3
arr_length >0	(4>0) Y
i++	0 → 1
i == idx	(0==3) N
a++	0 → 1

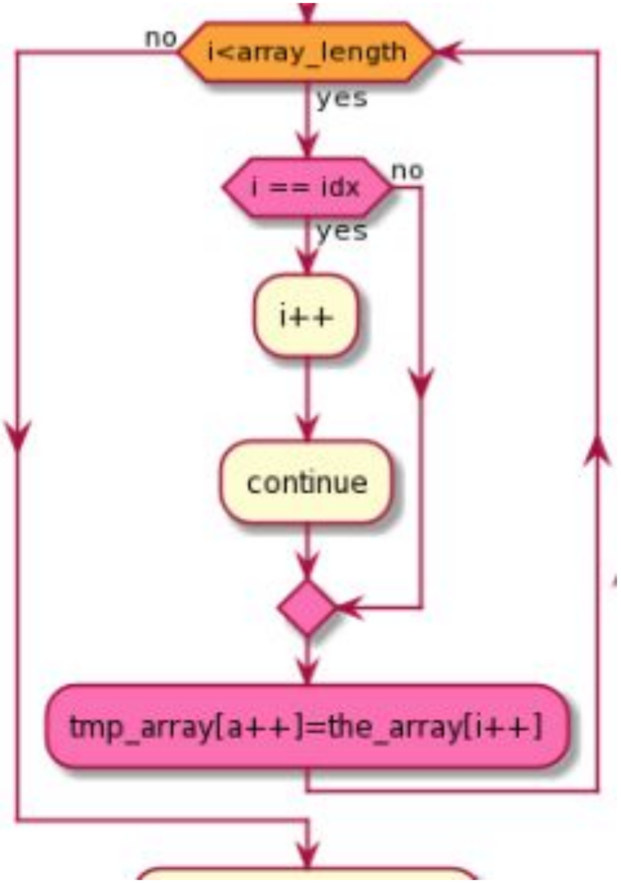




Copying(4) into tmp_array..1

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	1→2
i<arr_length	(1<4) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	6
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	4
idx	3
find_min()	3
arr_length >0	(4>0) Y
i++	1 →2
i == idx	(1==3) N
a++	1 → 2



no

i<array_length

yes

i == idx

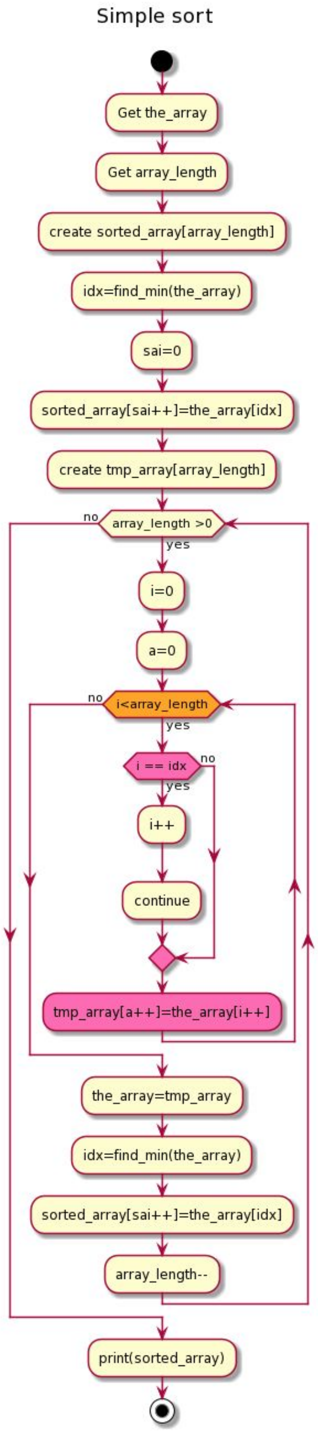
no

yes

i++

continue

tmp_array[a++]=the_array[i++]



Simple sort

Get the_array

Get array_length

create sorted_array[array_length]

idx=find_min(the_array)

sai=0

sorted_array[sai++]=the_array[idx]

create tmp_array[array_length]

no

array_length >0

yes

i=0

a=0

no

i<array_length

yes

i == idx

no

yes

i++

continue

tmp_array[a++]=the_array[i++]

the_array=tmp_array

idx=find_min(the_array)

sorted_array[sai++]=the_array[idx]

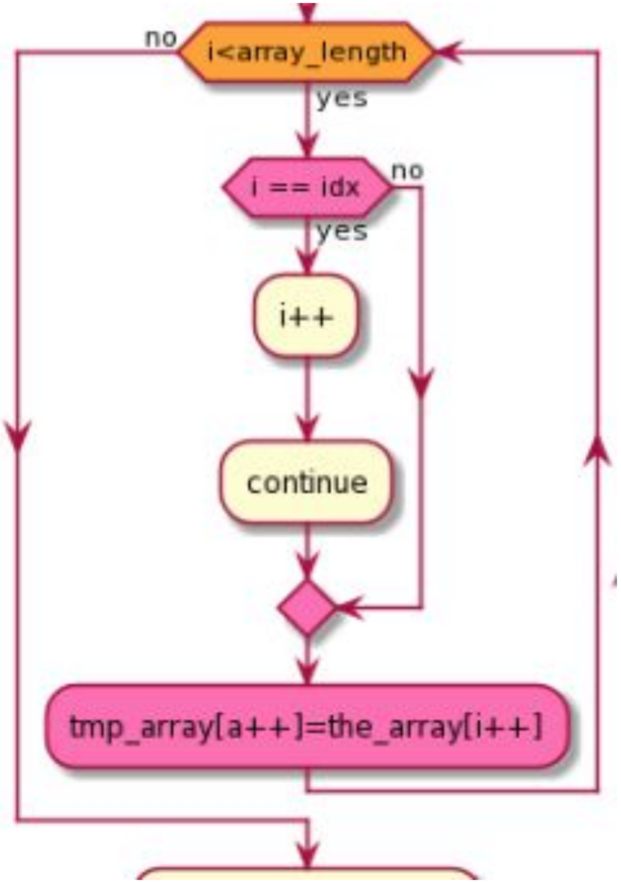
array_length--

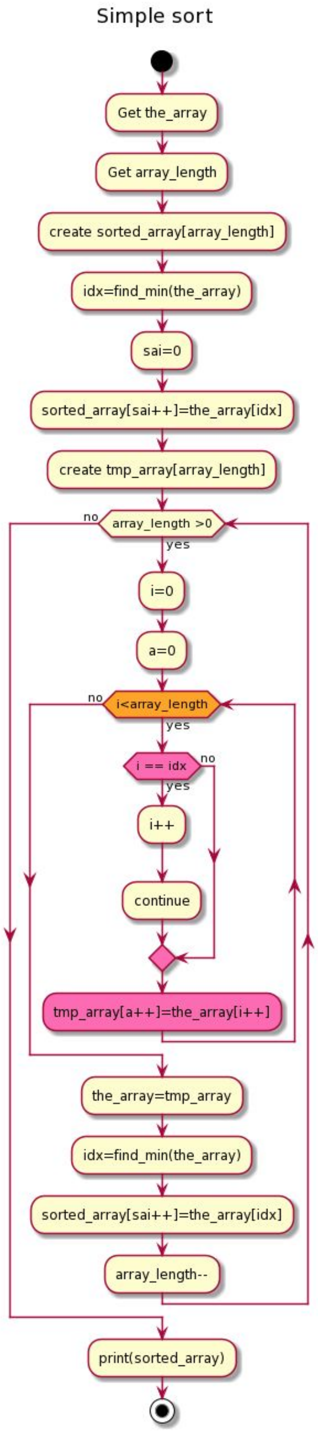
print(sorted_array)

