一.

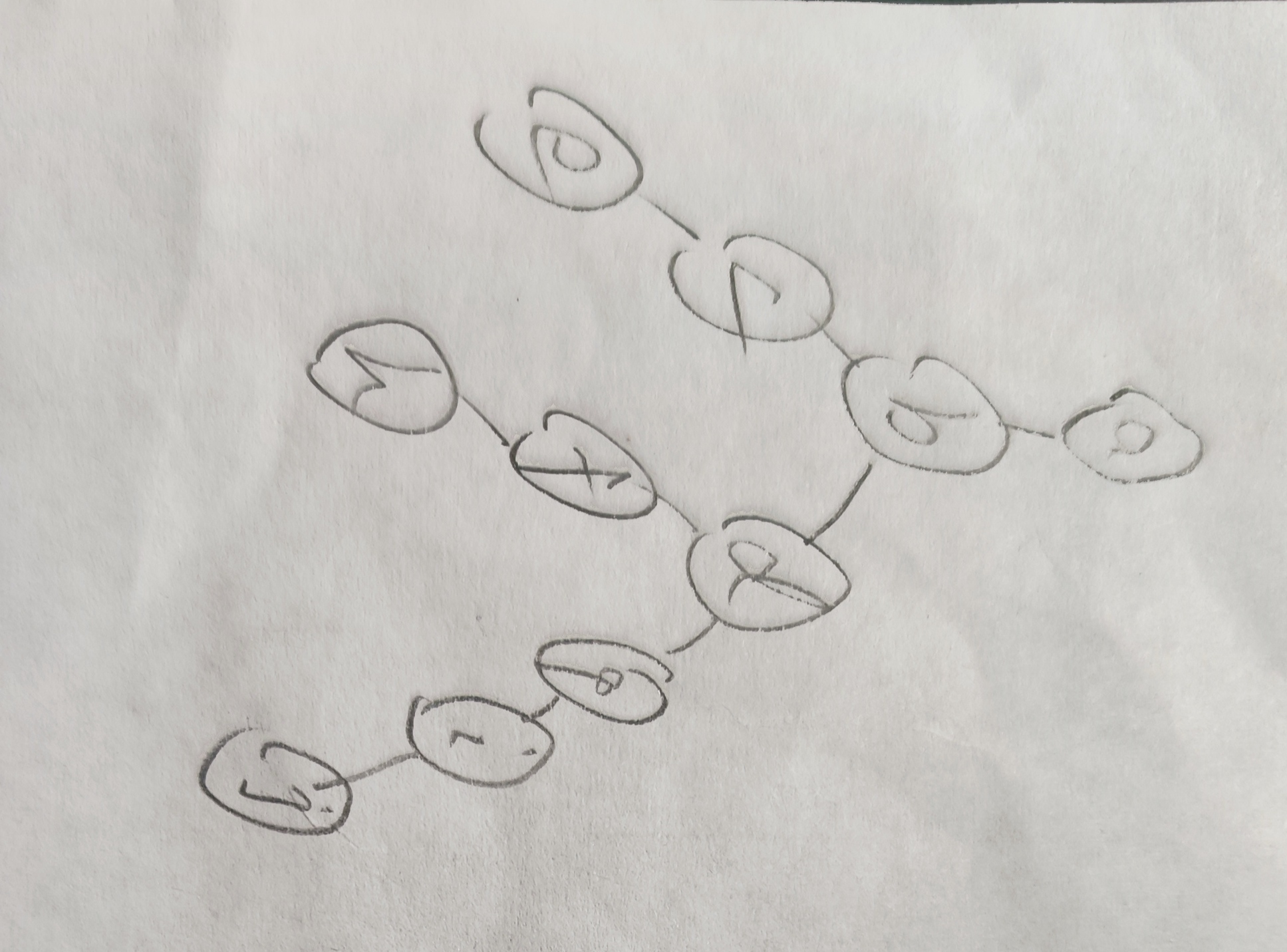
1. A
2. BEGD
3. 2
4. 3
5. 4
