***النظرى فى الكشكول***

***1)code for (represent of matrix) (bool&wight) in bage ();***

**2)*code for (represent of list) (bool&wight) in bage ();***

**3)*tricks;***

**4)*code for (Third type(Edges List) of prisentathion( in bage ();***

***1)matrix representation for directed graph (bool or wighit);***

const int N = 100; // adjacency matrix for bool

bool adjMatrixBool[N][N];

// Assume all input is ZERO based. Node values [0 - n-1]

/\*

3

0 1 0

1 1 0

0 1 1

\*/

cin>>n;

lp(i, n) lp(j, n) { //notation

int b;

cin>>b;

adjMatrixBool[i][j] = b;

}

-----------------------------------------------------------------------------------------------------

int adjMatrix[N][N]; // adjacency matrix for wight

/\*

3

0 10 5

2 7 9

3 2 0

\*/

cin>>n;

lp(i, n) lp(j, n) {

int b;

cin>>b;

adjMatrix[i][j] = b;

}

--------------------------------------------------------------------------------------

//adjMatrix initialize with OO

int edges;

cin>>n>>edges;

/\*

3 5

0 3 10

1 5 2

3 7 -2

0 3 2

2 4 6

\*/

lp(i, edges)

{

int from, to, cost;

cin>>from>>to>>cost; // min to take the smallest wight

adjMatrix[from][to] = min(adjMatrix[from][to], cost); // if directed

// if undirected: add following

adjMatrix[to][from] = adjMatrix[from][to];}

-----------------------------------------------------------------------------------------

***Second type list:***

vector< vector<int> > adjList1; // boolean relation

// Let's keep edge with min cost;

/\*

3

2 1 2

1 2

2 0 1

\*/

cin>>n;

adjList1 = vector< vector<int> >(n);

lp(i, n)

{

int cnt;

cin>>cnt;

lp(j, cnt)

{

int to;

cin>>to;

adjList1[i].push\_back(to);

}

}

vector< vector< pair<int, int> > > adjList2;// cost relation (node, cost): Select 1 of multiple edges

/\*

3

2 1 13 2 4

1 2 9 3 -4

2 0 -7 1 8

\*/

cin>>n;

adjList2 = vector< vector< pair<int, int> > >(n);

lp(i, n)

{

int cnt;

cin>>cnt;

lp(j, cnt)

{

int to, cost;

cin>>to>>cost;

adjList2[i].push\_back( make\_pair(to, cost));

}

}

---------------------------------------------------------------------------------------

cin>>n>>e; //n🡺node , e🡺edages

adjList2 = vector< vector< int> >(n);

lp(i, e)

{int from, to;

cin>>to>>from;

adjList2[from].push\_back( to);

}

**Trick:**

vector<int> adjMatrixAll[N][N]; // Keep all edges for a given pair of nodes

map< pair<int, int>, int> adjMatrixMap; // Imagine graph of 100000 node, but total number of edges is 50000!

-------------------------------------------------------------------------

***Third type(Edges List) of prisentathion:***

*// جطة فوق المين*

// Edges List

struct edge {

int from, to, w;

edge(int from, int to, int w): from(from), to(to), w(w) {}

bool operator < (const edge & e) const {

return w > e.w; // Sort with smaller weight first

}

};

vector<edge> edgeList; // الى هيرجع