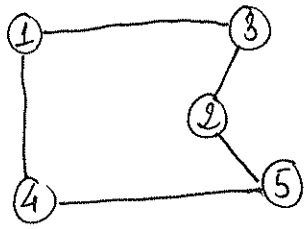
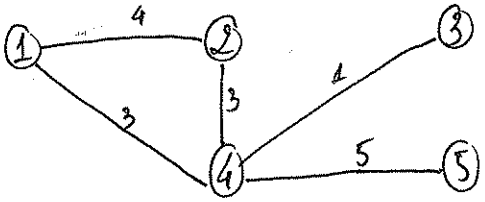
Đề 1:

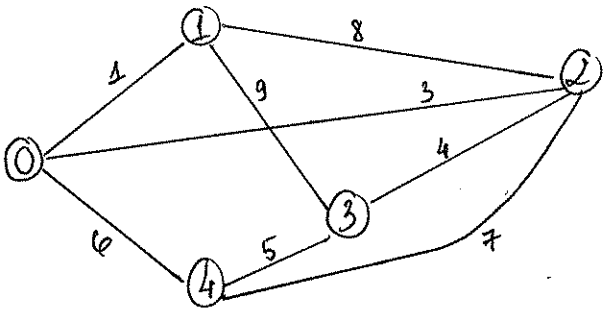
Câu 1: Tìm chu trình Euler của đồ thị sau:



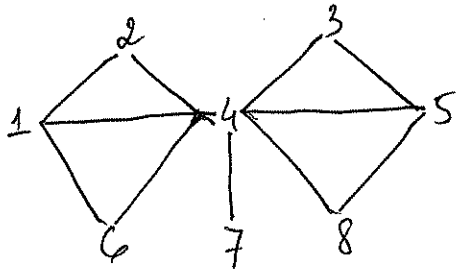
Câu 2: Sử dụng thuật toán Prim, tìm cây khung nhỏ nhất trong đồ thị sau



Câu 3: Sử dụng thuật toán Dijkstra, tìm đường đi ngắn nhất từ đỉnh A đến các đỉnh còn lại trong đồ thị sau:



Câu 4: Sử dụng thuật toán DFS để duyệt đồ thị sau, bắt đầu từ đỉnh ②



- Đề 2:

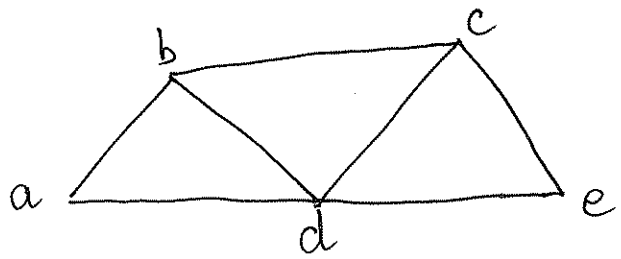
Câu 1: Có 5 việc cần tuyển 5 ứng viên đảm nhiệm, mỗi ứng viên 1 việc.

Gọi S_y là tập hợp các ứng viên thích hợp cho việc thứ y và giả sử

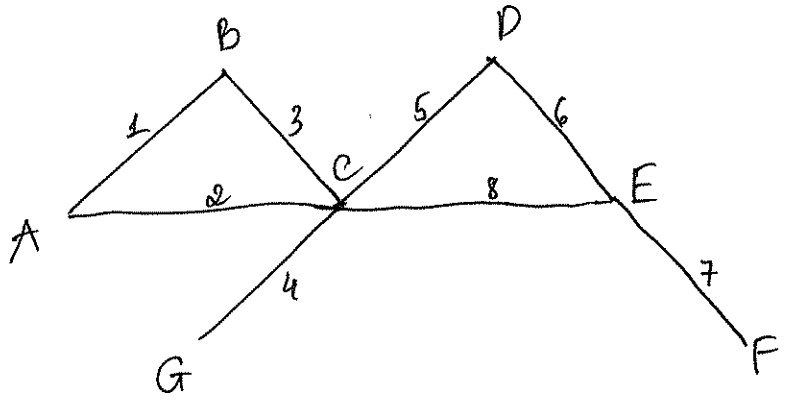
	A	B	C	D	E
S_1	X	X	X		
S_2				X	X
S_3				X	
S_4					X
S_5	X				X

Có thể tuyển đủ 5 ứng viên cho 5 công việc trên hay không?

Câu 2: Sử dụng BFS duyệt đồ thị sau:



Câu 3: Tìm cây khung nhỏ nhất trong đồ thị sau, dùng thuật toán Kruskal.

