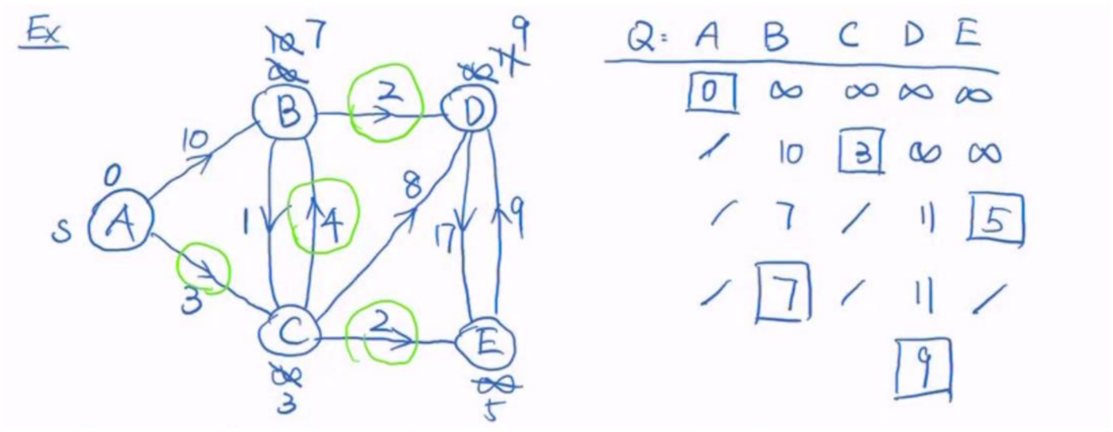
HW5_YIXIAO CHEN_002198256

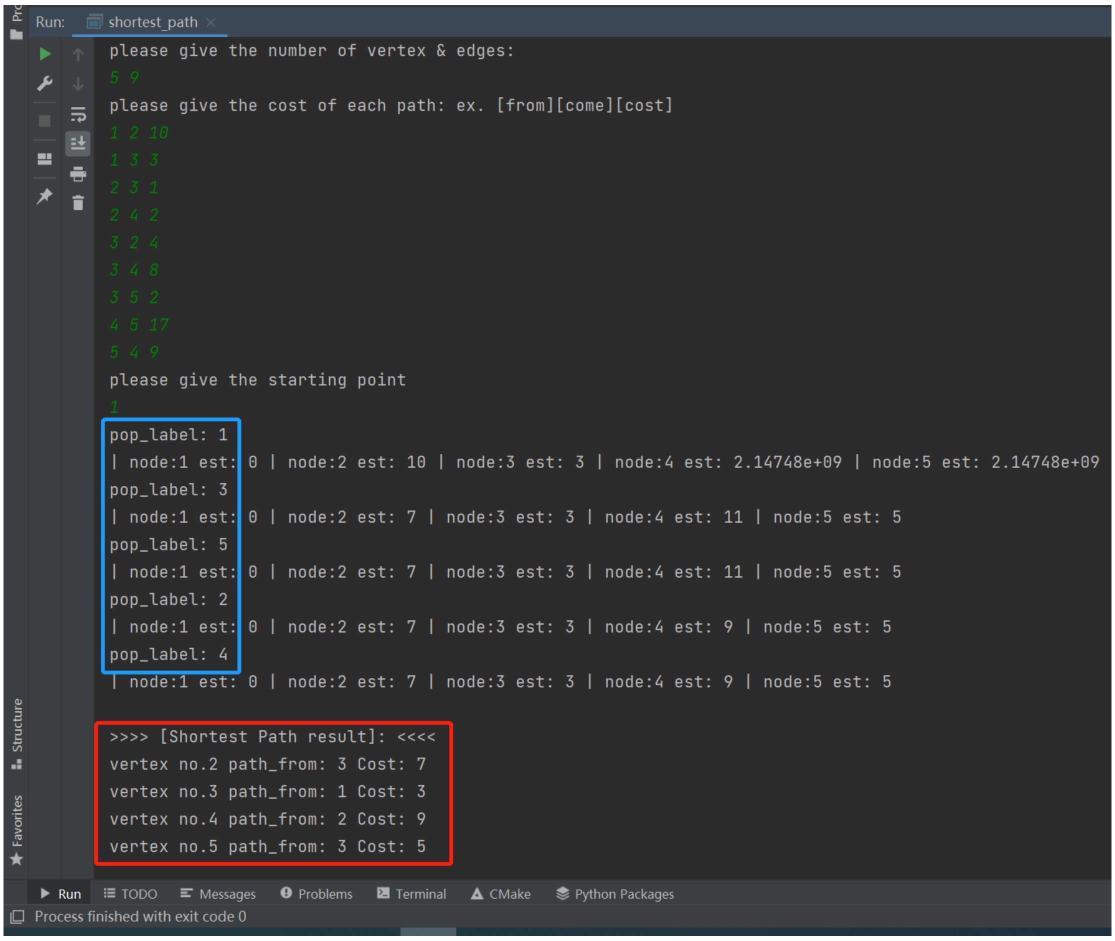
Question:

Question 1 (3 pt.) Write codes for Dijkstra's algorithm using unsorted array for priority Q.



------**Codes** after result for every question below------

[Dijkstra]:



```
please give the number of vertex & edges:
5 9
please give the cost of each path: ex. [from][come][cost]
1 2 10
1 3 3
2 3 1
2 4 2
3 2 4
3 4 8
3 5 2
4 5 17
5 4 9
please give the starting point
1
pop_label: 1
| node:1 est: 0 | node:2 est: 10 | node:3 est: 3 | node:4 est: 2.14748e+09 | node:5 est: 2.14748e+09
pop_label: 3
| node:1 est: 0 | node:2 est: 7 | node:3 est: 3 | node:4 est: 11 | node:5 est: 5
pop_label: 5
| node:1 est: 0 | node:2 est: 7 | node:3 est: 3 | node:4 est: 11 | node:5 est: 5
pop_label: 2
| node:1 est: 0 | node:2 est: 7 | node:3 est: 3 | node:4 est: 9 | node:5 est: 5
pop_label: 4
| node:1 est: 0 | node:2 est: 7 | node:3 est: 3 | node:4 est: 9 | node:5 est: 5

>>>> [Shortest Path result]: <<<<
vertex no.2 path_from: 3 Cost: 7
vertex no.3 path_from: 1 Cost: 3
vertex no.4 path_from: 2 Cost: 9
vertex no.5 path_from: 3 Cost: 5
```

