YUQIAN ZHANG #N19945556.

Fundamental Algorithms. Problem Set 2.

Q1. Operation of PARTITION (A, 1, 12) on the array

A = (13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 10)

1 2 3 4 5 6 7 8 9 10 11 12

A = (13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, (10))

B = (9, 5, 8, 7, 4, 2, 6, 10, 11, 12, 19, 13)

At Beginning: left pointer at position 1

right pointer at position 12

pivot x = A[12] = 10

for j = 1 to 11 :

j = 1	A[1] = 13 > 10	B[right] = 13	right = 11
j = 2	A[2] = 19 > 10	B[right] = 19	right = 10
j = 3	A[3] = 9 < 10	B[left] = 9	left = 2
j = 4	A[4] = 5 < 10	B[left] = 5	left = 3
j = 5	A[5] = 12 > 10	B[right] = 12	right = 9
j = 6	A[6] = 8 < 10	B[left] = 8	left = 4
j = 7	A[7] = 7 < 10	B[left] = 7	left = 5
j = 8	A[8] = 4 < 10	B[left] = 4	left = 6
j = 9	A[9] = 11 > 10	B[right] = 11	right = 8
j = 10	A[10] = 2 < 10	B[left] = 2	left = 7
j = 11	A[11] = 6 < 10	B[left] = 6	left = 8

B[left] = x (B[8] = 10)

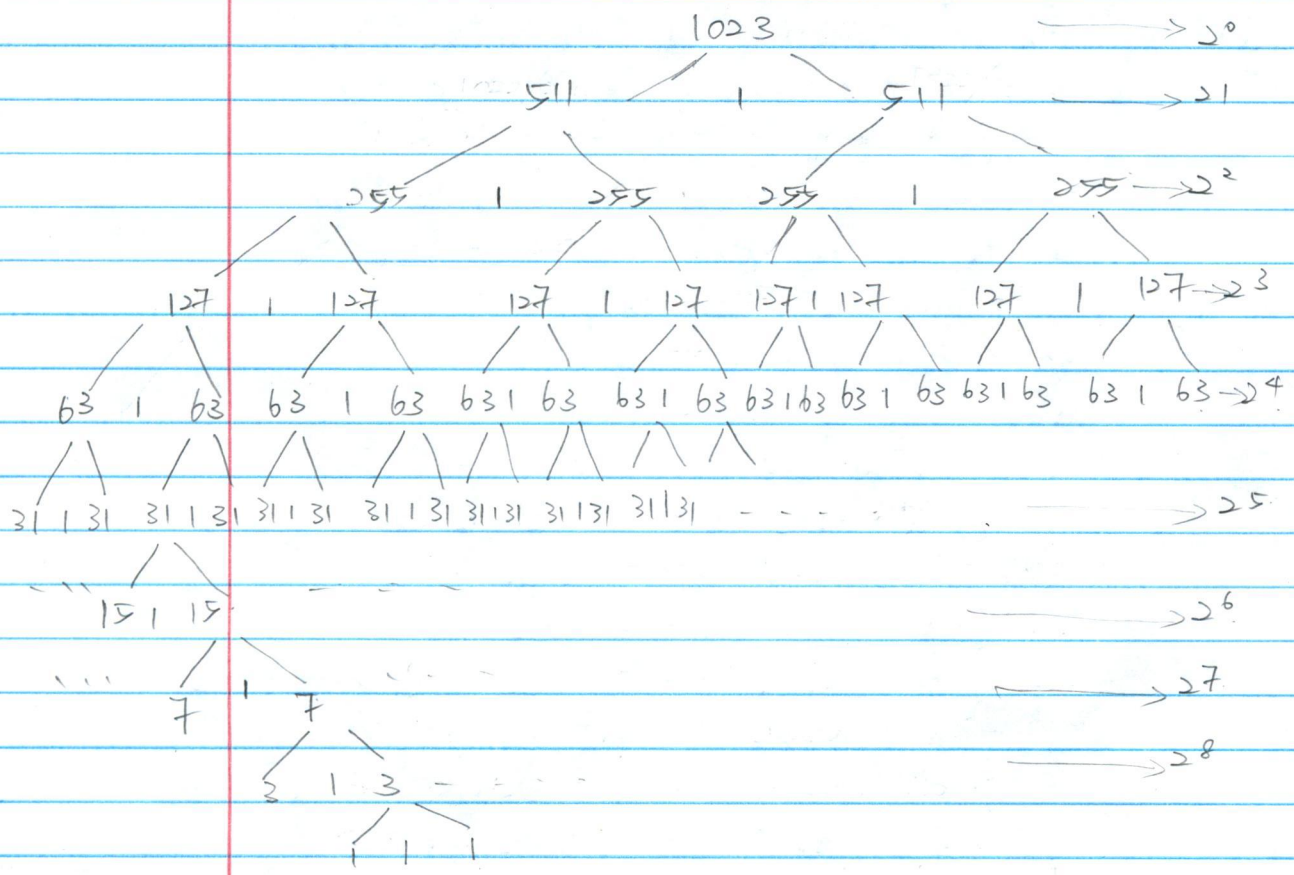
for j = 1 to 12

A[j] ← B[j] (reset the A vector)

return left (which is 8)