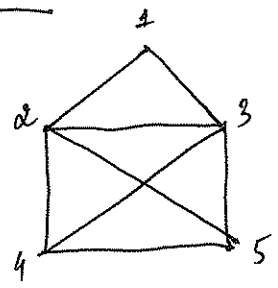


Câu 4: Sử dụng định lý hôn nhân (Hall), giải quyết bài toán sau. Có 4 chàng trai (B1, B2, B3, B4) và 5 cô gái (G1, G2, G3, G4, G5), mỗi chàng trai có 1 dsách các cô gái như sau:

	G1	G2	G3	G4	G5
B1	X			X	X
B2	X				
B3		X	X	X	
B4		X		X	

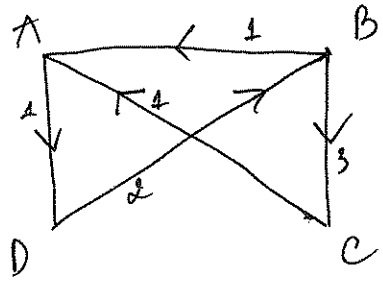
Đề 3:

Câu 1: Tìm chu trình hoặc đường đi Euler trong đồ thị sau

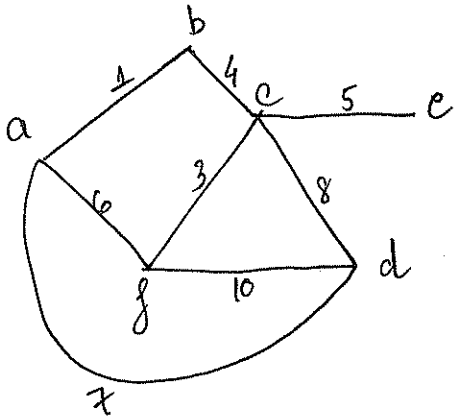


4 → 5 → 2 → 3 → 4 → 2 → 1 → 3 → 5

Câu 2: Sử dụng Flaysd tìm đường đi ngắn nhất giữa tất cả các đỉnh trong đồ thị sau:



Câu 3: Cho đồ thị sau:



1. Sử dụng Prim tìm cây khung nhỏ nhất
2. Bỏ qua thông tin về các trọng số. Chọn a là đỉnh bđ. Khi đó cây khung trở thành cây nhị phân tìm kiếm. Hãy duyệt cây theo các cách sau tiền tố - trung tố - hậu tố

Câu 4: Viết hàm in ra các đỉnh có bậc lẻ.

Giả sử rằng có khai báo đồ thị sau:

```
#define MAX_ARR 10
struct GRAPH
{
    int n;
    int a[MAX_ARR][MAX_ARR];
};
```

Giải

Câu 4:

```
void InRaCacDinhBacLe (GRAPH g)
{
    for (int i=0; i<g.n; i++) { int dem=0;
        for (int j=0; j<g.n; j++)
        {
            if (g.a[i][j] == 1)
                dem++;
        }
        if (dem % 2 != 0)
            printf ("%d \t", i+1);
    }
}
```

HUTECH
Đại học Công nghệ Tp.HCM

KHOA CÔNG NGHỆ THÔNG TIN

ĐỀ THI HỌC KỲ..... LẦN ....NĂM HỌC 2016-2017

Ngành/Lớp :.....

Môn thi : **Lý thuyết đồ thị**.....

Mã môn học : COS211...Số ĐVHT/TC: 3.....

Ngày thi : .....19/4/2017.....

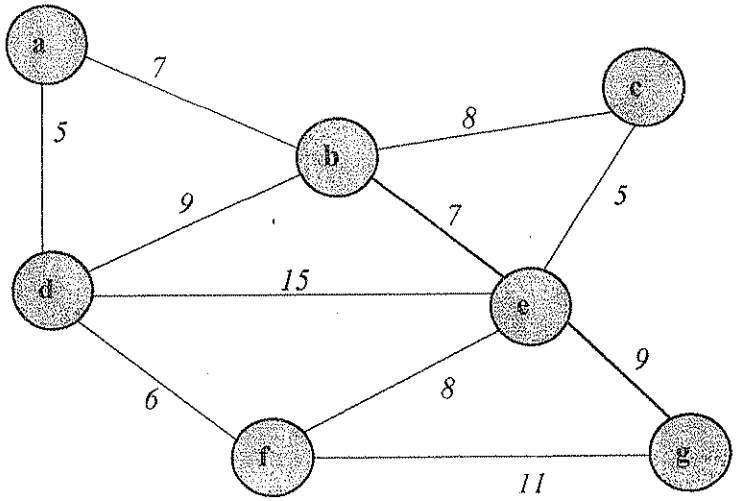
Thời gian làm bài: **90 phút**.....

Mã đề : 01.....

SỬ DỤNG TÀI LIỆU: CÓ ☐ KHÔNG ☑

CÂU 1. (3.0 điểm)

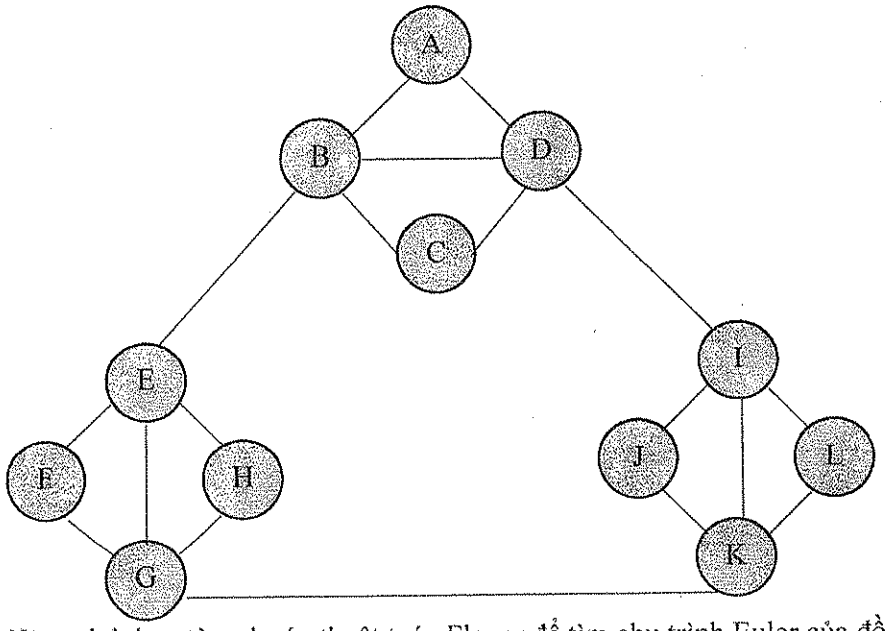
Cho đồ thị G=<V,E> bao gồm các đỉnh và các cạnh với trọng số (trọng lượng) như sau:



Hãy minh họa từng bước thuật toán Prim để tìm cây khung ngắn nhất của đồ thị nêu trên.

CÂU 2. (3.0 điểm)

Cho đồ thị gồm 12 đỉnh như sau:



Hãy minh họa từng bước thuật toán Fleury để tìm chu trình Euler của đồ thị nêu trên.

CÂU 3. (4.0 điểm)

Cho đồ thị có trọng số G = <V, E>, trong tập các đỉnh V = {V_1, V_2, V_3, V_4, V_5, V_6} xác định bởi ma trận trọng số sau:

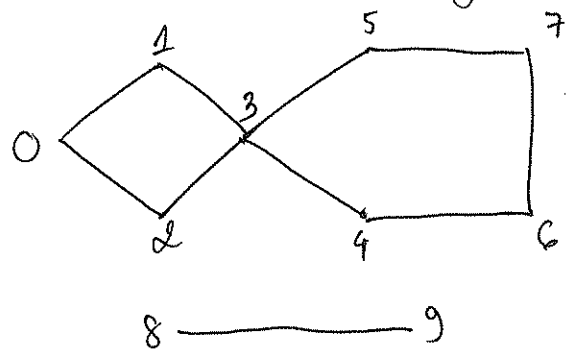
D =

	V_1	V_2	V_3	V_4	V_5	V_6
V_1	0	5	13	1	∞	∞
V_2	6	0	8	∞	∞	16
V_3	∞	7	0	4	8	∞
V_4	∞	3	∞	0	∞	∞
V_5	∞	6	4	∞	0	7
V_6	∞	∞	2	9	1	0

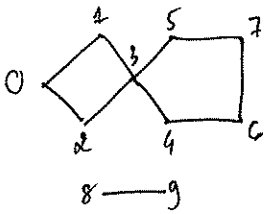
Dùng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh V_1 đến các đỉnh V_2, V_3, V_4, V_5, V_6. Yêu cầu viết rõ kết quả trung gian trong từng bước.

Cán bộ coi thi không được giải thích gì thêm.

Đề 1: Liên Thông



Bước 1: Khởi tạo

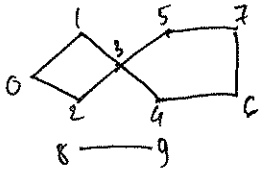


	0	1	2	3	4	5	6	7	8	9
Nhan	0	0	0	0	0	0	0	0	0	0

STPLT = 0

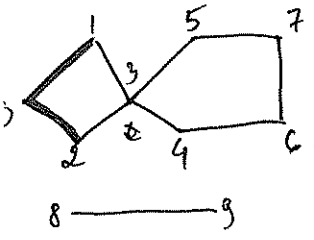
Bước 2: Chọn đỉnh 0. STPLT = 1

Nhan[0] = STPLT = 1



	0	1	2	3	4	5	6	7	8	9
Nhan	1	0	0	0	0	0	0	0	0	0

Bước 3:



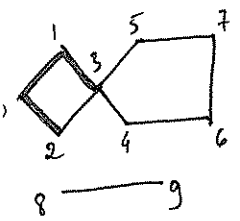
- Xét đỉnh kề với đỉnh 0, ta có đỉnh 1, 2 kề với 0
- Nhan[1] = Nhan[2] ≠ Nhan[0]

=> Nhan[1] = Nhan[2] = 1

- STPLT = 1

	0	1	2	3	4	5	6	7	8	9
Nhan	1	1	1	0	0	0	0	0	0	0

Bước 4:

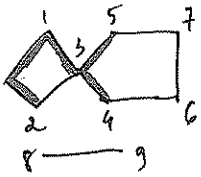


Chọn đỉnh 1. Xét đỉnh kề với đỉnh 1. Ta có đỉnh 3 kề với đỉnh 1.

Nhan[3] ≠ Nhan[1] => Nhan[3] = 1

	0	1	2	3	4	5	6	7	8	9
Nhan	1	1	1	1	0	0	0	0	0	0

Bước 5:



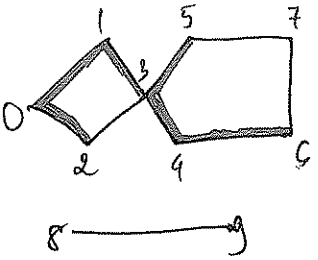
Chọn đỉnh 3. Xét đỉnh kề với đỉnh 3. Ta có đỉnh 5, 4, 2 kề với 3.

- Nhan[2] = STPLT = 1
- Nhan[4] = Nhan[5] ≠ Nhan[3]

=> Nhan[4] = Nhan[5] = 1

	0	1	2	3	4	5	6	7	8	9
Nhan	1	1	1	1	1	1	0	0	0	0

Bước 6:



Xét đỉnh 4. Ta có đỉnh 6 kề vs 4

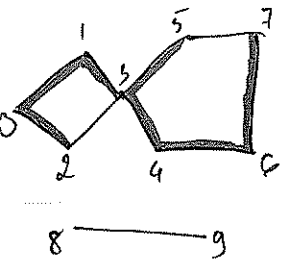
Nhan[6] ≠ Nhan[4]

=> Nhan[6] = 1

STPLT = 1

	0	1	2	3	4	5	6	7	8	9
Nhan	1	1	1	1	1	1	1	0	0	0

Bước 7:



Xét đỉnh 6. Ta có đỉnh 7 kề 6

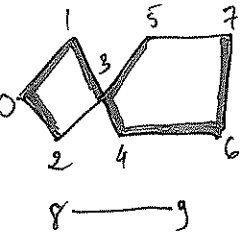
Nhan[7] ≠ Nhan[6]

=> Nhan[7] = 1

STPLT = 1

	0	1	2	3	4	5	6	7	8	9
Nhan	1	1	1	1	1	1	1	1	0	0

Bước 8:



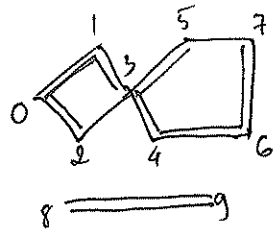
Xét đỉnh 7. Ta có 5 kề 7

- Nhan[5] = STPLT = Nhan[7]
- Ơi còn đỉnh 8, 9 chưa cập nhật Nhan

=> STPLT = 2

	0	1	2	3	4	5	6	7	8	9
Nhan	1	1	1	1	1	1	1	1	0	0

Bước 9:



Xét 8 => Nhan[8] = STPLT = 2

Ta có: 9 kề vs 8

Nhan[9] ≠ Nhan[8] => Nhan[9] = 2

STPLT = 2

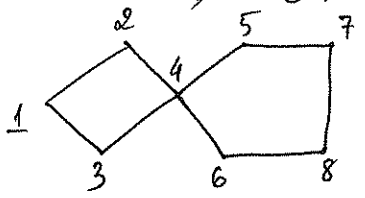
	0	1	2	3	4	5	6	7	8	9
Nhan	1	1	1	1	1	1	1	1	2	2

Bước 10: Tới đây không còn đỉnh nào chưa cập nhật Nhan.

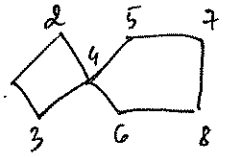
=> Dừng thuật toán.

- KL: Vậy STPLT = 2
  + TPLT thứ 1 gồm các đỉnh: 0, 1, 2, 3, 4, 5, 6, 7
  + TPLT thứ 2 gồm các đỉnh: 8, 9
  * Dừng khi Nhan ≠ 0

Đồ án đề 2: DFS.

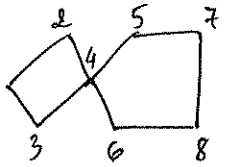


Bước 1: Khởi tạo



	1	2	3	4	5	6	7	8
ChuaXet	0	0	0	0	0	0	0	0
LuuVet	-1	-1	-1	-1	-1	-1	-1	-1

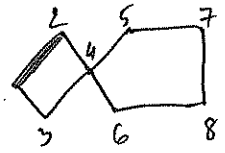
Bước 2: Xét đỉnh 1 (đề cho)



ChuaXet[1] = 1

	1	2	3	4	5	6	7	8
ChuaXet	1	0	0	0	0	0	0	0
LuuVet	-1	-1	-1	-1	-1	-1	-1	-1

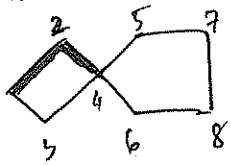
Bước 3: Từ 1 có đường đi đến 2, 3. Xét 2



- ChuaXet[2] = 1
- LuuVet[2] = 1

	1	2	3	4	5	6	7	8
ChuaXet	1	1	0	0	0	0	0	0
LuuVet	-1	1	-1	-1	-1	-1	-1	-1

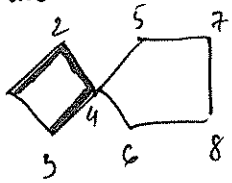
Bước 4: Từ 2 có đường đi đến 4. Xét 4.