Process finished with exit code 0

【Dijkstra】_by cyx

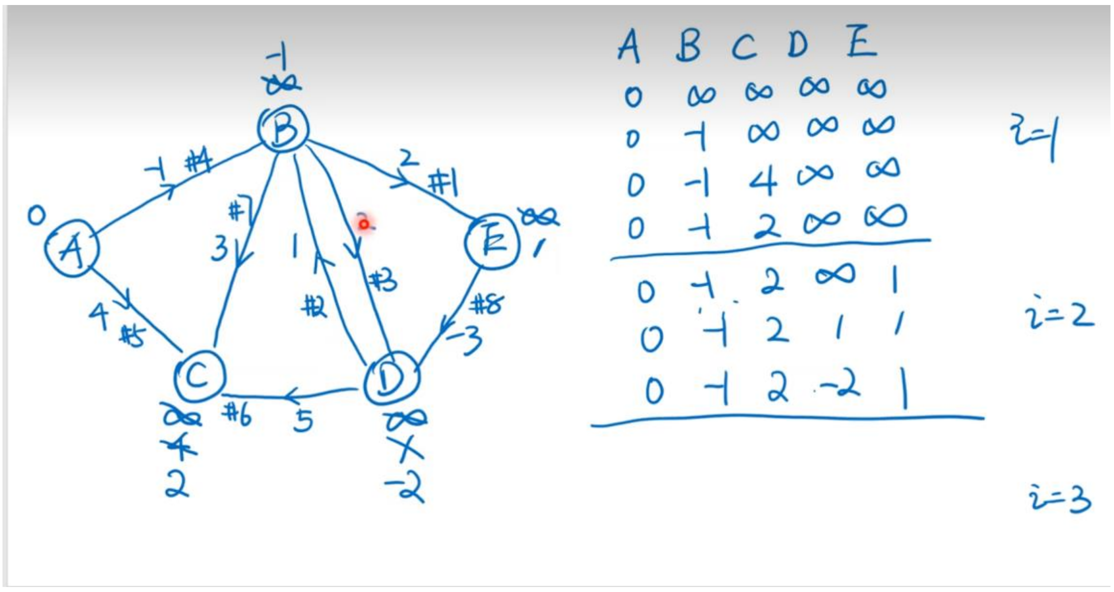
```
#include <iostream>
#include <vector>
#include <cmath>
//#include<queue>
#include <algorithm>
#include <cstring>

using namespace std;

struct Vertex {
	int label = 0;//后续打乱顺序后再输出可能会用到编号
	double estimate = INT_MAX;
	//vector<Vertex>vertex_come;
	//vector<Vertex>vertex_from;
	vector<Vertex *> vertex_come;
	vector<Vertex *> vertex_from;
	Vertex *parent = NULL;
};

/*struct cmp {
	bool operator()(Vertex *a, Vertex *b) {
		return a->estimate > b->estimate;
	}
};*/

bool cmp(Vertex *a, Vertex *b){
	return a->estimate > b->estimate;
}

int main() {
	cout << ">>>> [Dijkstra's algorithm] <<<<" << endl;

	int vertex_number, edge_number;
	cout << "please give the number of vertex & edges:" << endl;
	cin >> vertex_number >> edge_number;
	int consumption[vertex_number + 1][vertex_number + 1];
	memset(consumption, 0, sizeof(consumption));
```

```
Vertex vertex_array[vertex_number + 1];

int from, come, cost;
cout << "please give the cost of each path: ex. [from][come][cost]" << endl;
for (int i = 1; i <= edge_number; i++) {
	cin >> from >> come >> cost;
	//vertex_array[come].vertex_from.push_back(vertex_array[from]);
	//vertex_array[from].vertex_come.push_back(vertex_array[come]);
	vertex_array[from].vertex_come.push_back(&vertex_array[come]);
	vertex_array[come].vertex_from.push_back(&vertex_array[from]);
	consumption[from][come] = cost;
}

//consumption 测试代码
/*for(int i = 1;i<=vertex_number;i++){
	for(int j =1; j<=vertex_number;j++){
		cout << consumption[i][j]<<" ";
	}
	cout <<endl;
}*/

int start_point;
cout << "please give the starting point" << endl;
cin >> start_point;
//start_point = 1;
vertex_array[start_point].label = start_point;
vertex_array[start_point].estimate = 0;
vertex_array[start_point].parent = NULL;
//Vertex *source = &vertex_array[start_point];

//priority_queue<Vertex *, vector<Vertex *>, cmp> Q;
vector<Vertex *>vertex_queue;
for (int i = 1; i <= vertex_number; i++) {
	vertex_array[i].label = i;
	//Q.push(&vertex_array[i]);
	vertex_queue.push_back(&vertex_array[i]);
}
```