6. EF
7. A

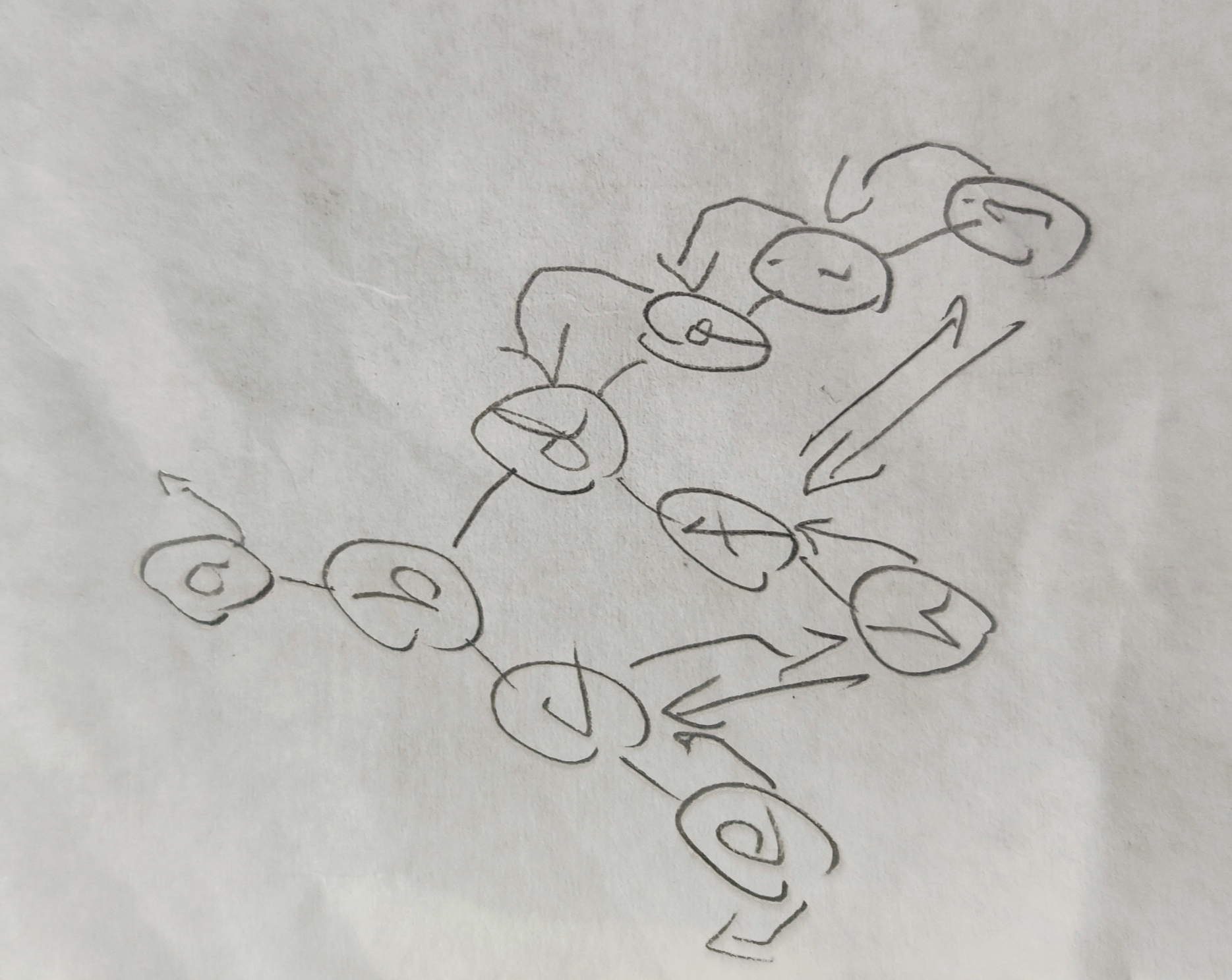
二．3+2\*2+3\*2=14

三.