- ChuaXet[4] = 1
- LuuVet[4] = 2

	1	2	3	4	5	6	7	8
ChuaXet	1	1	0	1	0	0	0	0
LuuVet	-1	1	-1	2	-1	-1	-1	-1

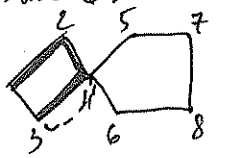
Bước 5: Từ 4 có đg đi đến 3, 5, 6. Xét 3.



- ChuaXet[3] = 1
- LuuVet[3] = 4

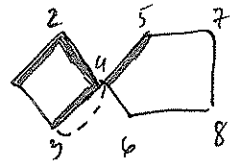
	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	0	0	0	0
LuuVet	-1	1	4	2	-1	-1	-1	-1

Bước 6: Từ 3 có đường đi đến 1, mà ChuaXet[1] = 1, quay lại xét 4



	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	0	0	0	0
LuuVet	-1	1	4	2	-1	-1	-1	-1

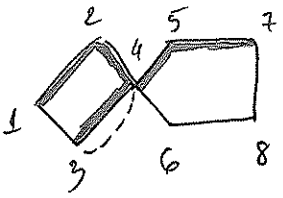
Bước 7: Từ 4 có đg đi đến 5, 6. Xét 5



- ChuaXet[5] = 1
- LuuVet[5] = 4

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	0	0	0
LuuVet	-1	1	4	2	4	-1	-1	-1

Bước 8:

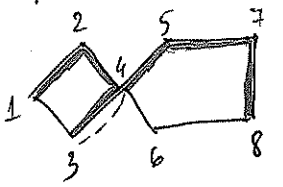


Từ 5 có đường đi đến 7. Xét 7

- ChuaXet[7] = 1
- LuuVet[7] = 5

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	0	1	0
LuuVet	-1	1	4	2	4	-1	5	-1

Bước 9:

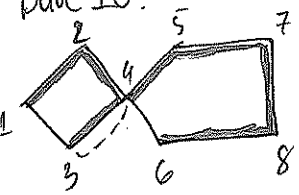


Từ 7 có đường đi đến 8. Xét 8

- ChuaXet[8] = 1
- LuuVet[8] = 7

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	0	1	1
LuuVet	-1	1	4	2	4	-1	5	7

Bước 10:



Từ 8 có đường đi đến 6. Xét 6.

- ChuaXet[6] = 1
- LuuVet[6] = 8

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	1	1	1
LuuVet	-1	1	4	2	4	8	5	7

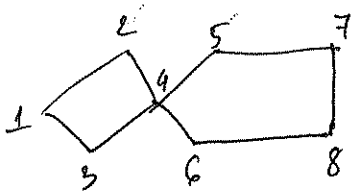
Bước 11:

Tới đây, tất cả các đỉnh đã xét => Dừng thuật toán

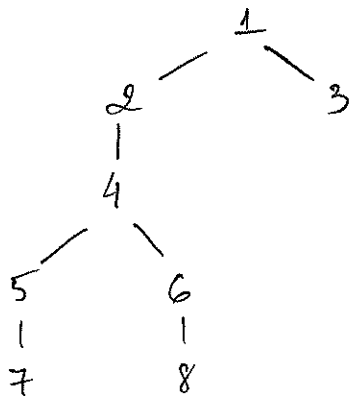
KL:

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	1	1	1
LuuVet	-1	1	4	2	4	8	5	7

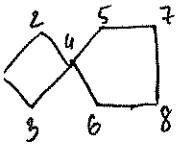
Vấn đề 3. BFS



(Nhập)



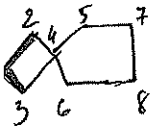
Bước 1: Khởi tạo



	1	2	3	4	5	6	7	8
ChuaXet	0	0	0	0	0	0	0	0
LuuVet	-1	-1	-1	-1	-1	-1	-1	-1

	0	1	2	3	4	5	6	7
Queue								

Bước 2:



. Xét đỉnh 1. Đưa 1 vào Queue => ChuaXet[1] = 1

. Queue chưa rỗng. Lấy từ Queue ra đỉnh 1

. Đỉnh 1 kề vs 2, 3 => Đưa 2, 3 vào Queue

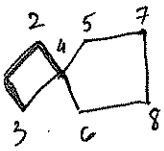
CX[2] = CX[3] = 1

LV[2] = LV[3] = 1

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	0	0	0	0	0
LuuVet	-1	+1	+1	-1	-1	-1	-1	-1

	0	1	2	3	4	5	6	7
Queue	~~1~~	2	3					

Bước 3:



. Queue chưa rỗng. Lấy từ Queue ra đỉnh 2

. Đỉnh 2 kề vs 4 => Đưa 4 vào Queue

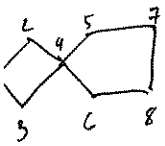
CX[4] = 1

LV[4] = 2

	1	2	3	4	5	6	7	8
Chuaxet	1	1	1	1	0	0	0	0
LuuVet	-1	1	1	2	-1	-1	-1	-1

Queue	~~1~~	~~2~~	3	4				

Bước 4:



. Queue chưa rỗng. Lấy từ Queue ra đỉnh 3.