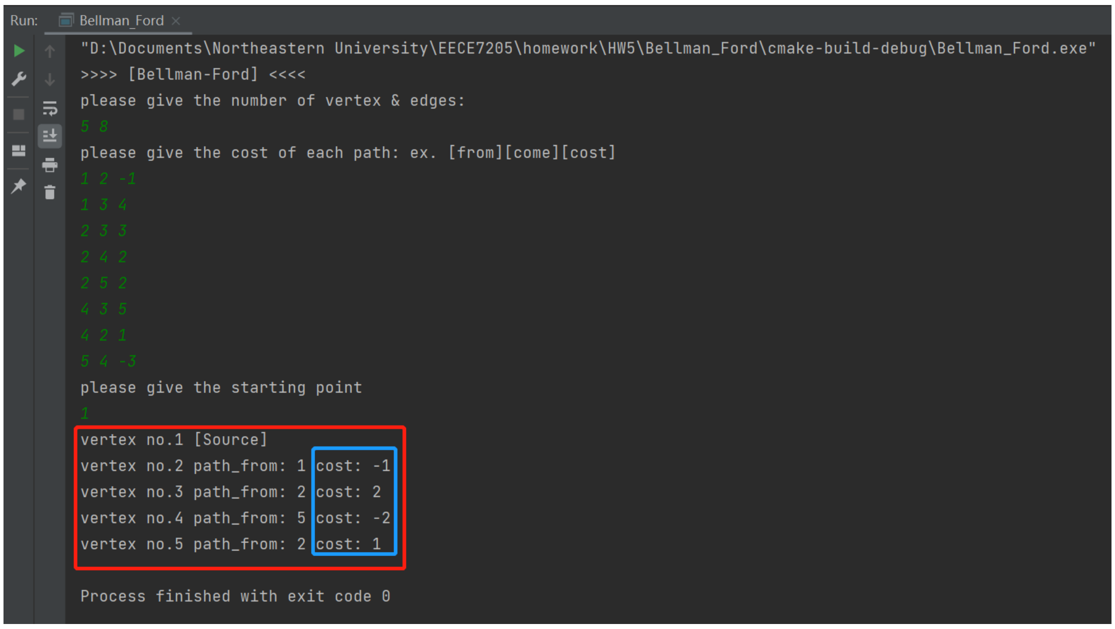
```
sort(vertex_queue.begin(),vertex_queue.end(),cmp);
//cout<<"top_label: "<<Q.top()->label<<endl;//测试优先级队列
//节点间关系测试函数
/*for (int i = 1; i <= vertex_number; i++) {
    cout << "vertex:" << vertex_array[i].label << " vertex_come: ";
    for (int j = 0; j < vertex_array[i].vertex_come.size(); j++) {
        cout << vertex_array[i].vertex_come[j]->label << " ";
    }
    cout << endl;
}*/

vector<Vertex *>q_operate;
Vertex *tar;
while (vertex_queue.size()!=0){
//while (Q.empty() != 1) {
    //tar = Q.top();
    //pop 测试函数
    tar = vertex_queue[vertex_queue.size()-1];
    //cout << "pop_label: "<<Q.top()->label<<endl;
    cout << "pop_label: "<<vertex_queue[vertex_queue.size()-1]->label<<endl;
    //Q.pop();
    for (int i = 0; i < tar->vertex_come.size(); i++) {
        if (tar->vertex_come[i]->estimate > tar->estimate + 
consumption[tar->label][tar->vertex_come[i]->label]) {
            tar->vertex_come[i]->estimate = tar->estimate + 
consumption[tar->label][tar->vertex_come[i]->label];
            tar->vertex_come[i]->parent = tar;
        }
    }
    //测试函数
    for(int j = 1;j<=vertex_number;j++){
        cout<<"| node:"<<j<<" est: "<<vertex_array[j].estimate<<" ";
    }
    cout<< endl;
    vertex_queue.pop_back();
    sort(vertex_queue.begin(),vertex_queue.end(),cmp);
    //cyx 独家笨方法
    /*for(int i = 0;i<Q.size();i++){
        q_operate.push_back(Q.top());
        Q.pop();
    }
    for(int i = 0;i<q_operate.size();i++){
        Q.push(q_operate[i]);
    }
    q_operate.clear();*/
}

cout << "\n>>>> [Shortest Path result]: <<<<" << endl;
for (int i = 1; i <= vertex_number; i++) {
    if (i == start_point) {
        continue;
    }
    cout << "vertex no." << i << " path_from: " << vertex_array[i].parent->label
<<" Cost: "<<vertex_array[i].estimate <<endl;
}

return 0;
}
//程序输入样例
/*5 9
1 2 10
1 3 3
2 3 1
2 4 2
3 2 4
3 4 8
3 5 2
4 5 17
5 4 9
1*/
```

Question:

Question 2 (2 pt.) Write codes for Bellman-Ford algorithm.



[Bellman-ford]:



```
"D:\Documents\Northeastern University\EECE7205\homework\HW5\Bellman_Ford\cmake-build-debug\Bellman_Ford.exe"
>>>> [Bellman-Ford] <<<<
please give the number of vertex & edges:
5 8
please give the cost of each path: ex. [from][come][cost]
1 2 -1
1 3 4
2 3 3
2 4 2
2 5 2
4 3 5
4 2 1
5 4 -3
please give the starting point
1
vertex no.1 [Source]
vertex no.2 path_from: 1 cost: -1
vertex no.3 path_from: 2 cost: 2
vertex no.4 path_from: 5 cost: -2
vertex no.5 path_from: 2 cost: 1

Process finished with exit code 0
```

【Bellmen-ford】_by cyx

```
#include <iostream>
#include <cstring>

using namespace std;

struct Vertex{
	int label;
	int estimation = INT_MAX;
	int path_label = 0;
};
struct Edge{
	Vertex *edge_from;
	Vertex *edge_come;
};

int main() {
	cout << ">>>> [Bellman-Ford] <<<<" << endl;

	int vertex_number, edge_number;
	cout << "please give the number of vertex & edges:" << endl;
	cin >> vertex_number >> edge_number;
	int consumption[vertex_number + 1][vertex_number + 1];
	memset(consumption, 0, sizeof(consumption));
	Vertex vertex_array[vertex_number + 1];
	for(int i=1;i<=vertex_number;i++){
		vertex_array[i].label = i;
	}
	Edge edge_array[edge_number+1];

	int from, come, cost;
	cout << "please give the cost of each path: ex. [from][come][cost]" << endl;
	for (int i = 1; i <= edge_number; i++) {
		cin >> from >> come >> cost;
		edge_array[i].edge_come = &vertex_array[come];
		edge_array[i].edge_from = &vertex_array[from];
		consumption[from][come] = cost;
	}
```

```
    int start_point;
    cout << "please give the starting point" << endl;
    cin >> start_point;
    //start_point = 1;
    vertex_array[start_point].estimation = 0;

    for (int i = 1; i <= vertex_number - 1; i++) {
        for (int j = 1; j <= edge_number; j++) {
            if          (edge_array[j].edge_come->estimation           >
edge_array[j].edge_from->estimation +

consumption[edge_array[j].edge_from->label][edge_array[j].edge_come->label]) {
                edge_array[j].edge_come->estimation                    =
edge_array[j].edge_from->estimation +

consumption[edge_array[j].edge_from->label][edge_array[j].edge_come->label];
                edge_array[j].edge_come->path_label                    =
edge_array[j].edge_from->label;
            }
        }
    }

    for(int i =1;i<=edge_number;i++){
        if(edge_array[i].edge_come->estimation                         >
edge_array[i].edge_from->estimation +

consumption[edge_array[i].edge_from->label][edge_array[i].edge_come->label]){
            cout <<"Sorry, there seems to be a negative loop in the graph, please
check. END"<<endl;
            return 0;
        }
    }

    for(int i=1;i<=vertex_number;i++){
        cout<< "vertex no." <<i<<" ";
        if(i == start_point){
            cout << "[Source]"<<endl;
        }
        else{
            cout <<"path_from: " << vertex_array[i].path_label<<" ";
            cout <<"cost: "<< vertex_array[i].estimation<<endl;
        }
    }

    return 0;
}
```