2016年

2019年11月20日 2:38

数据结构部分

一.

1. D
2. C
3. C
4. D
5. A
6. D
7. C
8. A
9. C
10. C

二.

1.

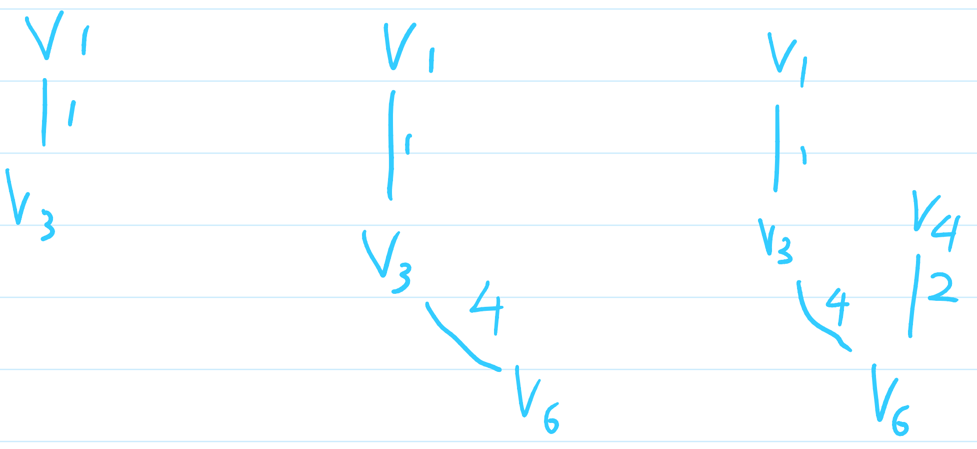
第一次:	18	29	25	47	22	58	10	51
第二次:	18	25	29	47	10	22	51	58
第三次:	10	18	22	25	29	47	51	58

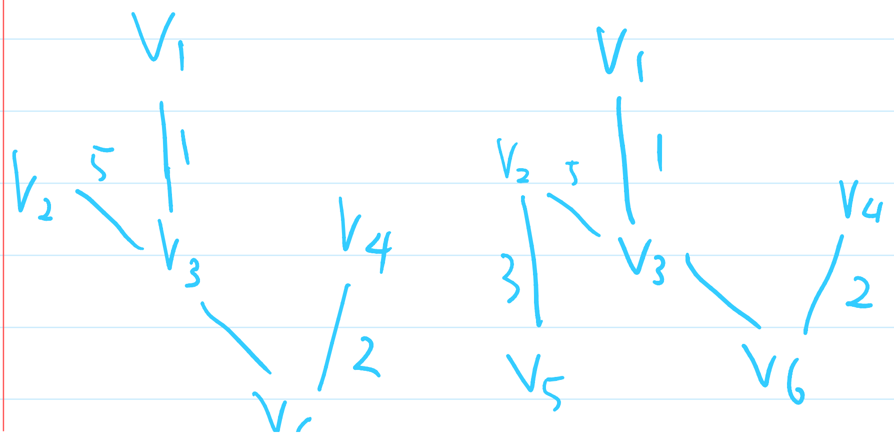
2.

$ASL=(3+2+3+1+3+2+3+4)/8=2.625$

3.