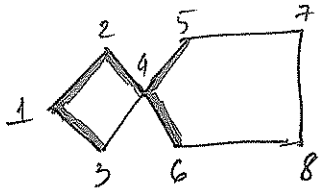
. Đỉnh 3 kề vs 1, 4. Mà 4, 1 xét rồi

=> Bỏ qua đỉnh 3.

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	0	0	0	0
LuuVet	-1	1	1	2	-1	-1	-1	-1

Queue	~~1~~	~~2~~	~~3~~	4				

Bước 5: Queue chưa rỗng, lấy 4 ra từ Queue

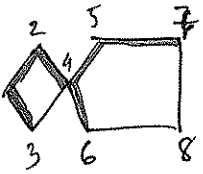


: Đỉnh 4 kề vs 5,6 => Đưa 5,6 vào Queue

- CX[5] = CX[6] = 1
- LV[5] = LV[6] = 4

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	1	0	0
LuuVet	-1	1	1	2	4	4	-1	-1
Queue	~~1~~	~~2~~	~~3~~	~~4~~	5	6		

Bước 6:

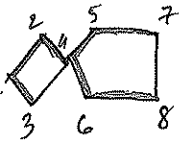


- Queue chưa rỗng. Lấy 5 ra từ Queue
- Đỉnh 5 kề vs 7. Đưa 7 vào Queue

CX[7] = 1 , LV[7] = 5

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	1	1	0
LuuVet	-1	1	1	2	4	4	5	-1
Queue	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	6	7	

Bước 7:

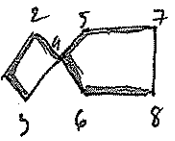


- Queue chưa rỗng. Lấy 6 ra từ Queue
- Đỉnh 6 kề vs 8. Đưa 8 vào Queue

CX[8] = 1 , LV[8] = 6

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	1	1	1
LuuVet	-1	1	1	2	4	4	5	6
Queue	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	7	8

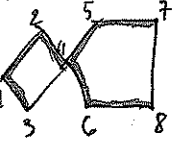
Bước 8:



- Queue chưa rỗng. Lấy 7 ra từ Queue
- Đỉnh 7 kề vs 8. Mà 8 xét rồi nên bỏ qua đỉnh 7.

	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	1	1	1
LuuVet	-1	1	1	2	4	4	5	6
Queue	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~	8

Bước 9:



- Queue chưa rỗng. Lấy 8 ra từ Queue
- Đỉnh 8 kề vs 6,7. Mà 6,7 xét rồi nên bỏ qua đỉnh 8.

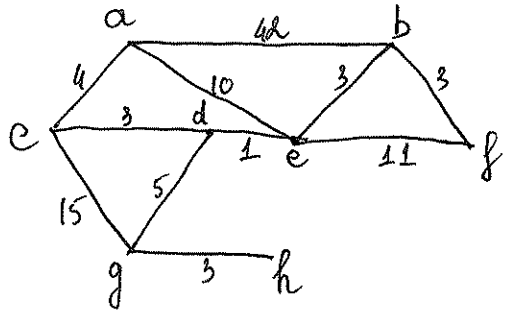
	1	2	3	4	5	6	7	8
ChuaXet	1	1	1	1	1	1	1	1
LuuVet	-1	1	1	2	4	4	5	6
Queue	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~	~~8~~

Bước 10: Queue rỗng. Tất cả các đỉnh đã xét
=> Dừng thuật toán

KL:

	1	2	3	4	5	6	7	8
CX	1	1	1	1	1	1	1	1
LV	-1	1	1	2	4	4	5	6

Đề 4: KRUSKAL



Bước 1: Duyệt các cạnh đồ thị để tạo dsách cạnh

list Edge = {(a,b); (a,c); (a,e); (b,e); (b,f); (e,g); (c,d); (d,g); (d,e); (e,f); (g,h)}

Bước 2: Sắp xếp lại các cạnh của list Edge theo trọng số tăng dần.