Copying(4) into tmp array..2

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	1→2
i<arr_length	(2<4) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	6
tmp_array[2]	10
tmp_array[3]	
tmp_array[4]	
arr_length	4
idx	3
find_min()	3
arr_length >0	(4> 0) Y
i++	2 →3
i == idx	(2==3) N
a++	2→ 3

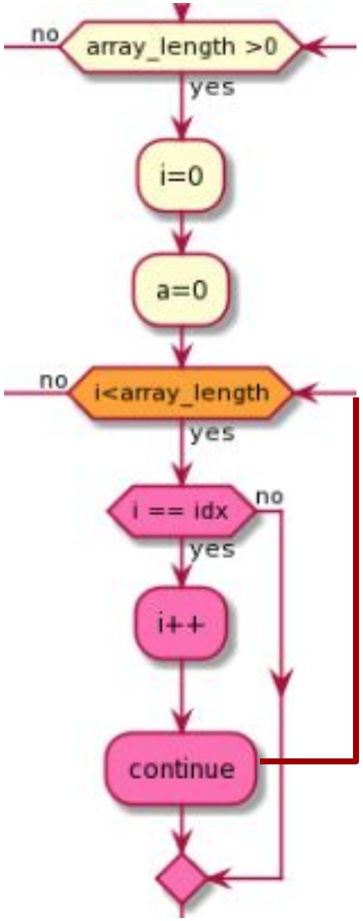


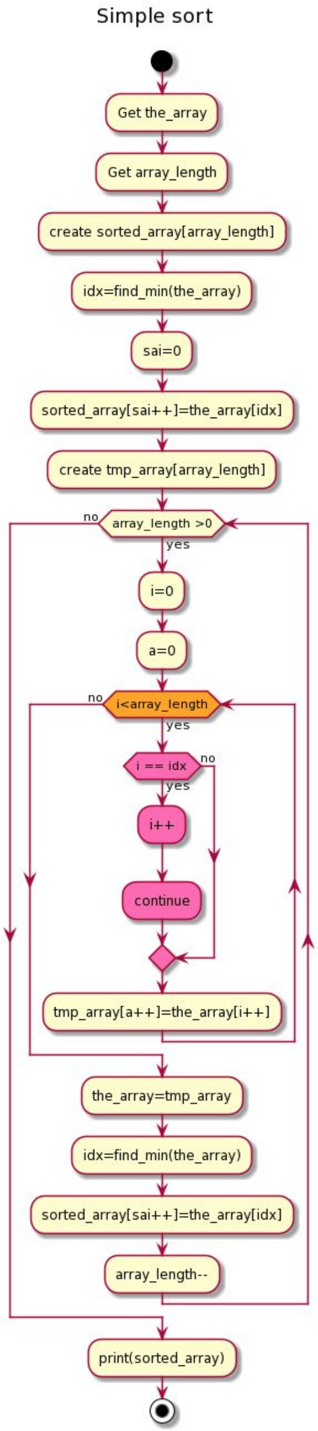


Copying(4) into tmp_array..3

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	-
sorted_arr[3]	-
sorted_arr[4]	-
sai++	1→2
i<arr_length	(3<4) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	6
tmp_array[2]	10
tmp_array[3]	5
tmp_array[4]	
arr_length	4
idx	3
find_min()	3
arr_length >0	(4 > 0) Y
i++	3 → 4
i == idx	(3==3) Y
a++	3

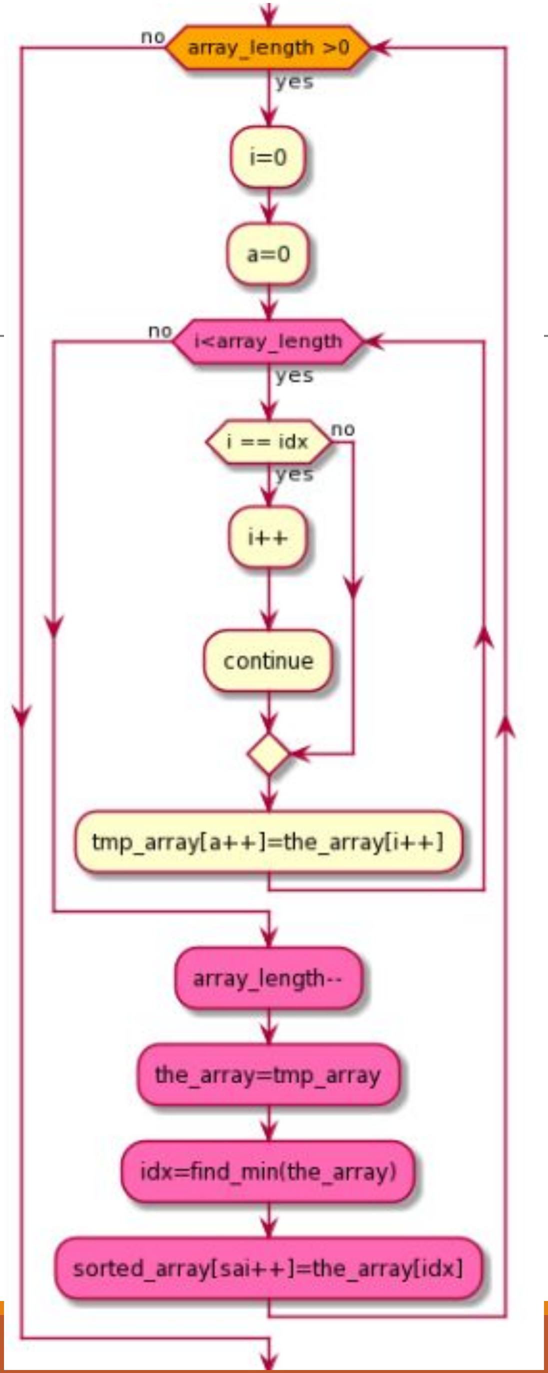


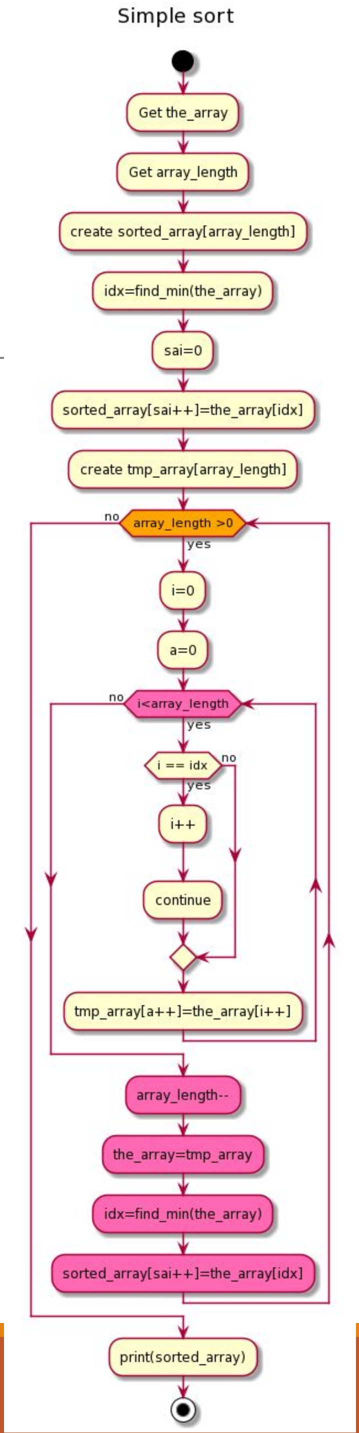


Update outer-loop

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	-
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	-
sorted_arr[4]	-
sai++	2→3
i<arr_length	(4<4) N