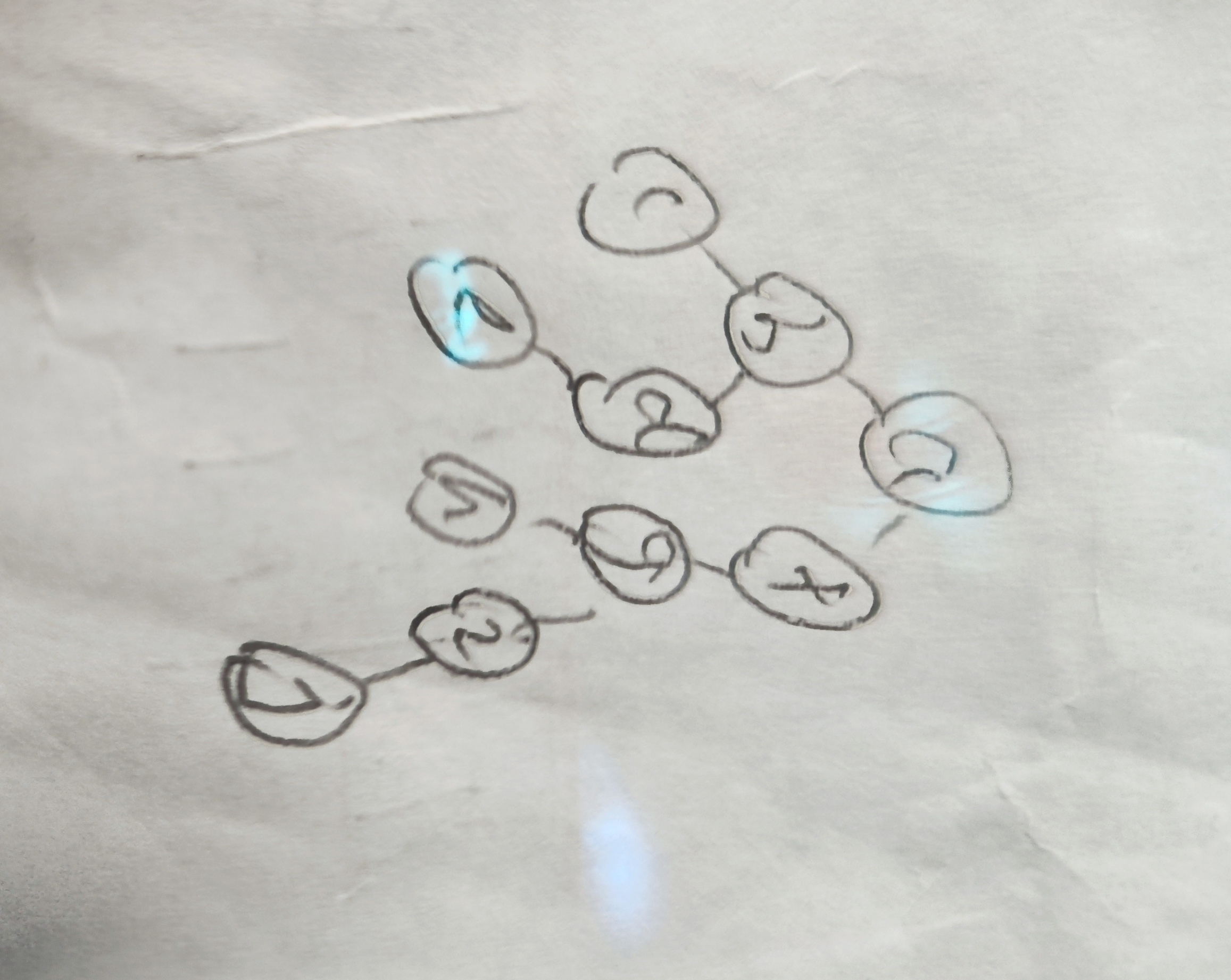
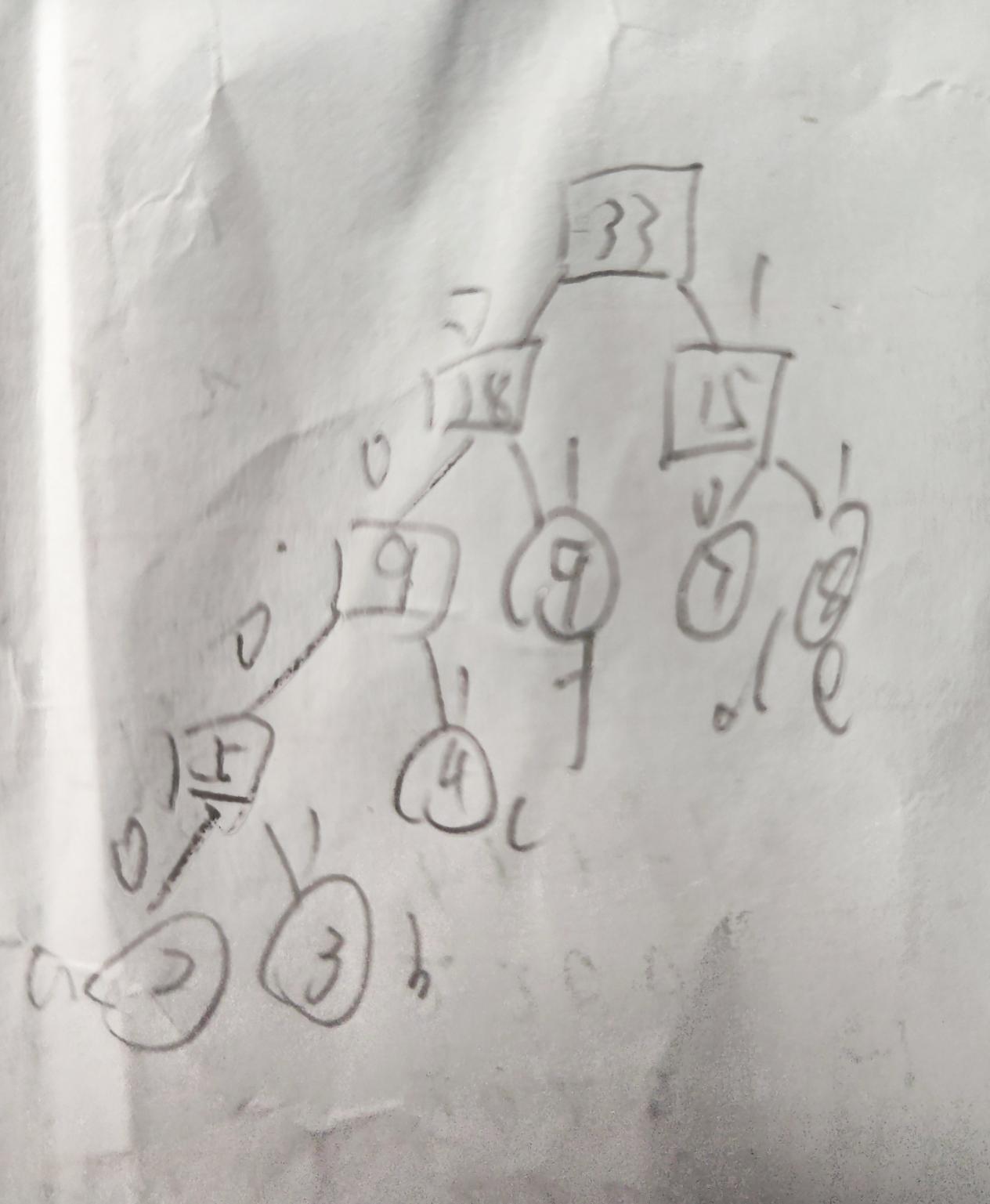
1. 双亲
2. 孩子链
3. 双亲
4. 孩子兄弟链
5. 孩子链
6. 孩子链

