(93)

CMPE 130 Midterm Exam #1 Spring 2015

15:00—16:15 Thu[illegible]y March 3, 2015

Student Na[illegible] _____(print)

Student ID 00[illegible]_____

(8 points) Problem 1 (A): Fill into line 6 and line 7 of the pseudo code in the "**while loop**" of INSERSION-SORT below. (The INSERSION-SORT sorts the numbers in the array in ascending order.)

```
INSERTION-SORT (A)
1   for j=2 to A.length
2   key=A[j]
3   //insert  A[j] into the sorted sequence A[1, ..., j-1]//
4   i= j-1
5   while i>0 and A[i] > key
6       swap(A[i], A[i+1])
7       i=i-1
8   A[i+1] = key
```

A(i+1)=A(i)

2 3 4 1
2 3 1 4
2 1 3 4

(6 points) Problem 1(B): Apply INSERTION-SORT to array

31	41	59	26

and show the intermediate and final result for index j=2, 3 and 4.

J=2

31	41	59	26

J=3

31	41	59	26

J=4

26	31	41	59

✓

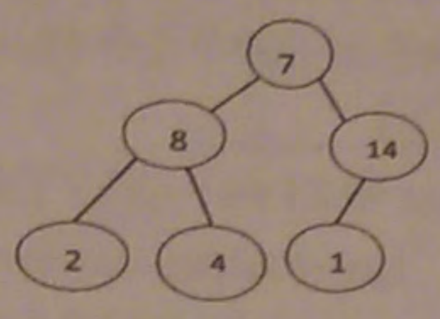
(6 points) Problem 1(C): Create an array out of the elements {41, 26, 59, 31} that has the worst performance under INSERTION-SORT.

Array that leads to the worst performance is

59	41	31	26

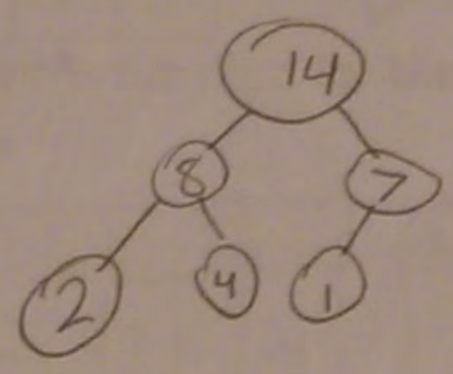
✓

(5 points) Problem 2(A): Is the data structure below a MAX-HEAP



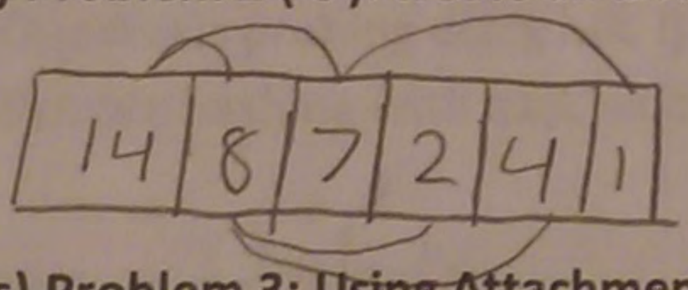
No.

(5 points) Problem 2 (B): If NO, apply MAX-HEAPIFY to convert it into a heap.
Write the resulting heap structure below.



✓

(5 points) Problem 2 (C): create an array corresponding to the result in 1(B)



✓

(15 points) Problem 3: Using Attachment #1 as a model, illustrate the operation of HEAPSORT on the array A<5, 13, 2, 25, 7, 17, 20, 8, 4>
Write your answer in the attached sheet.

✓

(15 points) Problem 4: QUICKSORT algorithm uses a core subroutine PARTITION.
Using Attachment #2 as a model, illustrate the operation of PARTITION on the array A<13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11>
Write your answer in the attached sheet.

✓

Problem 5 The Knuth-Morris-Pratt algorithm is widely used in matching an m-element pattern to an n-element string.

(5 points) 5 (A): What is the complexity of the brute-force string matching algorithm?

O(mn) ✓

(5 points) 5 (B): The KMP algorithm reduces the complexity to O(m+n) **by a pre-processing.**

(5 points) 5 (C): The pre-processing involves the use of prefix in the pattern. Compute the Prefix function of the KMP algorithm for the pattern below.

A	T	C	A	C	A	T	C

Fill in the answer prefix function( ) below.

0	0	0	1	0	1	2	3

✓

(5 points) 5(D) Write the prefix function for the pattern below

ababbabbabbababbabb

a	b	a	b	b	a	b	b	a	b	b	a	b	a	b	b	a	b	b
0	0	1	2	0	1	2	0	1	2	0	1	2	3	4	5	6	7	8

✓

Problem 6: Consider inserting keys 10,22, 31, 4, 15, 28, 17, 88, 59 into hash table of length m=11 using open addressing with the auxiliary hash function h(k)=k. Illustrate the result of inserting these keys using

(-7)

(6A) Linear probing (5 points) $(h(k)+i)\%m$

(6B) Quadratic probing using h(k, i)=h(k)+ i +3 i2 (5 points)**

(6C) Double hashing with h1(k)=k and h2(k)=1 + (k mod(m-1)) (5 points) $(h_1(k)+i\,h_2(k)$

Linear
$((h(k)+i)\%m)$

0	22	10%11=10
1	88	22%11=0
2		31%11=9
3		4%=4
4	4	-15%11=4
5	15	15+1%=5
6	28	28%11=6
7	17	-17%11=6
8	59	17+1%11=7
9	31	-88%11=0
10	10	88+1%11=1
		-59%11=4
		-59+1%11=5
		-59+2%11=6
		-59+3%11=7
		59+4%11=8

Quadratic
$((h(k)+i+3i^2)\%11)$

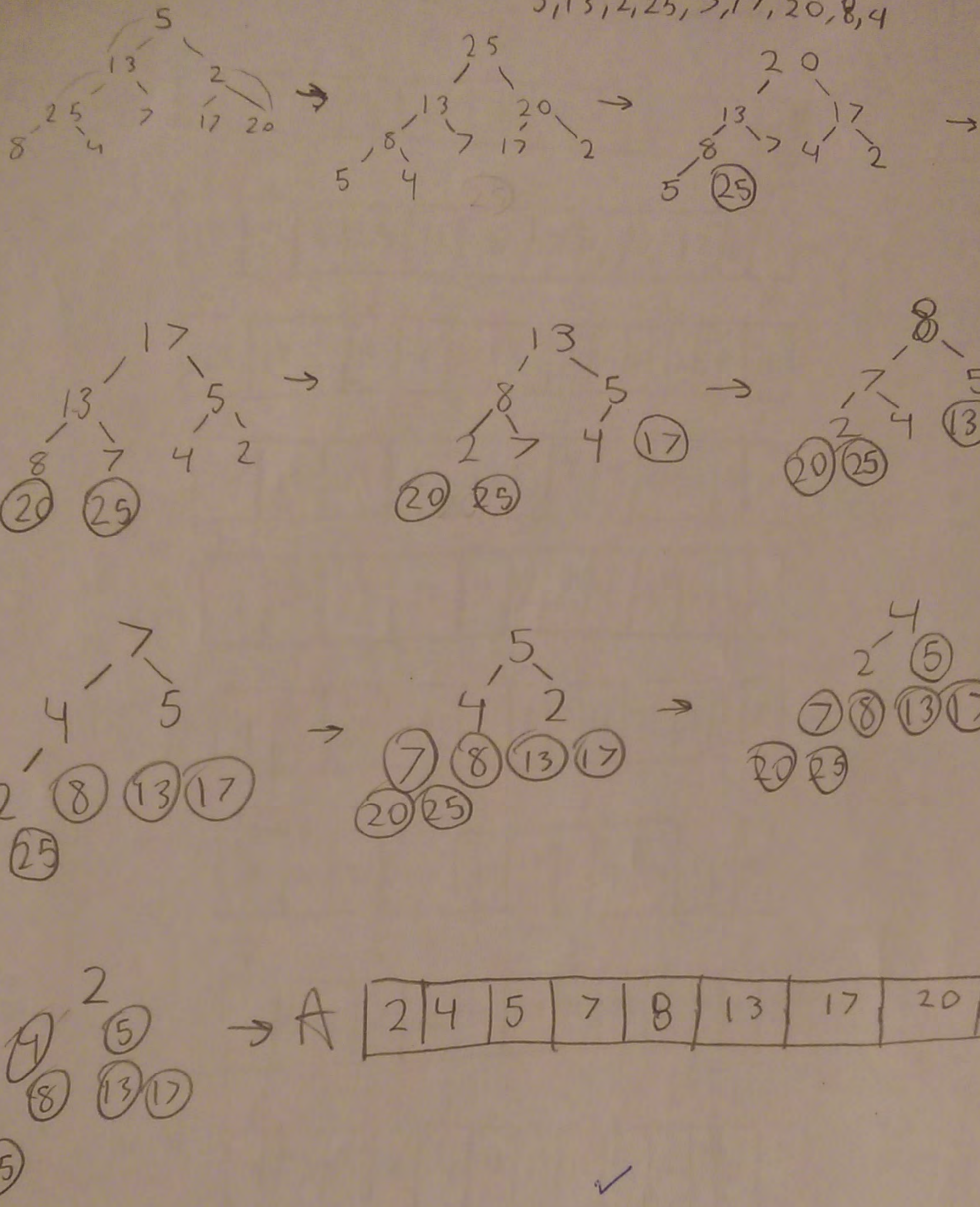
0	22	10%11=10
1	59	22%11=0
2	88	31%11=9
3	17	4%11=4
4	4	-15%11=4
5		15+4%11=8
6	28	28%11=6
7	~~88~~ 59	-17%11=6
8	15	-17+4%11=10
9	31	17+14%11=9
10	10	17+30%11=3
		-88%11=0
		-88+4%11=4
		-88+14%11=3
		88+30%11=7
		-59%11=4
		-59+4%11=8
		-59+14%11=7
		59+30%11=1

Double Hashing
$(h_1(k)+i\,h_2(k))\%m = (k+1\,(1+(k\%10)))\%11$

0	22	10%11=10
1	~~59~~	22%11=0
2	28 59	31%11=9
3	17	4%11=4
4	4	-15%11=4
5		15+6%11=10
6	15	15+12%11=6
7	88	-28%11=6
8		-28+9%11=4
9	31	28+18%11=2
10	10	-17%11=6
		17+8%11=3
		-88%11=0
		-88+9%11=9
		88+18%11=7
		-59%11=4
		-59+10%11=3
		-59+20%11=2
		59+30%11=1

Problem 3

5, 13, 2, 25, 7, 17, 20, 8, 4

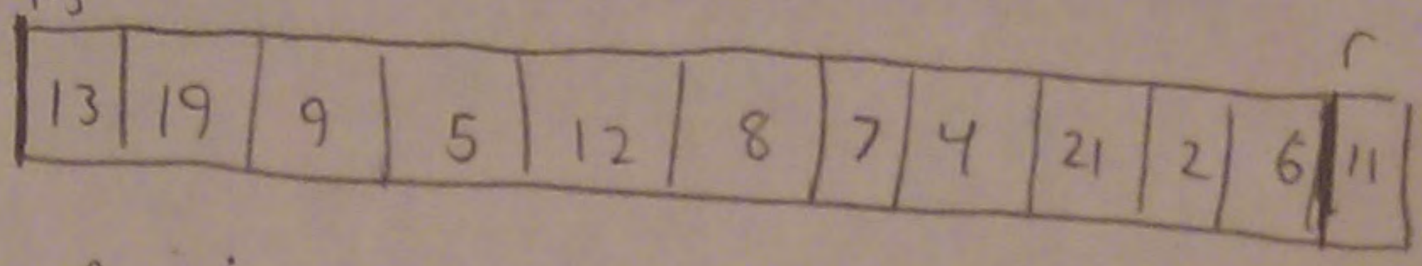


A

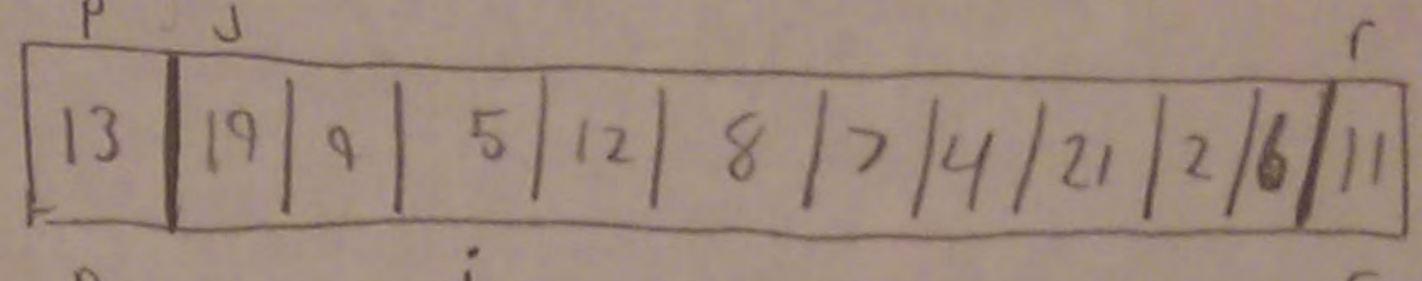
2	4	5	7	8	13	17	20	25