Variable	Value
tmp_array[0]	12
tmp_array[1]	6
tmp_array[2]	10
tmp_array[3]	5
tmp_array[4]	-
arr_length--	4 → 3
idx	1
find_min()	1
arr_length >0	(4 > 0) Y
i	4
i == idx	(4==4) Y
a++	3 → 4

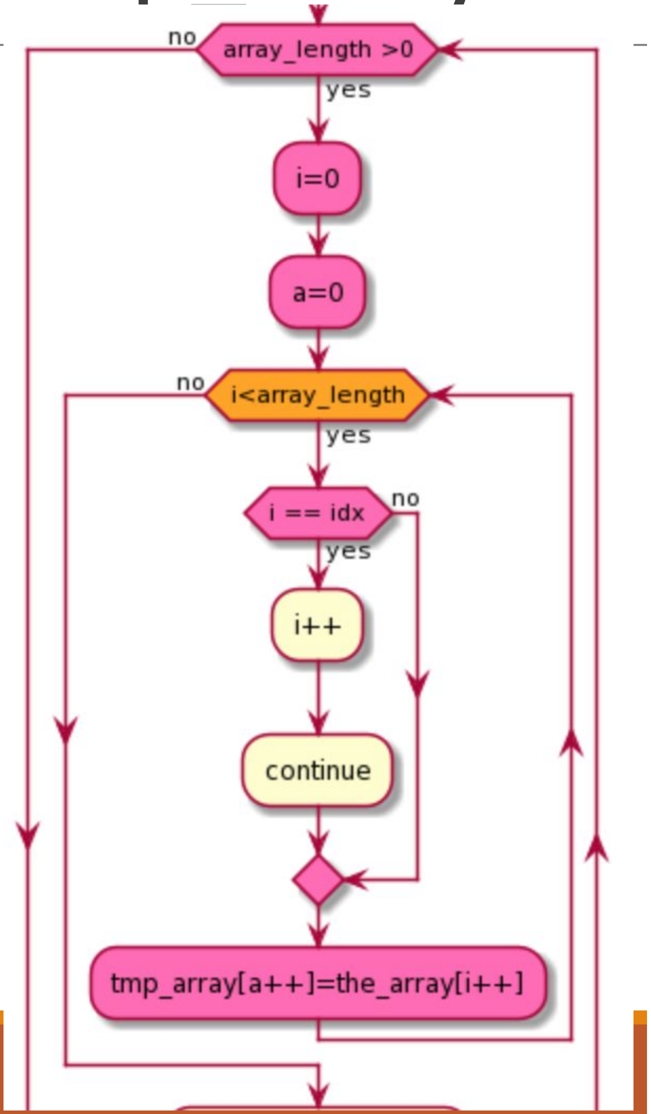


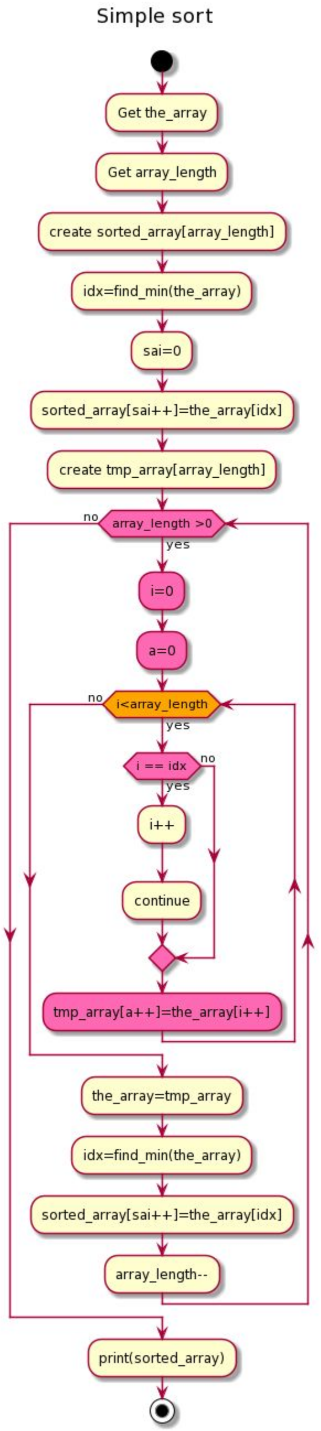


Copying(3) into tmp_array..0

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	-
sorted_arr[4]	-
sai++	2→3
i<arr_length	(0<3) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	3
idx	1
find_min()	1
arr_length >0	(3>0) Y
i++	0 → 1
i == idx	(0==1) N
a++	0 → 1

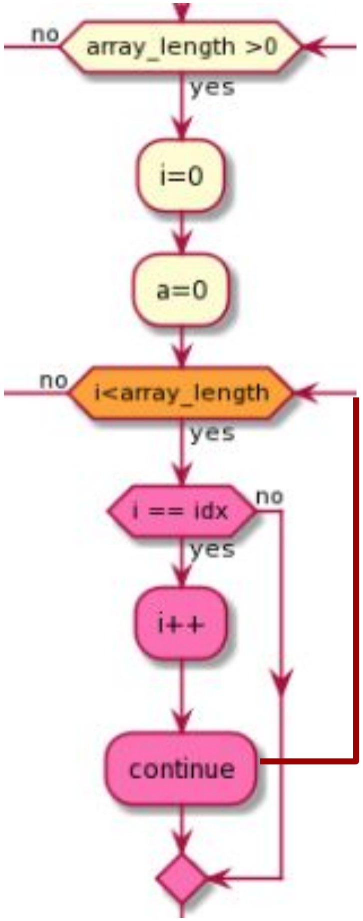




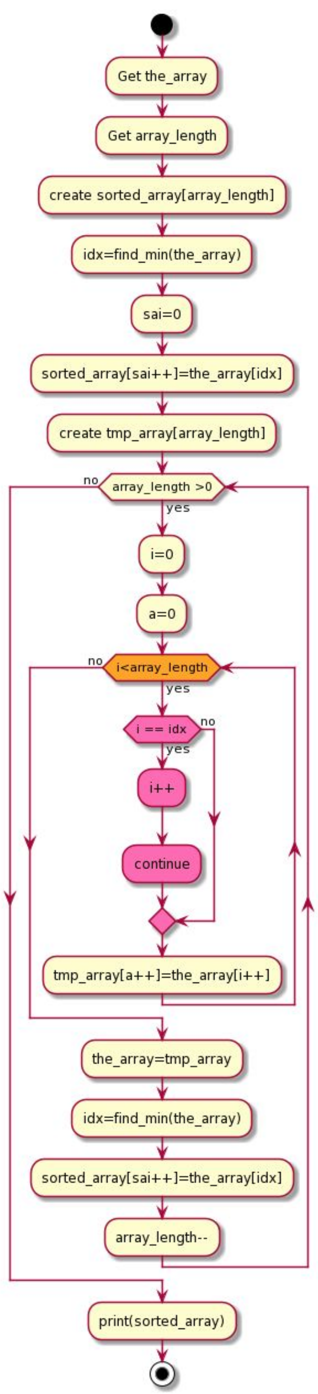
Copying(3) into tmp_array..1

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	-
sorted_arr[4]	-
sai++	2→3
i<arr_length	(1<3) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	3
idx	1
find_min()	1
arr_length >0	(3 > 0) Y
i++	1 → 2
i == idx	(1==1) Y
a++	0 → 1



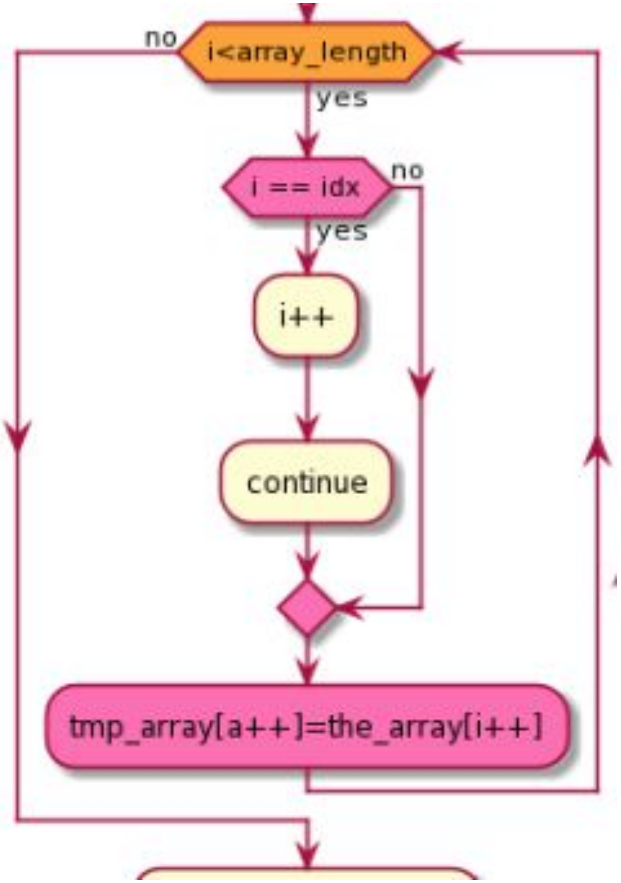
Simple sort

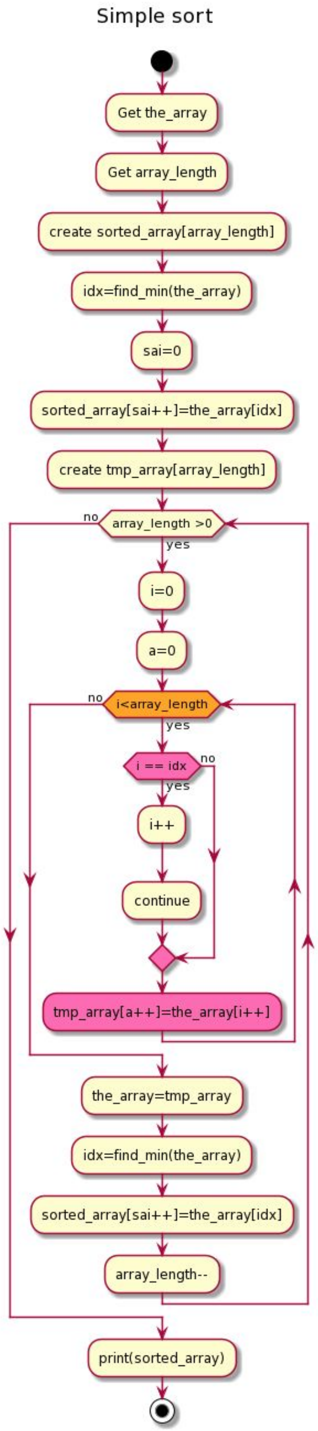


Copying(3) into tmp array..2

Variable	Value
the_array[0]	12
the_array[1]	6
the_array[2]	10
the_array[3]	5
the_array[4]	2
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	-
sorted_arr[4]	-
sai++	2→3
i<arr_length	(2<3) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	10
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	3
idx	1
find_min()	1
arr_length >0	(3> 0) Y
i++	2 →3
i == idx	(2==1) N
a++	1→ 2