四.

1.

1. abcedfhgij ecbhfdjiga echfjigdba
2. 

六.111 39

1. 
2. 

void ctree(btnode \**b*,sqbtree *a*,int *i*)

{

    if(*b*!=NULL)

    {

*a*[*i*]=*b*->data;

        ctree(*b*->lchild,*a*,2\**i*);

        ctree(*b*->rchild,*a*,2\**i*+1);

    }

    else

*a*[*i*]='#';

}

十三.void findminnode(btnode \**b*,char &*min*)

{

    if(*b*->data<min)

    min=*b*->data;

    findminnode(*b*->lchild,min);

    findminnode(*b*->rchild,min);

}

void minnode(btnode \**b*)

{

    if(*b*!=NULL)

    {

        char min=*b*->data;

        findminnode(*b*,min);

        printf("min=%c\n",min);

}

}

十七.void output(btnode \**p*)

{

    if(*p*!=NULL)

    {

        printf("%c",*p*->data);

        output(*p*->lchild);

        output(*p*->rchild);

    }

}

void child(btnode \**b*,char *x*)

{

    if(*b*->data==*x*)

    {

        if(*b*->lchild!=NULL)

        output(*b*->lchild);

        if(*b*->rchild!=NULL)

        output(*b*->rchild);

        return;

    }

    child(*b*->lchild,*x*);

    child(*b*->rchild,*x*);

}

二十.bool compbtree(btnode \**b*)

{

    btnode \*qu[maxsize],\*p;

    int front=0,rear=0;

    bool cm=true;

    bool bj=true;

    if(*b*==NULL) return true;

    rear++;

    qu[rear]=*b*;

    while(front!=rear)

    {

        front=(front+1)%maxsize;

        p=qu[front];

        if(p->lchild==NULL)

        {

            bj=false;

            if(p->lchild!=NULL)

            cm=false;

        }

        else

        {

            if(!bj) cm=false;

            rear=(rear+1)%maxsize;

            qu[rear]=p->lchild;

            if(p->rchild==NULL)

            bj=false;

            else

            {

                rear=(rear+1)%maxsize;

                qu[rear]=p->rchild;

            }

        }

    }

    return cm;

}