思想：先保证图连通，再选边权值最小的
初始状态选边权值最大的，构成回路就舍去





\ / 2 V_6 V_5 V_6

三、

1.

```
#include<iostream>
#include<queue>
#include<stdlib.h>
using namespace std;

struct BTnode
{
        int data;
        BTnode *left;
        BTnode *right;
};

void LevelOrder(BTnode *a)
{
        queue <BTnode*>q;
        int count=0;
        if(a)
        {
                q.push(a);
                while(!q.empty())
                {
                        BTnode *tem=q.front();
                        q.pop();
                        cout<<tem->data;
                        if(tem->left)
                                q.push(tem->left);
                        if(tem->right)
                                q.push(tem->right);
                }
        }
}

int main()
{
        /*
        BTnode *node6=(BTnode*)malloc(sizeof(BTnode));       node6->left=NULL;       node6->
        right=NULL;      node6->data=6;
        BTnode *node5=(BTnode*)malloc(sizeof(BTnode));       node5->left=NULL;       node5->
        right=NULL;      node5->data=5;
        BTnode *node4=(BTnode*)malloc(sizeof(BTnode));       node4->left=NULL;       node4->
        right=NULL;      node4->data=4;
        BTnode *node3=(BTnode*)malloc(sizeof(BTnode));       node3->left=NULL;       node3->
        right=node6;     node3->data=3;
        BTnode *node2=(BTnode*)malloc(sizeof(BTnode));       node2->left=node4;      node2->
        right=node5;     node2->data=2;
        BTnode *node1=(BTnode*)malloc(sizeof(BTnode));       node1->left=node2;      node1->
        right=node3;     node1->data=1;
        */
        /*
        BTnode *node5=(BTnode*)malloc(sizeof(BTnode));       node5->left=NULL;       node5->
        right=NULL;      node5->data=5;
        BTnode *node4=(BTnode*)malloc(sizeof(BTnode));       node4->left=NULL;       node4->
        right=NULL;      node4->data=4;
        BTnode *node3=(BTnode*)malloc(sizeof(BTnode));       node3->left=NULL;       node3->
        right=NULL;      node3->data=3;
        BTnode *node2=(BTnode*)malloc(sizeof(BTnode));       node2->left=node4;      node2->
        right=node5;     node2->data=2;
        BTnode *node1=(BTnode*)malloc(sizeof(BTnode));       node1->left=node2;      node1->
        right=node3;     node1->data=1;
        */
        BTnode *node7=(BTnode*)malloc(sizeof(BTnode));       node7->left=NULL;       node7->
        right=NULL;      node7->data=7;
        BTnode *node6=(BTnode*)malloc(sizeof(BTnode));       node6->left=NULL;       node6->
        right=NULL;      node6->data=6;
        BTnode *node5=(BTnode*)malloc(sizeof(BTnode));       node5->left=NULL;       node5->
        right=NULL;      node5->data=5;
        BTnode *node4=(BTnode*)malloc(sizeof(BTnode));       node4->left=node6;      node4->
        right=node7;     node4->data=4;
        BTnode *node3=(BTnode*)malloc(sizeof(BTnode));       node3->left=NULL;       node3->
        right=NULL;      node3->data=3;
        BTnode *node2=(BTnode*)malloc(sizeof(BTnode));       node2->left=node4;      node2->
        right=node5;     node2->data=2;
        BTnode *node1=(BTnode*)malloc(sizeof(BTnode));       node1->left=node2;      node1->
        right=node3;     node1->data=1;

        cout<<LevelOrder(node1);
}
```

2.

```
#include<iostream>
#include<stdlib.h>
using namespace std;

struct dnode
{
        int data;
        int feq;
        dnode *next;
        dnode *prev;
};

dnode* Create()
{
        int tem;
        dnode *head=(dnode*)malloc(sizeof(dnode));
        dnode *temNode1=head;
        dnode *temNode2,*temNode3;
        while(cin>>tem,tem!=-1)//输入-1结束
        {

                temNode1->data=tem;
                temNode1->feq=0;
                temNode2=(dnode*)malloc(sizeof(dnode));
                temNode3=temNode1;
                temNode1->next=temNode2;
                temNode2->prev=temNode1;
                temNode1=temNode2;
        }
        temNode3->next=head;
```

```
		head->prev=temNode3;
		return head;
}

void PrintList(dnode *a)
{
		dnode *head=a;
		while(a)
		{
				cout<<a->data<<" ";
				a=a->next;
				if(a==head)
						break;
		}
		cout<<endl;
}

dnode* FeqSort(dnode *A)
{
		dnode *head=A;
		dnode *dnodes[1000];
		int n=0;
		while(A)
		{
				dnodes[n++]=A;
				A=A->next;
				if(A==head)
						break;
		}

		for(int i=0;i!=n-1;i++)
		{
				int max=i;
				for(int j=1;j!=n;j++)
				{
						if(dnodes[max]->feq<dnodes[j]->feq)
								max=j;
				}
				dnode *tem=dnodes[i];
				dnodes[i]=dnodes[max];
				dnodes[max]=tem;
		}

		head=dnodes[0];
		for(int i=0;i!=n-1;i++)
		{
				dnodes[i]->next=dnodes[i+1];
				dnodes[i+1]->prev=dnodes[i];
		}
		head->prev=dnodes[n-1];
		dnodes[n-1]->next=head;

		return head;
}

int main()
{
		dnode *A=Create();
		PrintList(A);
		A=FeqSort(A);
		PrintList(A);
}
```

数据库部分

一、

1. B
2. D
3. C
4. C
5. A
6. A
7. D
8. B
9. C
10. C

二、

1. (1) $\pi_{SPECIALITY}(\sigma_{ENAME='杨洋'}(E))$

(2) $\pi_{E\#,ENAME}(\sigma_{SEX='女' \wedge CITY='上海}(E \bowtie W \bowtie D))$

2. (1) Select E#, ENAME from E where

SPECIALITY=' [illegible]