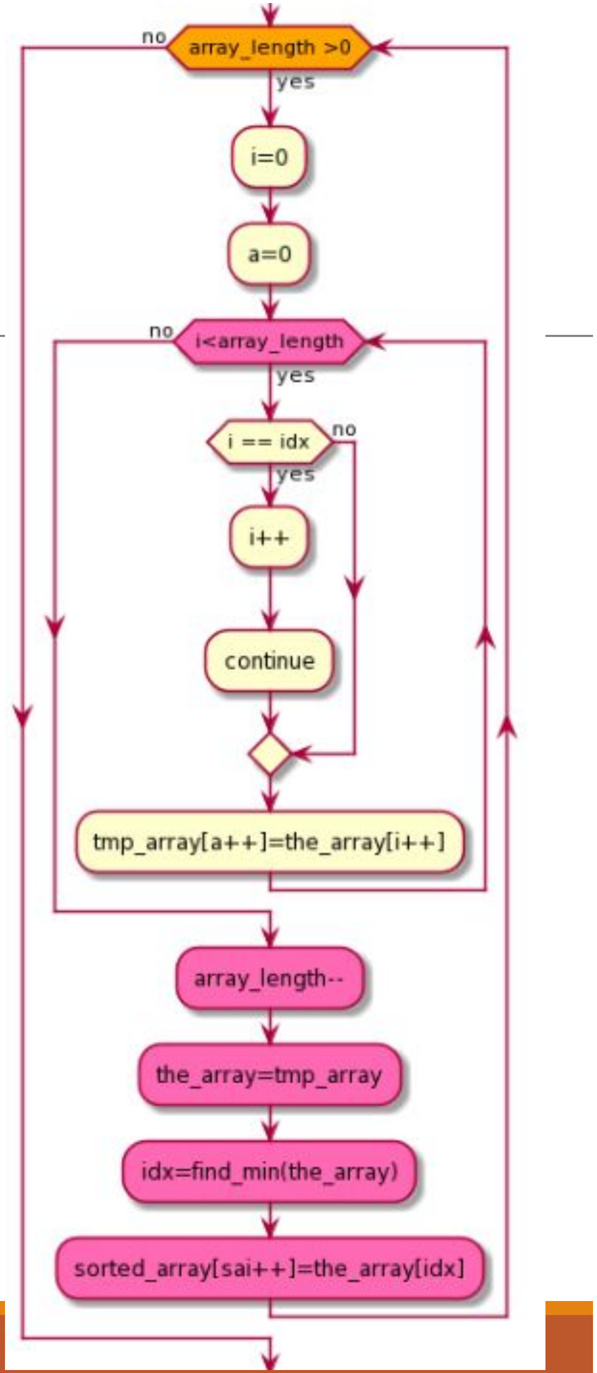




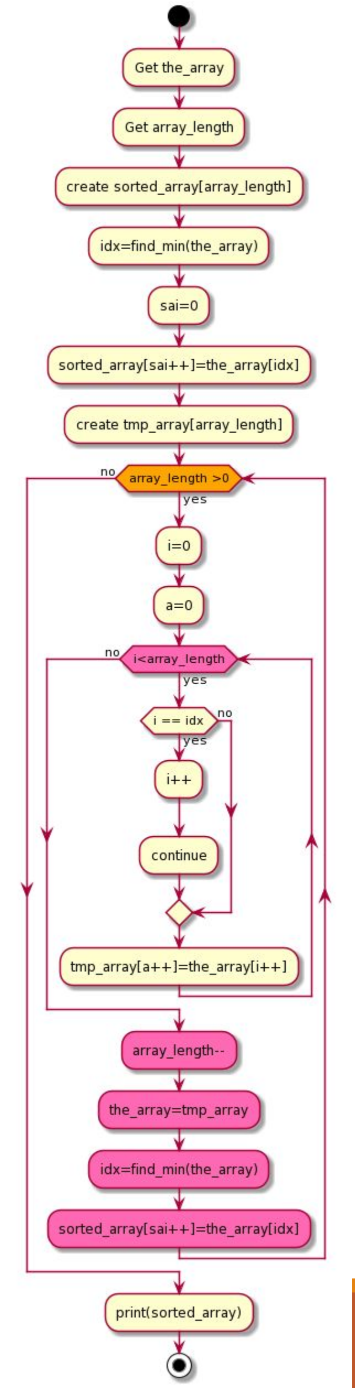
Update outer-loop

Variable	Value
the_array[0]	12
the_array[1]	10
the_array[2]	-
the_array[3]	-
the_array[4]	-
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	10
sorted_arr[4]	-
sai++	3→4
i<arr_length	(3<3) N

Variable	Value
tmp_array[0]	12
tmp_array[1]	10
tmp_array[2]	-
tmp_array[3]	-
tmp_array[4]	-
arr_length--	3 → 2
idx	1
find_min()	1
arr_length >0	(3 > 0) Y
i	3
i == idx	(1==1) Y
a++	1 → 2



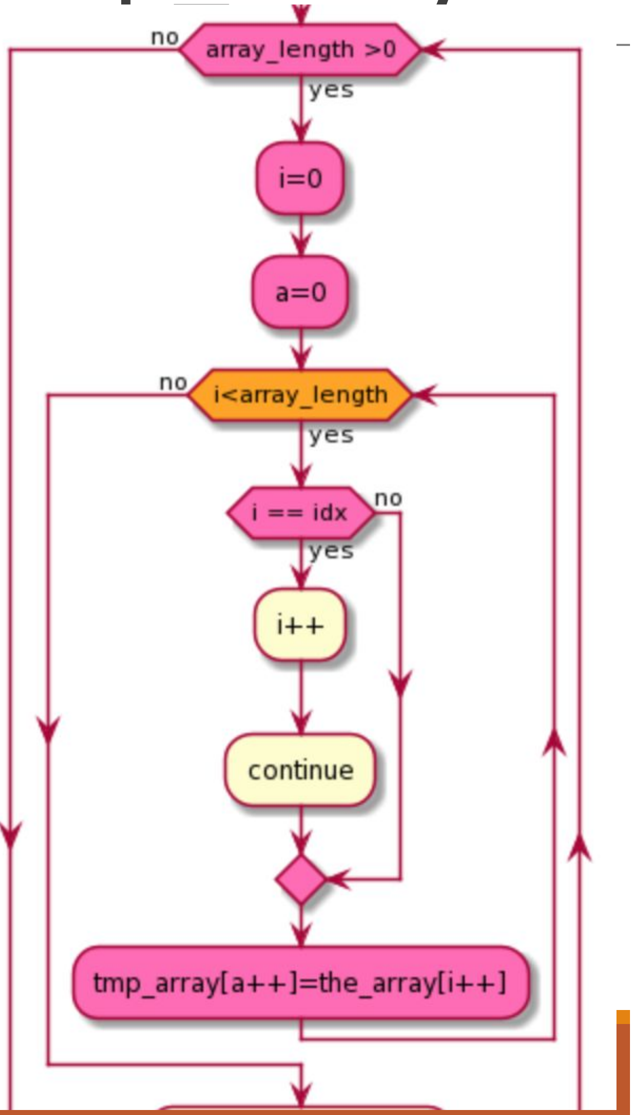
Simple sort



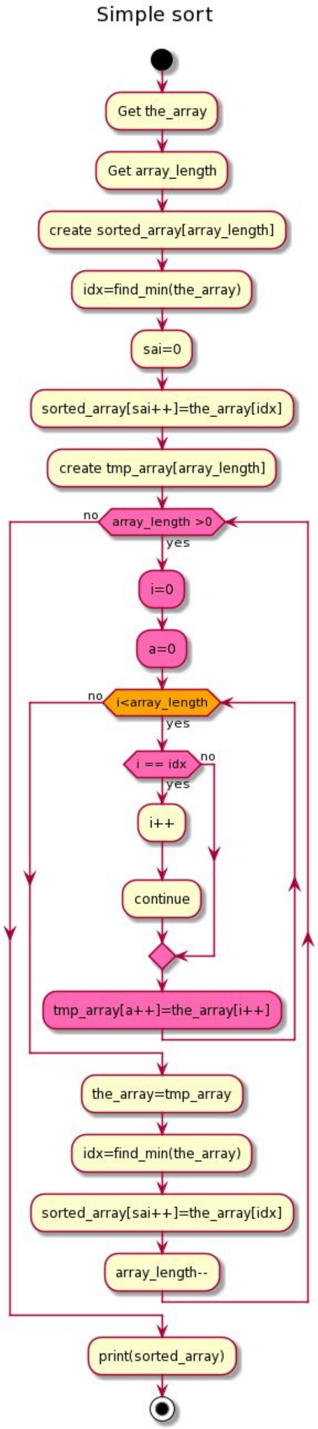
Copying(2) into tmp_array..0

Variable	Value
the_array[0]	12
the_array[1]	10
the_array[2]	-
the_array[3]	-
the_array[4]	-
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	10
sorted_arr[4]	-
sai++	3→4
i<arr_length	(0<2) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	2
idx	1
find_min()	1
arr_length >0	(2>0) Y
i++	0 → 1
i == idx	(0==1) N
a++	0 → 1



array_length >0
no
yes
i=0
a=0
i<array_length
no
yes
i == idx
no
yes
i++
continue
tmp_array[a++]=the_array[i++]



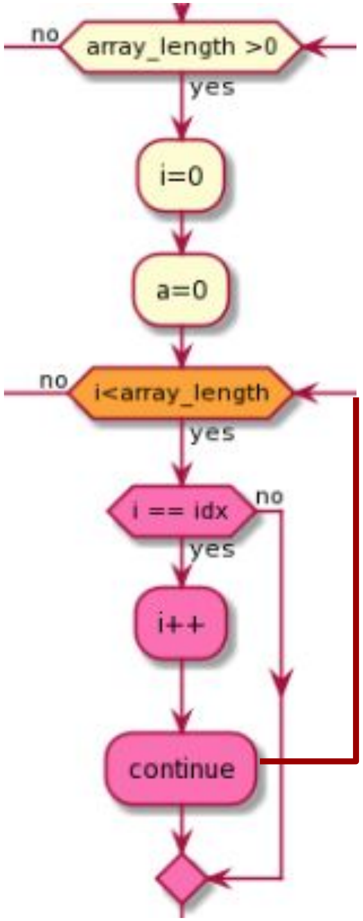
Simple sort

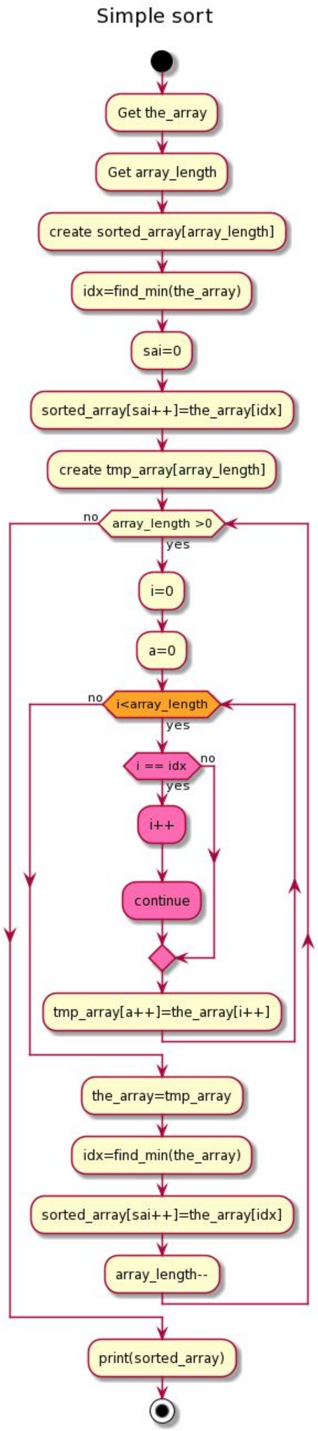
Get the_array
Get array_length
create sorted_array[array_length]
idx=find_min(the_array)
sai=0
sorted_array[sai++]=the_array[idx]
create tmp_array[array_length]
array_length >0
no
yes
i=0
a=0
i<array_length
no
yes
i == idx
no
yes
i++
continue
tmp_array[a++]=the_array[i++]
the_array=tmp_array
idx=find_min(the_array)
sorted_array[sai++]=the_array[idx]
array_length--
print(sorted_array)

Copying(2) into tmp_array..1

Variable	Value
the_array[0]	12
the_array[1]	10
the_array[2]	-
the_array[3]	-
the_array[4]	-
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	10
sorted_arr[4]	-
sai++	3→4
i<arr_length	(1<2) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	2
idx	1
find_min()	1
arr_length >0	(2 > 0) Y
i++	1 → 2
i == idx	(1==1) Y
a++	0 → 1

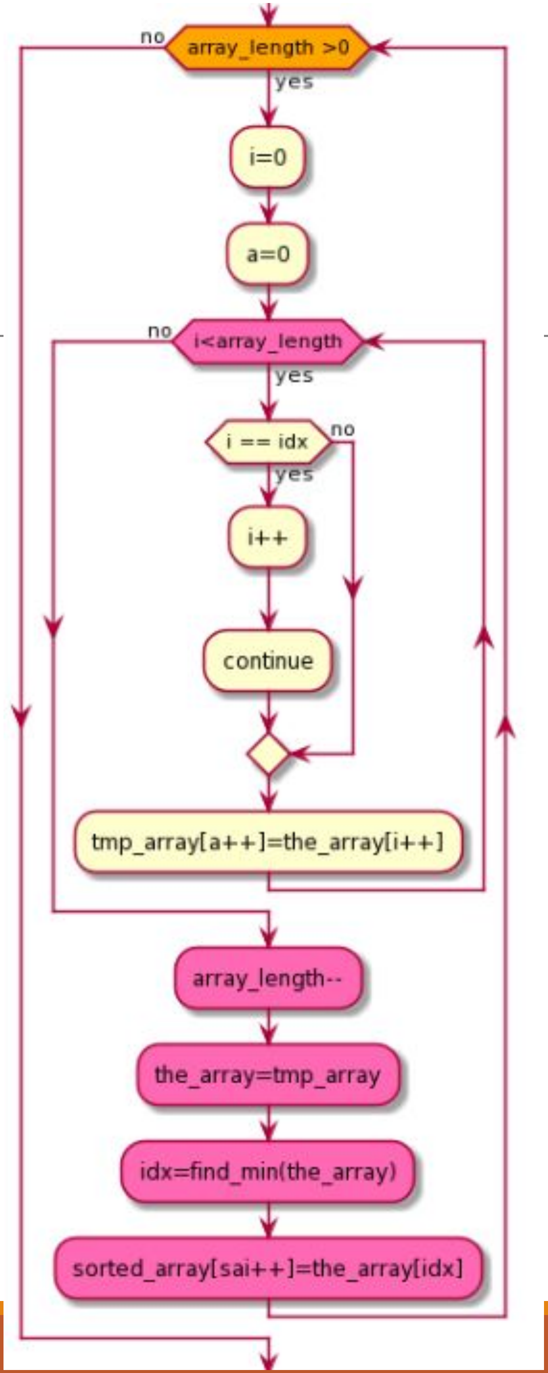


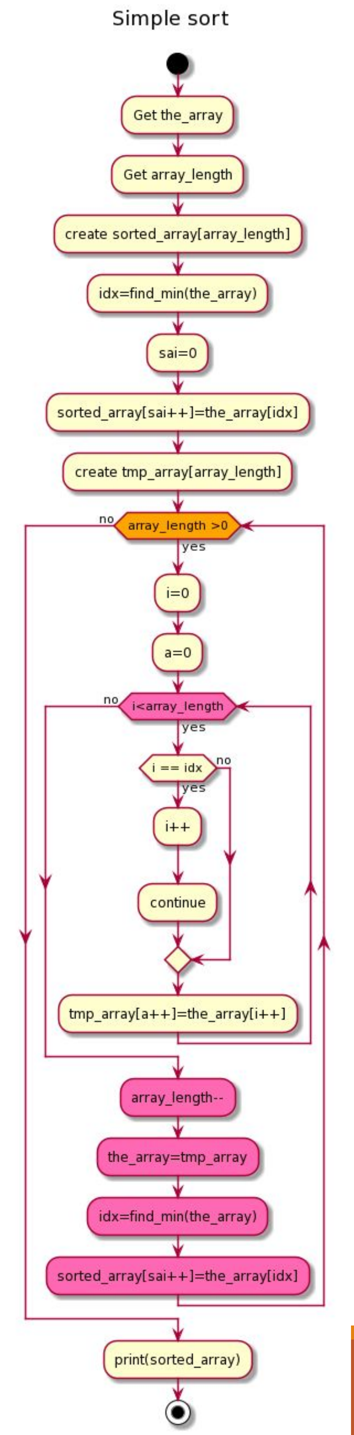


Update outer-loop

Variable	Value
the_array[0]	12
the_array[1]	-
the_array[2]	-
the_array[3]	-
the_array[4]	-
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	10
sorted_arr[4]	12
sai++	4→5
i<arr_length	(2<2) N

Variable	Value
tmp_array[0]	12
tmp_array[1]	-
tmp_array[2]	-
tmp_array[3]	-
tmp_array[4]	-
arr_length--	2 → 1
idx	0
find_min()	0
arr_length >0	(2 > 0) Y
i	2
i == idx	(1==1) Y
a++	0 → 1

