选择题

（13）选C

V0~V1~V3 ~V4~V2的过程，下一个遍历的顶点均是前一顶点列出的相邻顶点中最接近的一个；但V2的相邻顶点中没有未遍历过的顶点，故回溯到V4，V5是V4的下一个未遍历相邻顶点，最后来到V6。故深度优先遍历的结果为：0，1，3，4，2，5，6

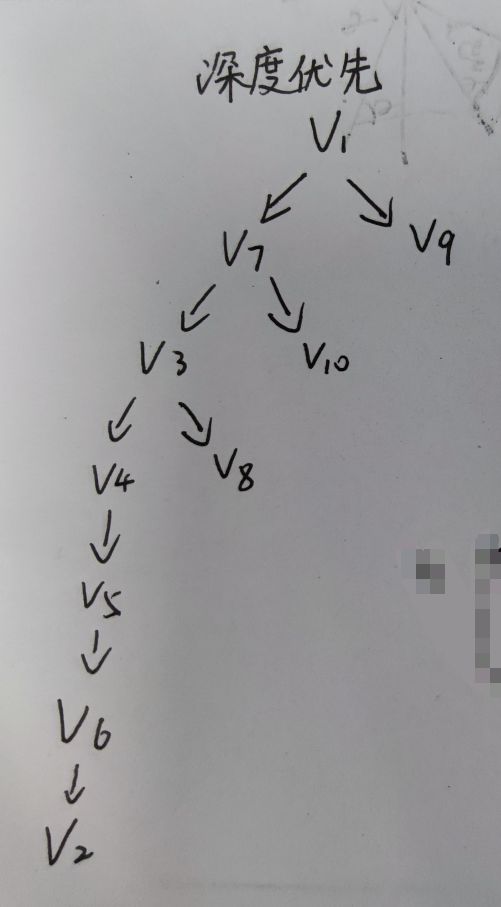
（14）选D和D

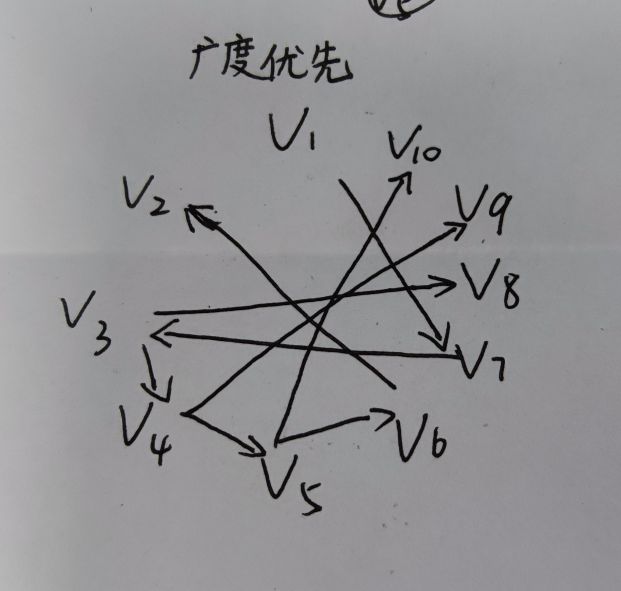
广度优先遍历时，V1,V2,V3均是V0的相邻顶点，因此依次被遍历到，答案为0,1,2,3

深度优先遍历时，V0~V1~V2的过程中，下一个遍历的顶点均是前一顶点列出的相邻顶点中（除V0以外）最接近的一个，而V2的相邻顶点中V0和V1都已遍历，于是遍历V3。

答案为0，1，2，3.

应用题





算法题

void DFS(ALGraph G, int v,int m)//从第v个顶点开始非递归实现深度优先遍历

{

int visited[m] = { 0 };

InitStack(&S);

Push(S, v);//将顶点v进栈

while (S!=NULL)

{

Pop(S, k);//栈顶元素k出栈

if (!visited[k])

{

Cout<<k;

visited[k] = 1;

p = G.vertices[k].firstarc;

while (p != NULL)

{

w = p->adjvex;

if (!visited[w])

Push(S, w);

p = p->nextarc;

}

}

}

}

二：

constexpr int MaxVertexNum = 30; //最大顶点数

typedef char VertexType; //顶点数据类型为字符型，s和t是简单路径的起点和

//终点，假定任意两个顶点的数据都不同

typedef int EdgeType; //1：表示有边存在；0：表示两个顶点间无边

typedef struct

{

VertexType vexs[MaxVertexNum]; //顶点表

EdgeType edges[MaxVertexNum][MaxVertexNum];//邻接矩阵

int n，e; //图中当前的顶点数和边数

} MGragh;

int findvex(VertexType s, VertexType vexs[MaxVertexNum])//找到s和t分别对应的序号

{

int k;

for (int i = 0;i < MaxVertexNum;i++)

{

if (vexs[i] = s)

k = i;

}

printf("%d", k);

}

int select(edge[][],S, i, j)

{

int k = 1, a = 1;

while (k <= i - 1)

{

if (S.data[k]==j)

{

return 0;

a = 0;

}

if (a == 0)

break;

else

k++;

}

return 1;

}

int place(S,edge[][], vexs[], visited[])

{

while (!StackEmpty(S))

{

int count = 0;

if (vexs[S.top]==t)

{

printf( "第%d条路径为：", ++count);

printf("%c", s);

for (j = 2;j <= S.top;j++)

printf("->%c",vexs[S.data[j]]);

FLAGG = 0;

}

int flag = 0;

for (int j = 1;j <= n;j++)

if (edge[S.data[S.top]][j] == 1 & select(edge[], S, S.top + 1,j))

{

S.top++;

S.data[S.top] = j;

visited[j] = 1;

flag = 1;

break;

}

if (flag == 0)

{

while (S.top > 0)

{

if (S.data[S.top] == MaxVertexNum)

{

visited[S.data[S.top]] = 0;

S.top--;

}

for (j = S.data[S.top] + 1;j <= MaxVertexNum;j++)

{

if (edge[S.data[S.top]][j] == 1 & select(edge[], S, S.top + 1;j)&visited[j]=0)

{

S.data[S.top] = j;

break;

}

if (j > MaxVertexNum)

S.top--;

else

break;

}

}

}

}

if (FLAGG) printf(out, "No Result meets needs\n");

}

void searchpath(MGraph& G, VertexType s, VertexType t)

{

Status CreateUDN(MGraph & G);//初始化一个邻接矩阵

int k1 = findvex(s), k2 = findvex(t);

int visited[MaxVertexNum] = { 0 };

InitStack(&S);

Push(S, s);//s入栈

place(S, edge[][], vexs[], visited[]);

}