13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11

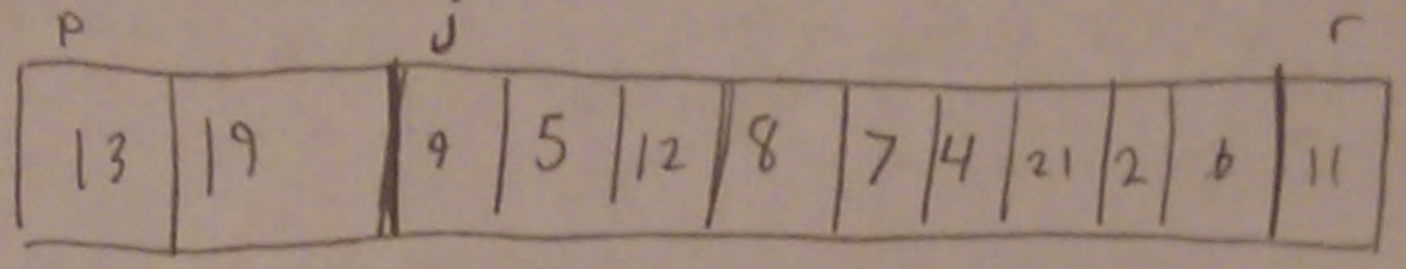
a



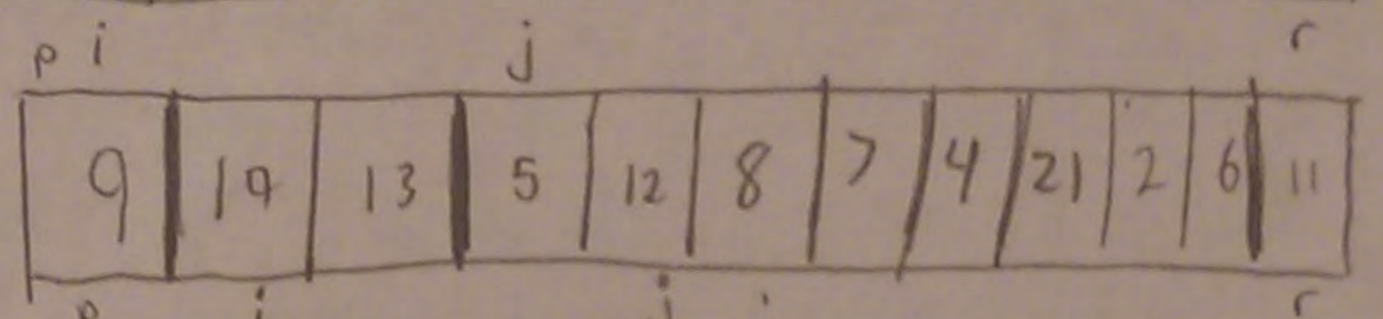
b



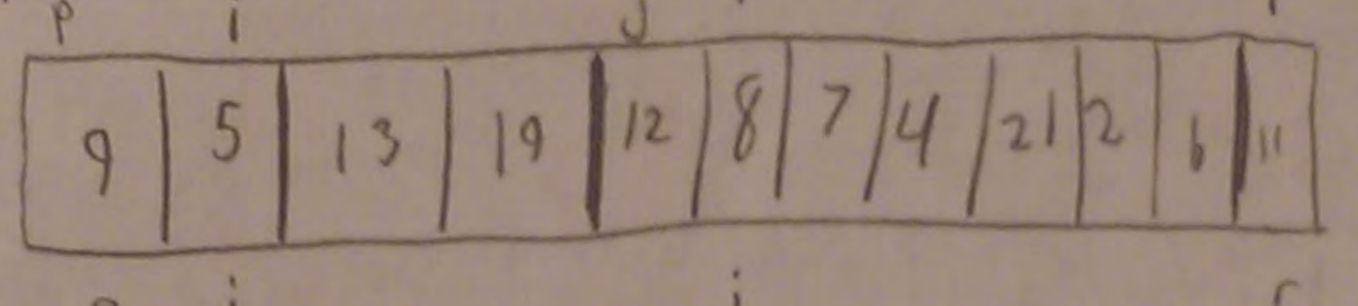
c



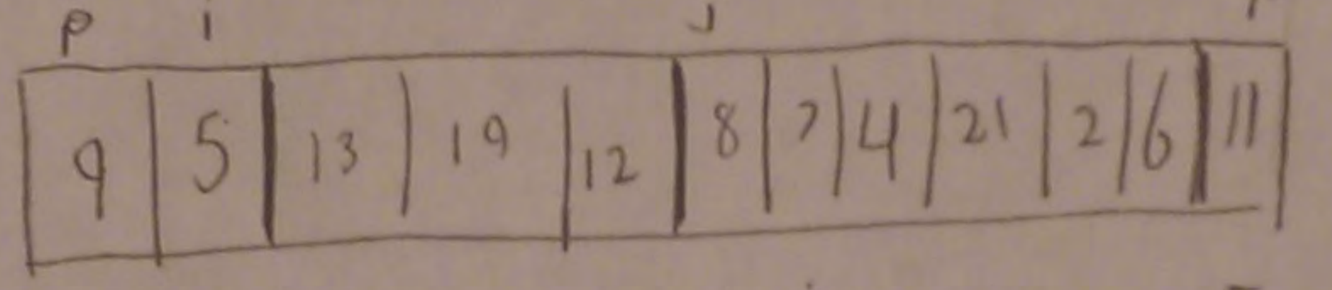
d



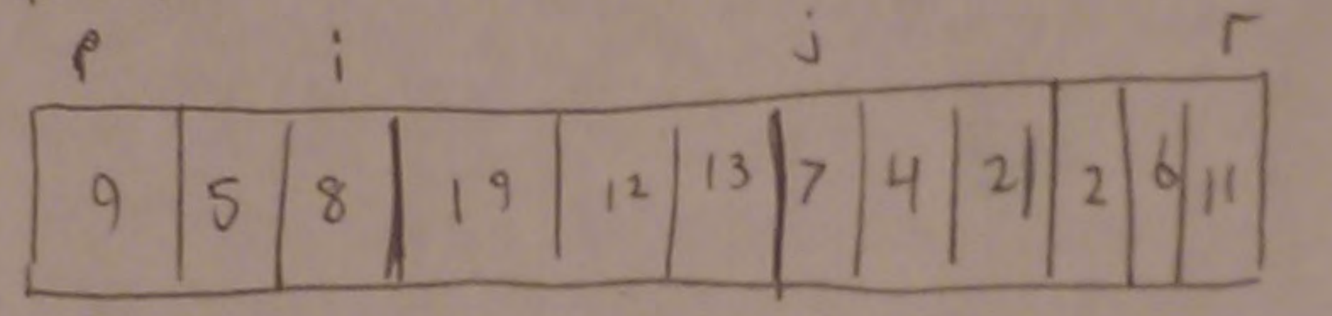
e



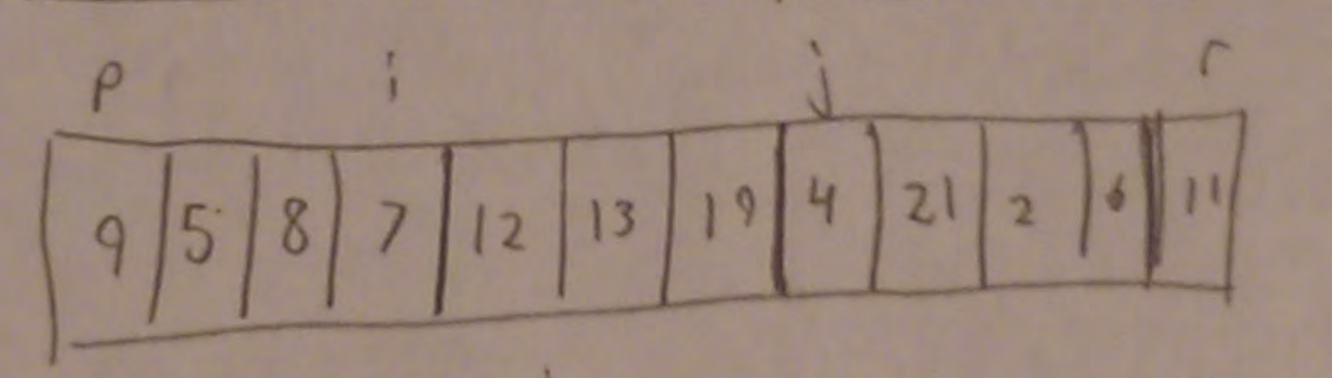
f



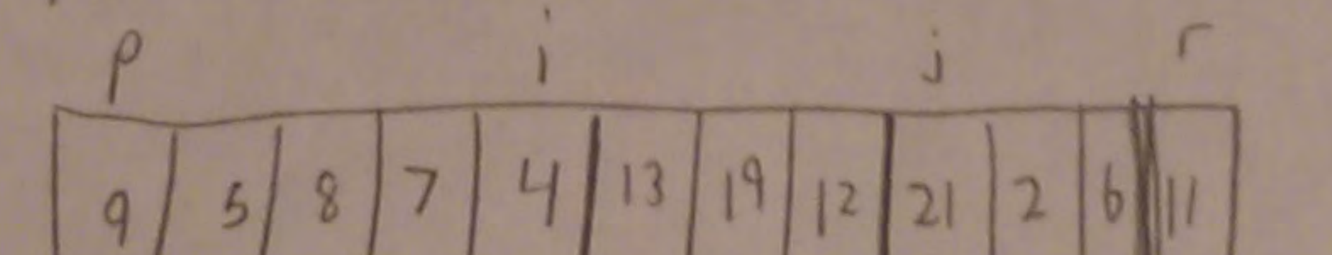
g



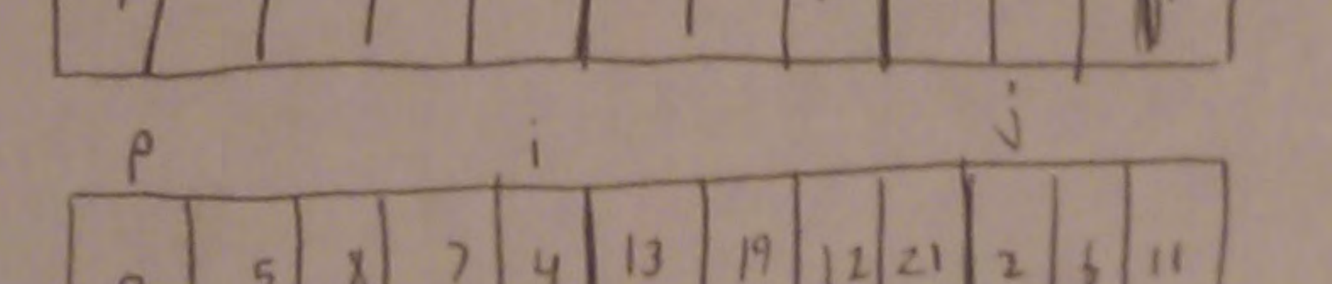
h



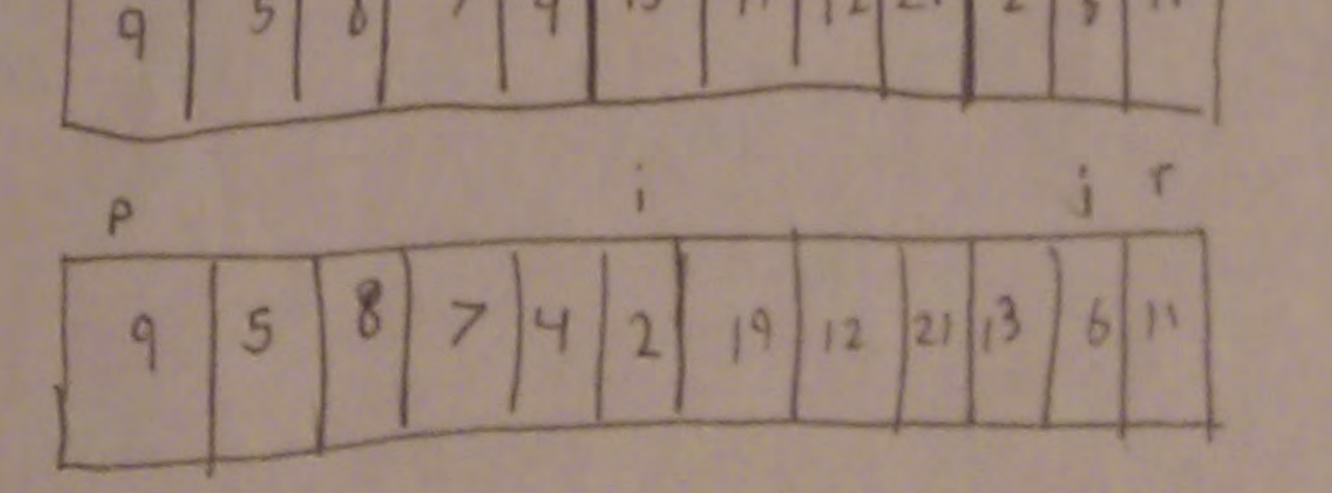
i



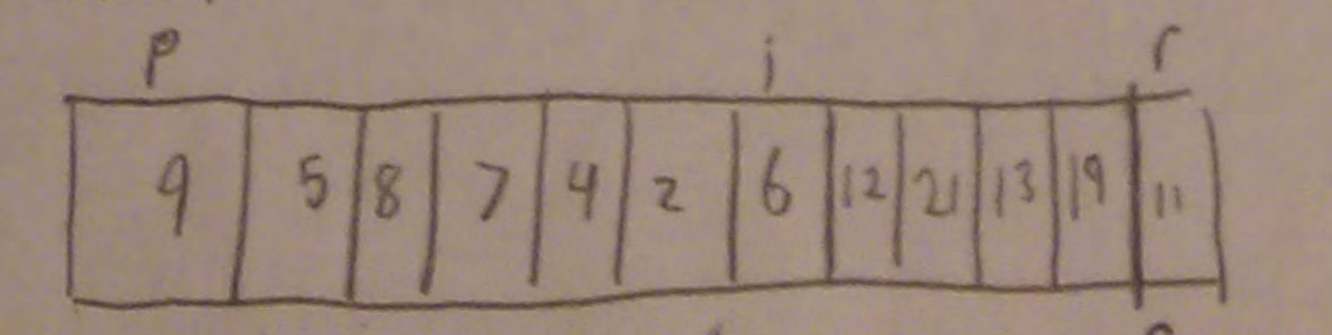
j



k



L



✓

m