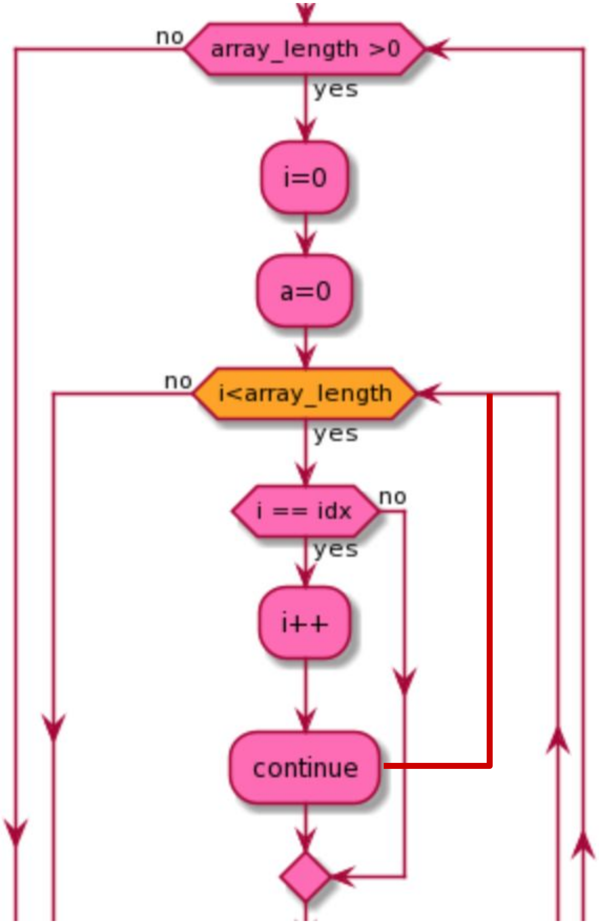




Copying(1) into tmp_array..0

Variable	Value
the_array[0]	12
the_array[1]	-
the_array[2]	-
the_array[3]	-
the_array[4]	-
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	10
sorted_arr[4]	12
sai++	4→5
i<arr_length	(0 < 1) Y

Variable	Value
tmp_array[0]	12
tmp_array[1]	
tmp_array[2]	
tmp_array[3]	
tmp_array[4]	
arr_length	1
idx	0
find_min()	0
arr_length >0	(1 > 0) Y
i++	0 → 1
i == idx	(0==0) Y
a	0



no

array_length >0

yes

i=0

a=0

no

i<array_length

yes

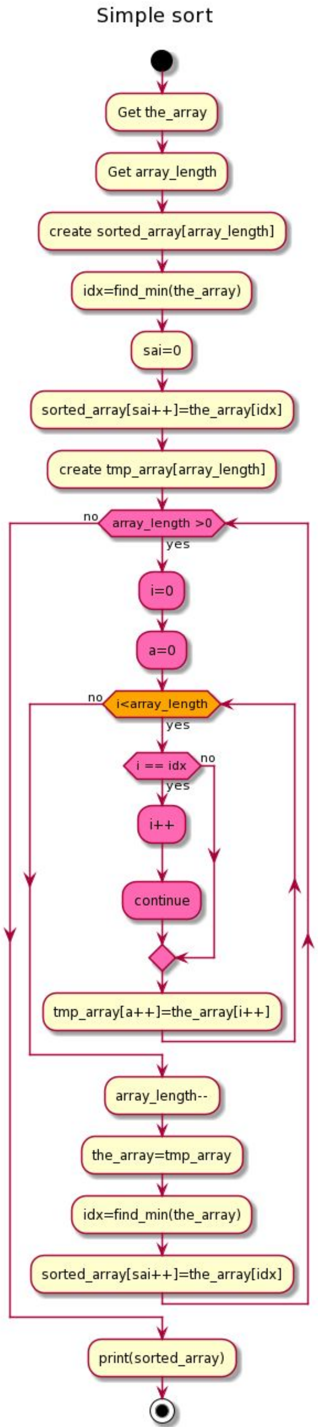
i == idx

no

yes

i++

continue



Simple sort

Get the_array

Get array_length

create sorted_array[array_length]

idx=find_min(the_array)

sai=0

sorted_array[sai++]=the_array[idx]

create tmp_array[array_length]

no

array_length >0

yes

i=0

a=0

no

i<array_length

yes

i == idx

no

yes

i++

continue

tmp_array[a++]=the_array[i++]

array_length--

the_array=tmp_array

idx=find_min(the_array)

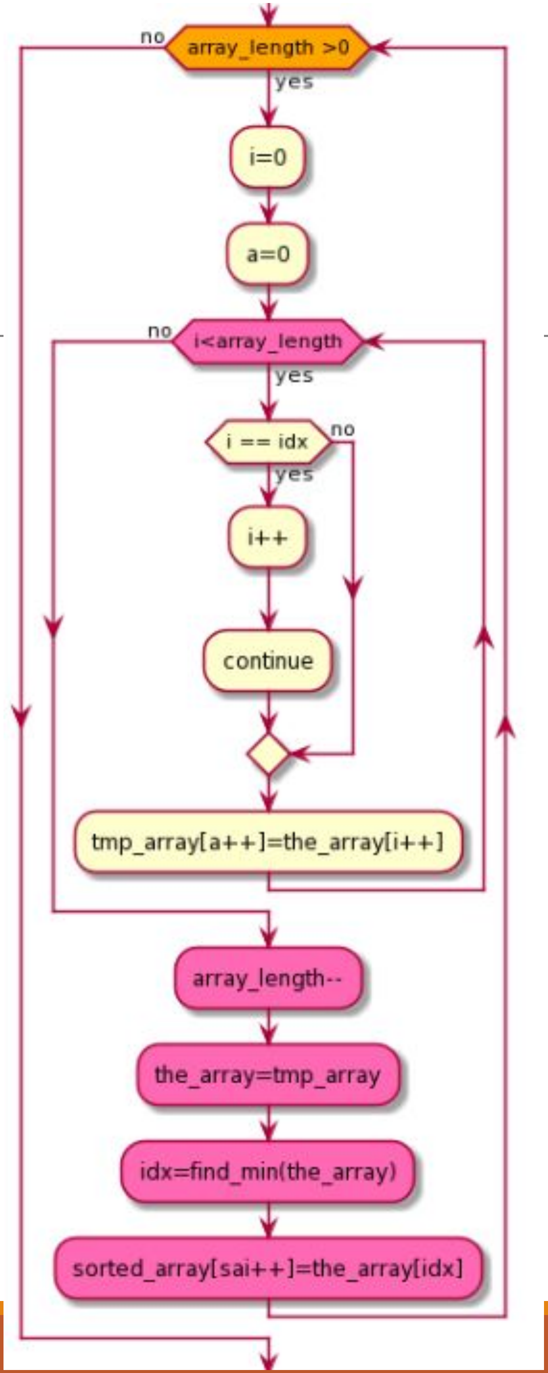
sorted_array[sai++]=the_array[idx]

print(sorted_array)

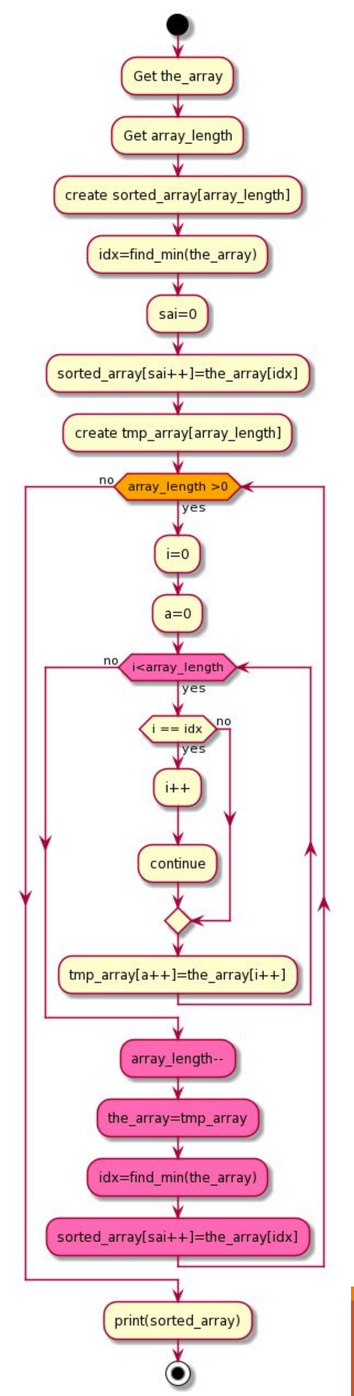
Update outer-loop

Variable	Value
the_array[0]	12
the_array[1]	-
the_array[2]	-
the_array[3]	-
the_array[4]	-
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	10
sorted_arr[4]	12
sai++	5→6
i<arr_length	(1<1) N

Variable	Value
tmp_array[0]	12
tmp_array[1]	-
tmp_array[2]	-
tmp_array[3]	-
tmp_array[4]	-
arr_length--	1 → 0
idx	-
find_min()	-
arr_length >0	(1 > 0) Y
i	1
i == idx	(1==1) Y
a++	0 → 1



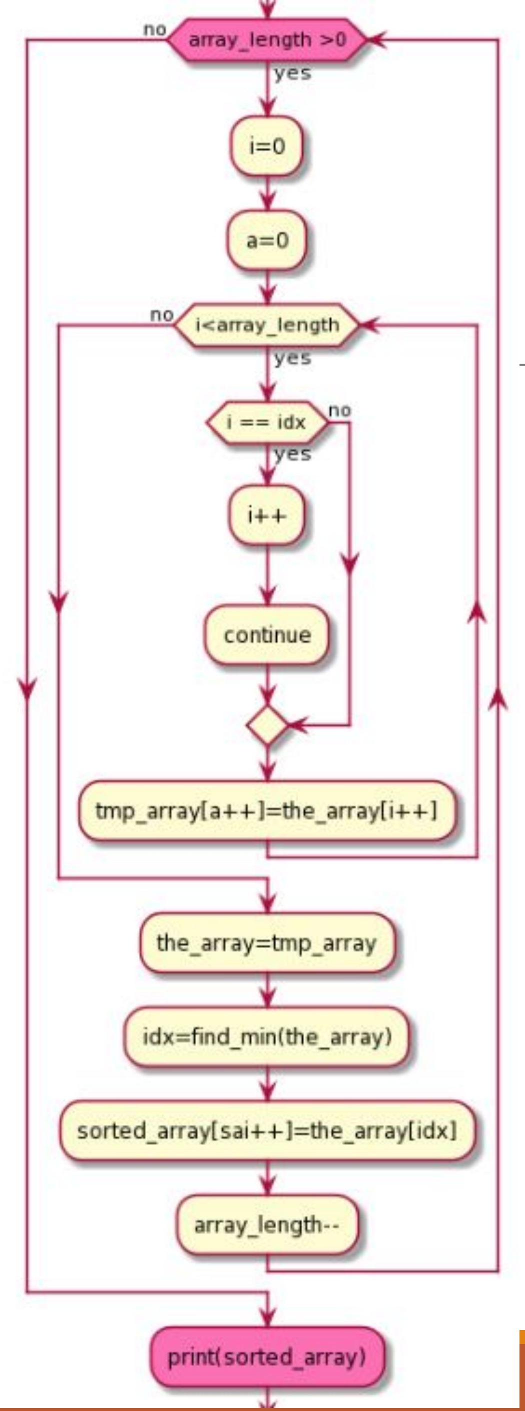
Simple sort



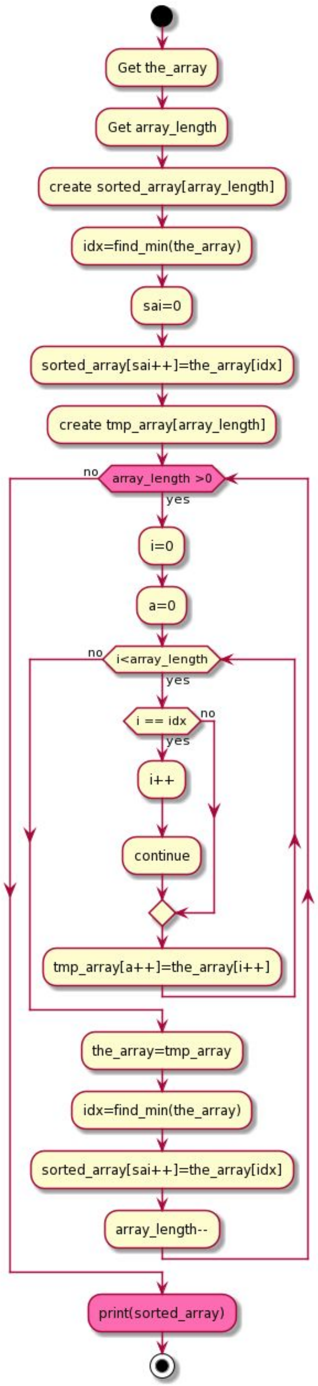
Output sorted array

Variable	Value
the_array[0]	12
the_array[1]	-
the_array[2]	-
the_array[3]	-
the_array[4]	-
sorted_arr[0]	2
sorted_arr[1]	5
sorted_arr[2]	6
sorted_arr[3]	10
sorted_arr[4]	12
sai++	5→6
i<arr_length	(1<1) N

Variable	Value
tmp_array[0]	12
tmp_array[1]	-
tmp_array[2]	-
tmp_array[3]	-
tmp_array[4]	-
arr_length	0
idx	-
find_min()	-
arr_length >0	(0 > 0) N
i	1
i == idx	(1==1) Y
a++	0 → 1



Simple sort



Simple sort strategy code

```
void ssort(int arr[], int n)
{
        int sorted_arr[n];
        int idx=findMin(arr,n);
        int sai=0;
        sorted_arr[sai++]=arr[idx];
        int tmp_array[n];
        for (int i = n; i > 0; --i) {
                int a=0;
                for (int j = 0; j < i; j++) {
                        if (j == idx)continue;
                        tmp_array[a++]=arr[j];
                }
                for(int
k=0;k<i-1;k++)arr[k]=tmp_array[k];
                idx = findMin(arr,i-1);
                sorted_arr[sai++] = arr[idx];
        }
        printArray(sorted_arr,n);
}
```

```
start
:Get the_array;
:Get array_length;
:create sorted_array[array_length];
:idx=find_min(the_array);
:sai=0;
:sorted_array[sai++]=the_array[idx];
:create tmp_array[array_length];
while (array_length>0) is (yes)
  :i=0;
        :a=0;
   while(i<array_length) is (yes)
   if(i == idx) is (yes) then
     :i++
     :continue;
   else (no)
   endif
   :tmp_array[a++]=the_array[i++];
   end while (no)
   :array_length--;
   :the_array=tmp_array;
   :idx=find_min(the_array);
   :sorted_array[sai++]=the_array[idx];
end while (no)
:print(sorted_array);
stop
```

Exercise

Study the insertion sort algorithm from the links provided on Brightspace and in your personal study and describe the algorithm using a simple visual walk-through strategy.

Test-oriented strategies

- Incremental development
- Assertive tests
- Test driven assertion
- Test header file

Exercise

Write a test driven find function that finds an element in array and returns the index. The program should have a test class which runs a test_find() method that tests your function under varying sizes of arrays and find target integers within and not contained within the array.

Any Questions?