By using the version given in class, finally it will return 8. as pivot position

Q2. precise value of $\angle(1023)$



$1023 \times 2^0 = 1023$

$511 \times 2^1 = 1022$

$254 \times 2^2 = 1016$

$126 \times 2^3 = 1008$

$62 \times 2^4 = 992$

$30 \times 2^5 = 960$

$14 \times 2^6 = 896$

$6 \times 2^7 = 768$

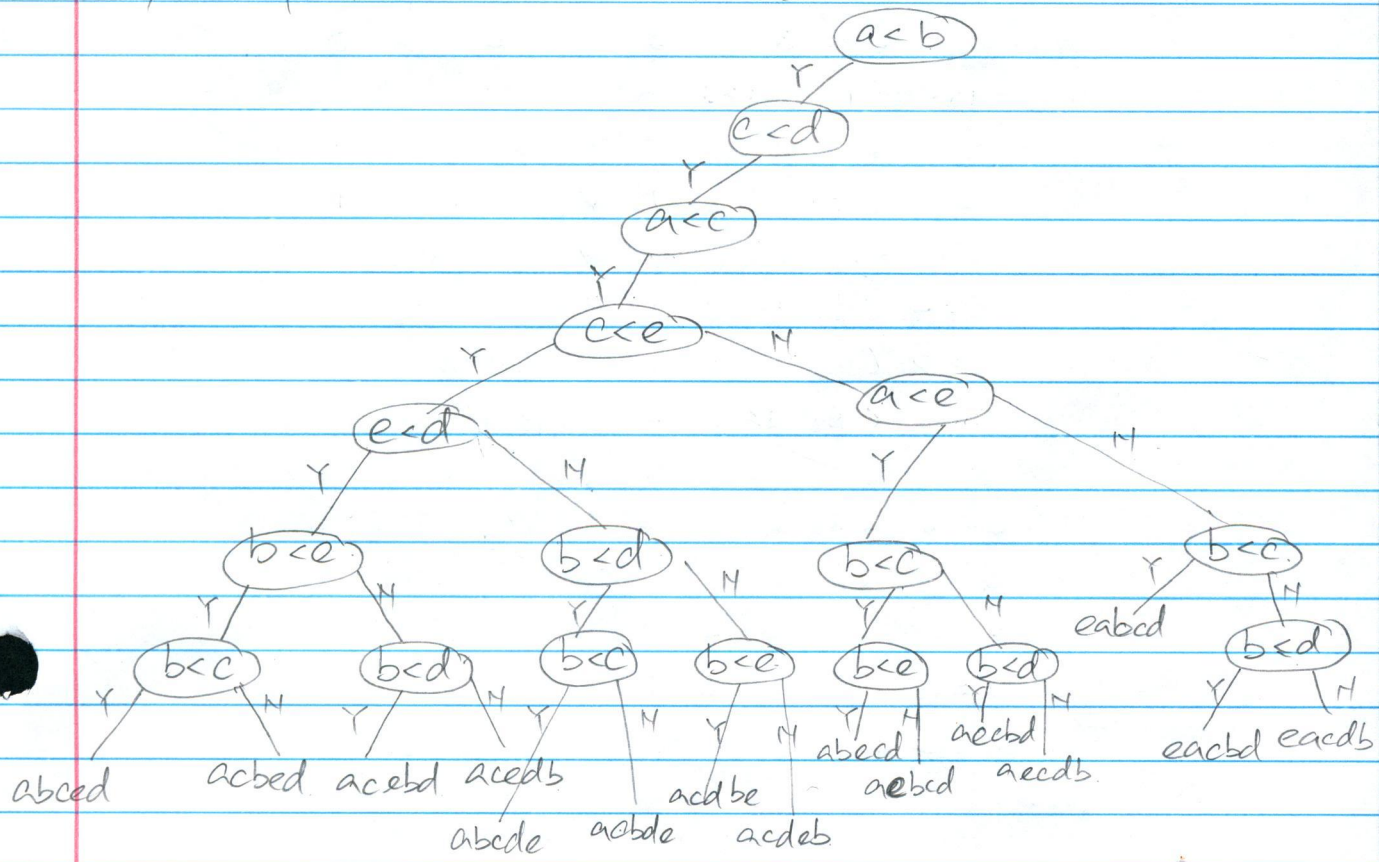
$+ \quad 2 \times 2^8 = 512$

8194

Thus, precise value of $\angle(1023) = 8194$

Q3. Assume that you already know that $a<b$, $c<d$, $a<c$
Sort the elements with 4 further comparisons.

By assumption: $a<b$. $c<d$ $a<c$



Q4. If 'a<b' Answer is Yes 'a<c'? will NOT succeed.

Lets say $a<b$ is confirmed, Then

a) If $a<c$ is Yes: There are two possibilities

$a<c<b$ ①
$a<b<c$ ②

In both case ① and ②, there are 4×5 ways to insert e d
Thus, in total 40 ways. And this is the worst case.

