COMP108
Algorithmic Foundations

Searching

Prudence Wong

http://www.csc.liv.ac.uk/~pwong/teaching/comp108/201617

Searching

- Input: n numbers $a_1, a_2, \ldots, a_n$; and a number X
- Output: determine if X is in the sequence or not
- Algorithm (Sequential search):
  1. From i=1, compare X with a_i one by one as long as i <= n.
  2. Stop and report "Found!" when X = a_i .
  3. Repeat and report "Not Found!" when i > n.

Sequential Search

To find 7

➢	**12** **7**	34	2	9	7	5 ←six numbers ←number X
➢	12	**34** **7**	2	9	7	5
➢	12	34	**2** **7**	9	7	5
➢	12	34	2	**9** **7**	7	5
➢	12	34	2	9	**7** **7**	5 found!

Sequential Search (2)

To find 10

➢	**12** **10**	34	2	9	7	5
➢	12	**34** **10**	2	9	7	5
➢	12	34	**2** **10**	9	7	5
➢	12	34	2	**9** **10**	7	5
➢	12	34	2	9	**7** **10**	5
➢	12	34	2	9	7	**5** **10** not found!

Pseudo Code - Ideas

variable `i` to step through the array
boolean `found` to indicate whether X is found

```
i = ?
found = ?
while i <= ? && found == ? do
begin
  /* check whether the i-th entry of
  the array equals X and set found
  accordingly */
  i = i+1
end
if found==true then
  report "Found!"
else report "Not Found!"
```


Pseudo Code

```
i = 1
found = false
while i <= n && found == false do
begin
  if X == a[i] then
    found = true
  else
    i = i+1
end
if found==true then
  report "Found!"
else report "Not Found!"
```


Number of comparisons?

```
i = 1
found = false
while i<=n && found==false do
begin
  if X == a[i] then
    found = true
  else
    i = i+1
end
```

How many comparisons this algorithm requires?

Best case: X is 1st no.
⇒ 1 comparison

Worst case: X is last
OR X is not found
⇒ n comparisons

Finding maximum / minimum...

2nd max / min...

Finding max from n +ve numbers

```
input: a[1], a[2], ..., a[n]
i = 1
M = 0
while (i <= n) do
begin
  if a[i] > M then
    M = a[i]
  i = i + 1
end
output M
```

What about minimum?

Skeleton is the same as before:

```
i = 1
while (i <= n) do
begin
  i = i + 1
end
```


Finding **min** from n +ve numbers

```
input: a[1], a[2], ..., a[n]
i = 1
M = a[1]
while (i <= n) do
begin
  if a[i] < M then
    M = a[i]
  i = i + 1
end
output M
```

How many comparisons?

Finding **location** of minimum

```
input: a[1], a[2], ..., a[n]
loc = 1 // location of the min number
i = 2
while (i <= n) do
begin
  if (a[i] < a[loc]) then
    loc = i
  i = i + 1
end
output a[loc]
```

Example

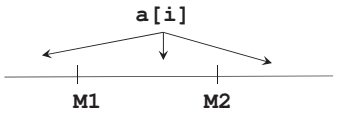
a[1..5]={50,30,40,20,10}

(@ end of) Iteration	loc	a[loc]	i
	1	50	2
1	2	30	3
2	2	30	4
3	4	20	5
4	5	10	6

Finding 1st and 2nd min

for simplicity, assume function min() and max()

```
input: a[1], a[2], ..., a[n]
M1 = min(a[1], a[2])
M2 = max(a[1], a[2])
i = 3
while (i <= n) do
begin
  if (a[i] < M1) then
    M2 = M1, M1 = a[i]
  else if (a[i] < M2) then
    M2 = a[i]
  i = i + 1
end
output M1, M2
```



Finding 1st and 2nd min

for simplicity, assume function min() and max()

```
input: a[1], a[2], ..., a[n]
M1 = min(a[1], a[2])
M2 = max(a[1], a[2])
i = 3
while (i <= n) do
begin
  // how to update M1 & M2?

  i = i + 1
end
output M1, M2
```

Two variables: M1, M2

3 cases: