Q1.

(a). if ~~root != null~~. inorder. traverial.

```
arr = [ ]
function inorder( root)
    if root != null.
        inorder (root. left)
        ~~root. val~~. arr. append( root. val)
        inorder (root. right)
value = arr[k].
O(n)
```

Q2: ROW = m.
COL = n.
def cmp(a, b)

(b). ~~min. 最小堆~~?

Modify TreeNode

add index in tree structure.

find from root, if root bigger than k. go to root. right, else. go to root. left.

```
~~Q2. DP. dis = 0~~
    ~~for i in (m+n-1)~~:

    ~~for i = 0 in n~~:
        int ~~k = MAX_VALUE~~.
        ~~for j in m~~:
```

Q2. DP. or. dijkstra. ~~alg~~

Q3. (Max matrix minuos Allocation)

(a) (1).

$$\begin{matrix} 0 & 0 & 5 & 1 & 2 \\ 0 & 2 & 1 & 5 & 2 \\ 5 & 0 & 1 & 2 & 0 \\ 1 & 2 & 0 & 0 & 2 \\ 4 & 0 & 0 & 3 & 6 \end{matrix}$$

(2).

P3 → P1 → P0 → { P2 → P4 / P4 → P2

(b)

~~Page size~~	~~segment~~	~~offset~~
~~7~~	~~11~~	~~21~~

24 bit

7	17
page number	~~Page~~ offset.

$2^{24} / 2^{7} = 2^{17}$

(c).

~~2M = 2~~

← 24 bit →

Segment	Page	offset
← 3 →	~~128~~ 14	~~21~~ 7

Segment size = 2^{21}

segmentation = $2^{24}/2^{21} = 2^{3}$

(c). <0, 857>

use 0 find frame number 8.

bytes is 1024. 8 × 1024 + 857.

<5.989>.

5 → 22. 22 × 1024 + 989

(a) (1) $\Pi_{name}(Courses) - \Pi_{cname}(course \bowtie Enroll)$.

(2) $n_1 \leftarrow G_{count(distinct\ cid)}(\sigma_{pid="p123"}(Teach))$

$\Pi_{Sname}\ Student \bowtie_{sid} G_{count(distinct\ cid)=n_1}(\sigma_{pid="p123"}(Enroll \bowtie Teach))$

(a) Give relational algebra queries for the tasks below:

(1) Find the names of all the courses that have not been taken by any student; **[3 marks]**

(2) Find the names of all students that have taken all the courses taught by the professor with pid = "p123"; **[3 marks]**

(b)(1) Select cname
from enroll nature join student nature join Course
group by cname having count(distinct dept)

(b) Write SQL queries for the tasks below:

(1) Find the names of all the courses that have been taken by students from more than one department; **[3 marks]**

(2) Find the name of the professor that teaches a course with the largest enrollment number; if there is more than one professor with the same largest enrollment number, report all of them; **[3 marks]**

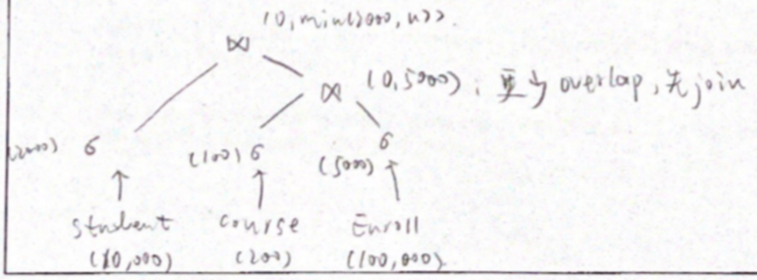
(b)(2) select pname
from prof nature join enroll nature join teach
group by cid having Count(sid) ≥ all ((count(sid)
group by
cid

(c) Consider the following SQL query:

select sname, cname, grade
from STUDENT, COURSE, ENROLL
where STUDENT.sid = ENROLL.sid
and COURSE.cid = ENROLL.cid
and dept = "SEEM"
and credit = 2
and grade = "A"

Assume that there are 10,000 students, and 20% of them are from SEEM department; there are 200 courses, and 50% of them have a credit of 2; and there are 100,000 tuples in ENROLL, and 5% of them have grade A.

Using the above information for optimizing a query, draw the most efficient query tree for the SQL query. You should clearly indicate all the selection and join operations on nodes in the tree. Justify your answer by calculating the size of intermediate results. **[8 marks]**



Q5.

(a). M: 3 O: 3 ~~N: 2~~ K: 6 E: 4 Y: 4. ~~D: 1 A: 2~~ C:3

MK: 3 OK: 3 OE: 3 KE: 4 KY: 4, CK: 3

OKE: 3

closed: K. ~~E~~. ~~Y~~. C KE. KY.

maximum: 超集都不是 frequent

(b) confi = $\frac{su(E \cup C)}{su(E\cancel{E})} = \cancel{\frac{4}{4}}\ \frac{1}{4}\ \cancel{84}$.

sup(EC) $\frac{1}{6}$

~~lift = $\frac{1}{4}$ / 1~~ sup(E) = $\frac{4}{6}$.

lift = $\frac{confi}{sup(E) \cdot sup(C)} = \frac{\frac{1}{4}}{\frac{4}{6} \times \frac{3}{6}}$ $\frac{EC}{C}$ sup(

(c).

(d)

Q6. (a) min (M_1, M_2) max : $M_1 + M_2$

(b)