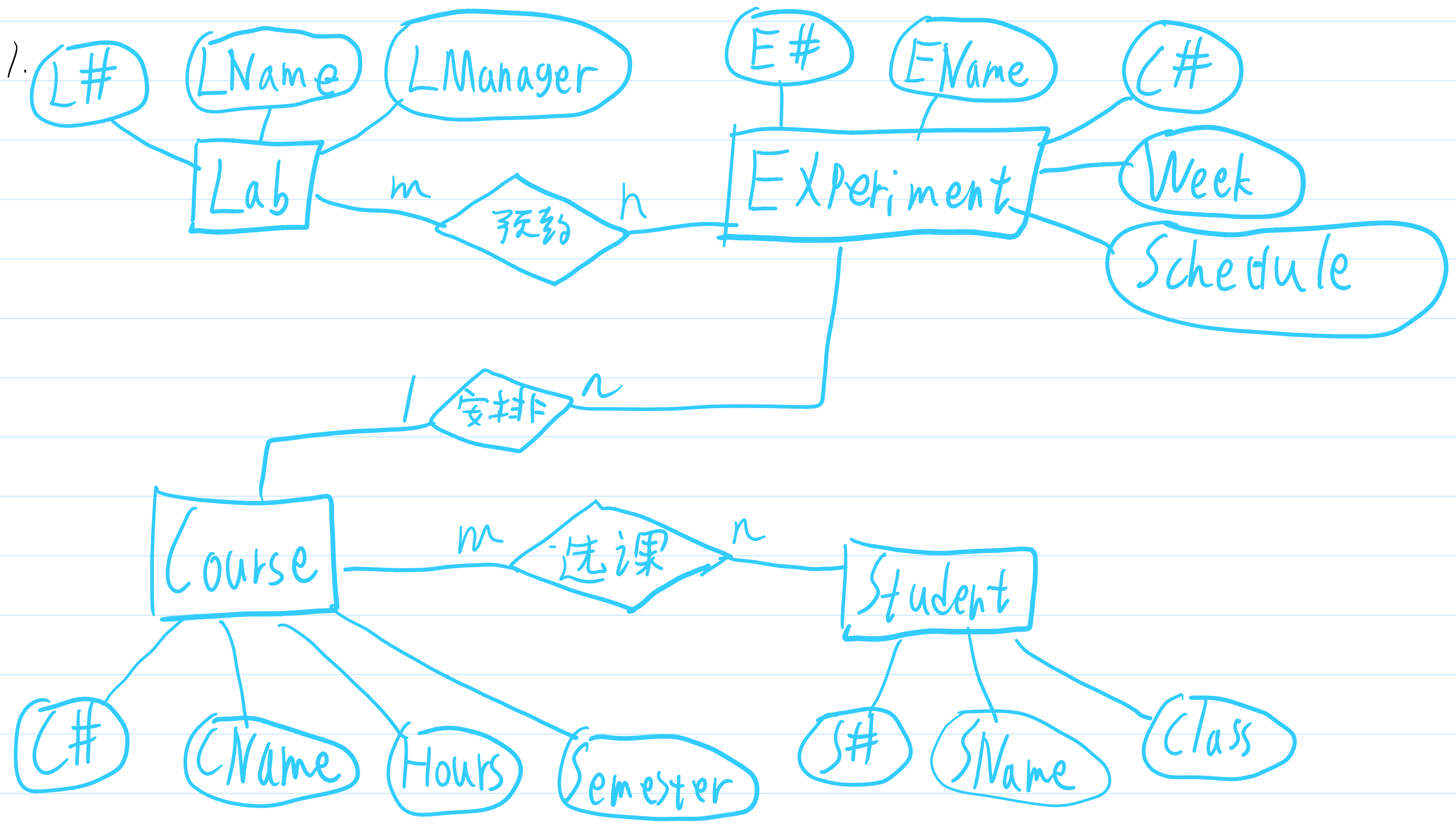
[illegible]

SPECIALITY='软件工程'

(2) Select E#, ENAME, SALARY, CITY from E, D, W where E.E#=W.E# And W.D#=D.D# And E.ENAME like '杨%'

三、

1\.



2\.

学生($\underline{\text{S\#}}$ SName Class)

课程($\underline{\text{C\#}}$ CName Hours Semester)

实验($\underline{\text{E\#}}$, EName , C#, Week, Schedule, $\underline{\text{C\#}}$)

实验室($\underline{\text{L\#}}$, LName, LManager)

选课($\underline{\underline{\text{S\#}}\quad\underline{\text{C\#}}}$)

预约($\underline{\underline{\text{L\#}}\text{ , }\underline{\text{E\#}}}$)