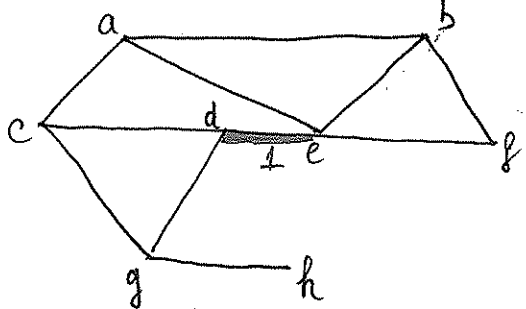
list Edge = {(d,e); (b,f); (c,d); (e,b); (g,h); (a,c); (d,g); (a,e); (e,f); (c,g); (a,b)}

Khởi tạo $T = \emptyset$

Bước 3: Lấy từ list Edge ra cạnh nhỏ nhất. Nếu $T \cup \{e\}$ không chứa chu trình $\Rightarrow T = T \cup \{e\}$

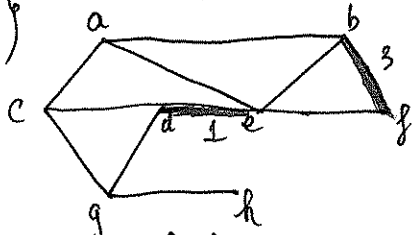
Lấy từ list Edge ra cạnh (d,e). $T \cup (d,e)$ k° tạo ra chu trình.

$\Rightarrow T = \{(d,e)\}$



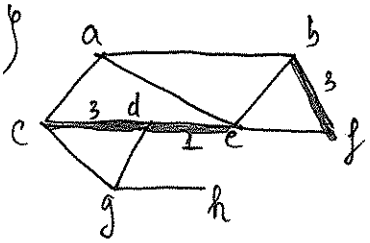
Bước 4: Lấy từ list Edge ra cạnh (b,f). $T \cup (b,f)$ k° tạo ra chu trình.

$\Rightarrow T = \{(d,e); (b,f)\}$



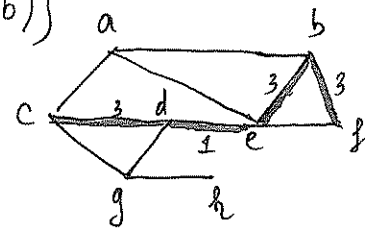
Bước 5: Lấy từ list Edge ra cạnh (c,d). $T \cup (c,d)$ k° tạo ra chu trình.

$\Rightarrow T = \{(d,e); (b,f); (c,d)\}$



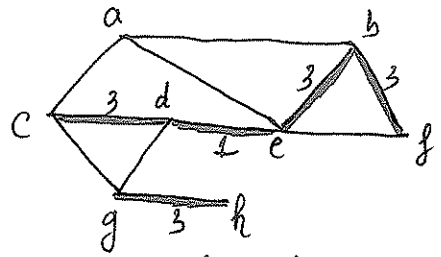
Bước 6: Lấy từ list Edge ra cạnh (e,b). $T \cup (e,b)$ k° tạo ra chu trình.

$\Rightarrow T = \{(d,e); (b,f); (c,d); (e,b)\}$



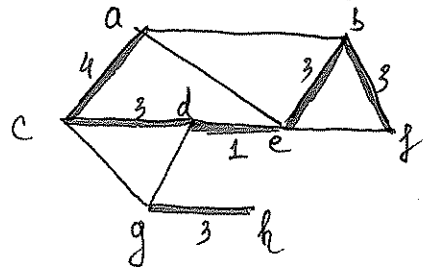
Bước 7: Lấy từ list Edge ra cạnh (g,h). T ∪ (g,h) k° tạo chu trình.

=> T = {(d,e); (b,f); (c,d); (e,b); (g,h)}



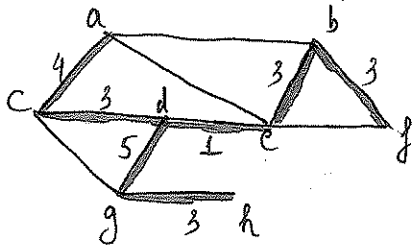
Bước 8: Lấy từ list Edge ra cạnh (a,c). T ∪ (a,c) k° tạo chu trình.

=> T = {(d,e); (b,f); (c,d); (e,b); (g,h); (a,c)}



Bước 9: Lấy từ list Edge ra cạnh (d,g). T ∪ (d,g) k° tạo chu trình.

=> T = {(d,e); (b,f); (c,d); (e,b); (g,h); (a,c); (d,g)}

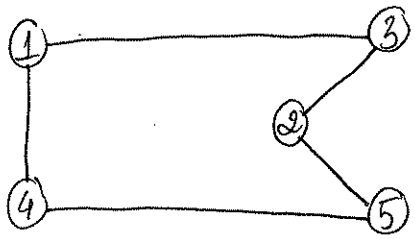


Bước 10: T có 7 cạnh = n - 1 (n là số đỉnh) => Dừng thuật toán

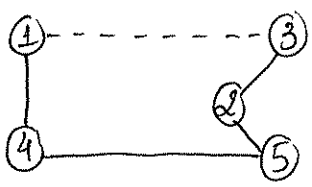
KL: Cây khung nhỏ nhất gồm các cạnh (d,e); (b,f); (c,d); (e,b); (g,h); (a,c); (d,g).