$40 < 2^6$ at least 6 further questions (comparisons) should be asked

b) If $a<c$ is NO.: we will have $c<a<b$

In this case, there are 4×5 ways to insert e d

$20 < 2^5$ at least 5 further questions should be asked

In summary, he will not succeed by asking 'Is $a<c$?'

Q7. counting-sort k=6 A=(6, 0, 2, 2, 0, 1, 3, 4, 6, 1, 3)

A=(6, 0, 2, 2, 0, 1, 3, 4, 6, 1, 3)

k=6

Auxilliary C=(0, 0, 0, 0, 0, 0, 0)

For S=1 to N: (C[A[S]]++)

S=1 C=(0, 0, 0, 0, 0, 0, 1)

S=2 C=(1, 0, 0, 0, 0, 0, 1)

S=3 C=(1, 0, 1, 0, 0, 0, 1)

S=4 C=(1, 0, 2, 0, 0, 0, 1)

S=5 C=(2, 0, 2, 0, 0, 0, 1)

S=6 C=(2, 1, 2, 0, 0, 0, 1)

S=7 C=(2, 1, 2, 1, 0, 0, 1)

S=8 C=(2, 1, 2, 1, 1, 0, 1)

S=9 C=(2, 1, 2, 1, 1, 0, 2)

S=10 C=(2, 2, 2, 1, 1, 0, 2)

S=11 C=(2, 2, 2, 2, 1, 0, 2)

For S=1 to k: C[S]= C[S] + C[S-1]

then C=(2, 4, 6, 8, 9, 9, 11)

For J= N Down to 1

N=11 value=A[11]=3 place=C[3]=8 B[8]=3 C[3]--

B=(3)

C=(2 4 6 7 9 9 11)

N=10 value=A[10]=1 place=C[1]=4 B[4]=1 C[1]--

B=(1 3)

C=(2 3 6 7 9 9 11)

N=9 value=A[9]=6 place=C[6]=11 B[11]=6 C[6]--

B=(1 3 6)

C=(2 3 6 7 9 9 10)

N=8 value=A[8]=4 place=C[4]=9 B[9]=4 C[4]--

B=(1 3 4 6)

C=(2 3 6 7 8 9 10)

N = 7 value = A[7] = 3 place = C[3] = 7 B[7] = 3 C[3] --

B = (1 2 3 4 5 6 7 8 9 10 11
      _ _ _ 1 _ _ 3 3 4 _ 6)

C = (0 1 2 3 4 5 6
      2 3 6 6 8 9 10)

N = 6 value = A[6] = 1 place = C[1] = 3 B[3] = 1 C[1] --

B = (1 2 3 4 5 6 7 8 9 10 11
      _ _ 1 1 _ _ 3 3 4 _ 6)

C = (2 2 6 6 8 9 10)

N = 5 value = A[5] = 0 place = C[0] = 2 B[2] = 0 C[0] --

B = (1 2 3 4 5 6 7 8 9 10 11
      _ 0 1 1 _ _ 3 3 4 _ 6)

C = (1 2 6 6 8 9 10)

N = 4 value = A[4] = 2 place = C[2] = 6 B[6] = 2 C[2] --

B = (1 2 3 4 5 6 7 8 9 10 11
      _ 0 1 1 _ 2 3 3 4 _ 6)

C = (1 2 5 6 8 9 10)

N = 3 value = A[3] = 2 place = C[2] = 5 B[5] = 2 C[2] --

B = (1 2 3 4 5 6 7 8 9 10 11
      _ 0 1 1 2 2 3 3 4 _ 6)

C = (1 2 4 6 8 9 10)

N = 2 value = A[2] = 0 place = C[0] = 1 B[1] = 0 C[0] --

B = (1 2 3 4 5 6 7 8 9 10 11
      0 0 1 1 2 2 3 3 4 _ 6)

C = (0 2 4 6 8 9 10)

N = 1 value = A[1] = 6 place = C[6] = 10 B[10] = 6 C[6] --

B = (1 2 3 4 5 6 7 8 9 10 11
      0 0 1 1 2 2 3 3 4 6 6) [Sorted]

C = (0 2 4 6 8 9 9) [Final Auxiliary]

Thus, final sorted array by counting sort is B = (0, 0, 1, 1, 2, 2, 3, 3, 4, 6, 6)

Q6. Max-Heap with n entries. Find third largest.

One way using Extract-Max twice the Maximum

Time $O(\lg n + \lg(n-1)) + O(1)$,

which is $O(\lg n)$

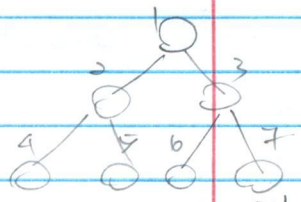
Better way: Because this is Max-Heap

If we are searching for third largest entry,

with property of MaxHeap,

It must be among node 2 to node 7.

for 2 to 7.



Thus, we can search the 3rd largest in constant time $O(1)$

Collaboration with Sijun Dou & Chao Gao