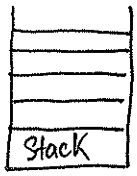
Tổng trọng số: 1 + 3 + 3 + 3 + 3 + 4 + 5 = 22

Đề tài số 5: Euler

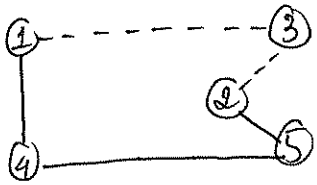


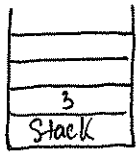
Bước 1: Xuất phát từ đỉnh 1. Có cạnh (1,3); (1,4).
Xét cạnh (1,3), xóa cạnh (1,3)



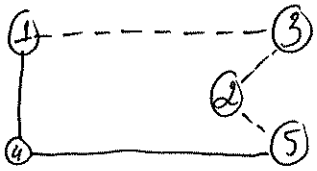


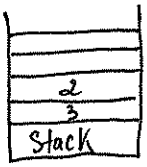
Bước 2: Tại 3 có cạnh (3,2).
Xét cạnh (3,2), xóa cạnh (3,2)



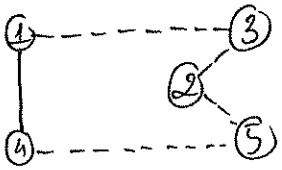


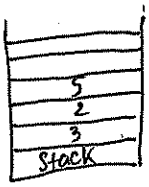
Bước 3: Tại 2 có cạnh (2,5)
Xét cạnh (2,5), xóa cạnh (2,5)



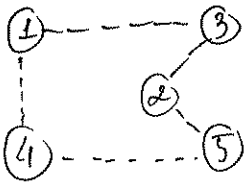


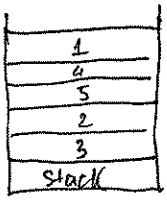
Bước 4: Tại 5 có cạnh (5,4)
Xét cạnh (5,4), xóa cạnh (5,4)



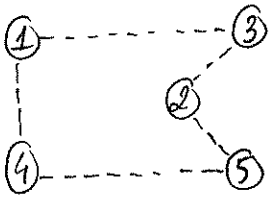


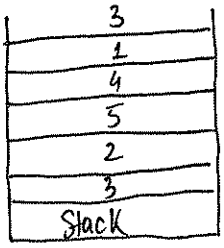
Bước 5: Tại 4 có cạnh (4,1)
Xét cạnh (4,1), xóa cạnh (4,1)





Bước 6: Tại 1 không còn cạnh nối, quay lui về 4, 5, 2, 3. Đưa 3 vào Stack



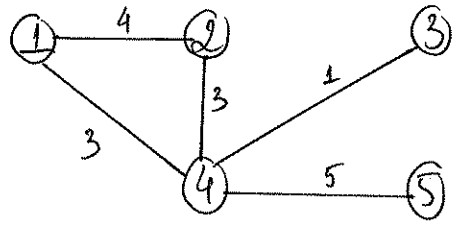


Kết luận:

Vậy đồ thị có chu trình Euler

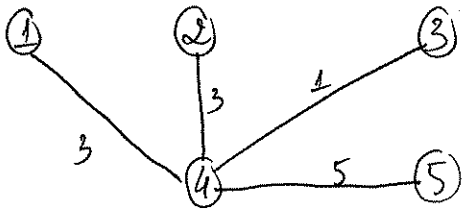
3 -> 1 -> 4 -> 5 -> 2 -> 3

Đề 6: PRIM



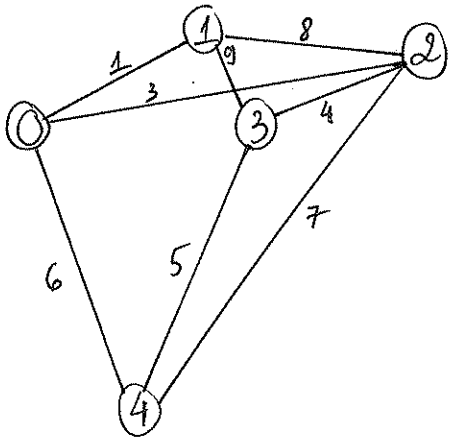
Vòng lặp	1	2	3	4	5	V_T
Khởi tạo	—	[1, 4]	∞	[1, 3]	∞	1
1	—	[4, 3]	[4, 1]	—	[4, 5]	1, 4
2	—	~~∞~~ [4,3]	—	—	[4, 5]	1, 4, 3
3	—	—	—	—	[4, 5]	1, 4, 3, 2
4	—	—	—	—	—	1, 4, 3, 2, 5
						((1,4) (4,3) (3,2) (2,5))

Kết luận: Cây khung nhỏ nhất gồm các cạnh (1,4); (2,4); (3,4); (4,5)



Tổng trọng số nhỏ nhất: 3 + 3 + 1 + 5 = 12.

Vấn đề 7: DIJKSTRA



Vòng lặp	0	1	2	3	4
Khởi tạo	0, 0	1, 0	3, 0	∞	6, 0
1	—	—	3, 0	9 ~~8~~, 1	6, 0
2	—	—	—	4, 2	6, 0
3	—	—	—	~~7~~ ~~8~~	6, 0
4	—	—	—	—	—

trọng số
(a, b)
cạnh nối trước

Đáy đường đi ngắn nhất từ 0 đến các đỉnh còn lại là