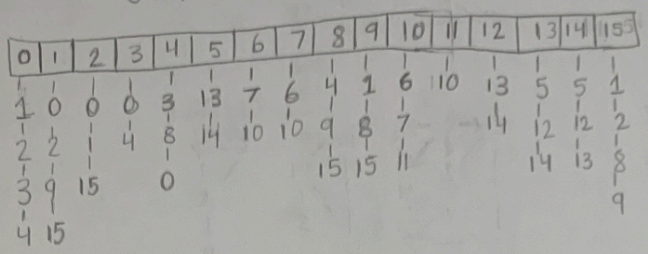
Programming assignment 12

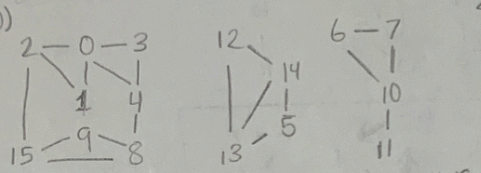
1 Undirected graphs

1-1)

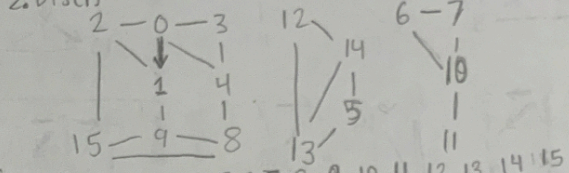


1-2)

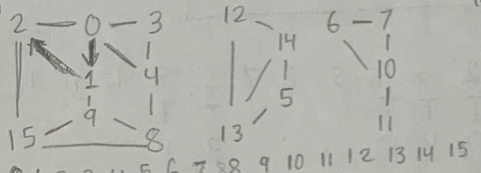
1. Dfs(0)



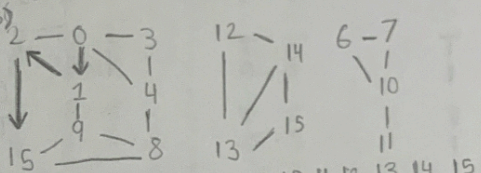
2. Dfs(1)



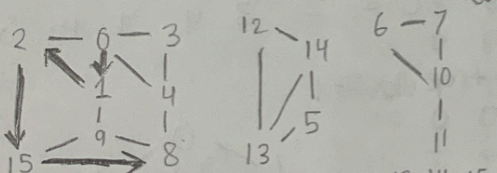
3. Dfs(2)



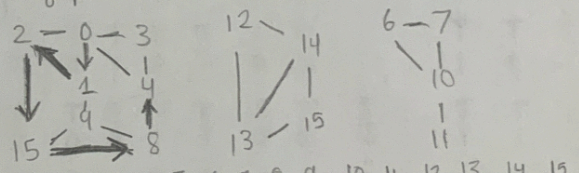
4. Dfs(15)



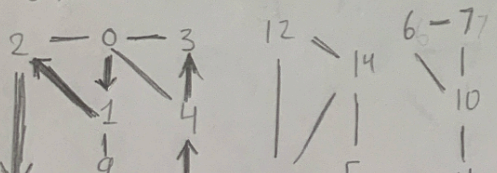
5. Dfs(8)



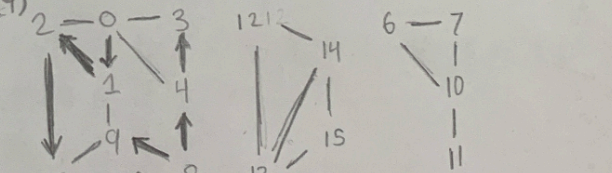
6. Dfs(4)



7. Dfs(3)

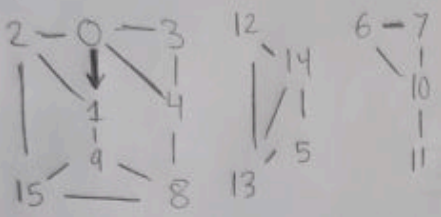


8. Dfs(9)

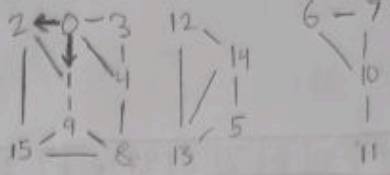


1-3)

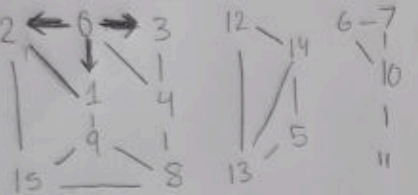
1.



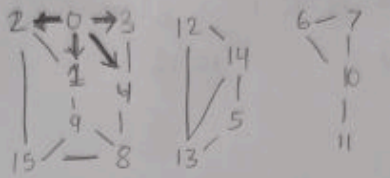
2.



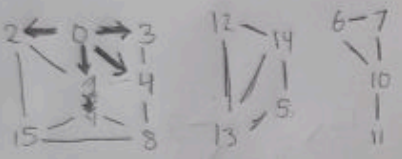
3.



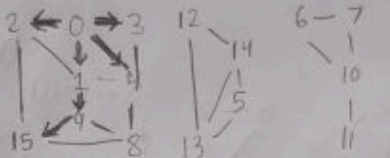
4.



5.

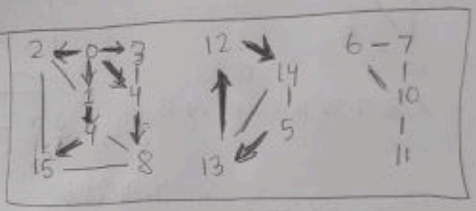


6.



7.

8.

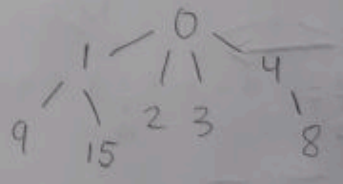


Final:

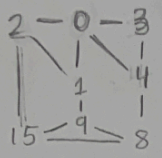
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	T	T	T	T				T	T						T
	0	0	0	0				9	1						1

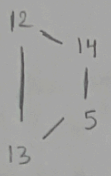
Queue: 0 1 2 3 4 9 15 8

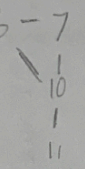
tree built:



1-4)





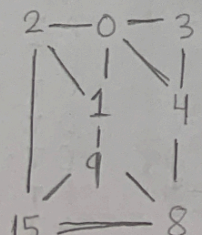


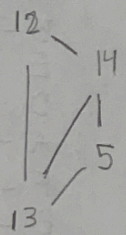
dfs(0) count=0

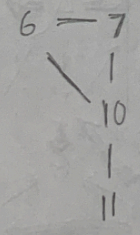
marked →
ID →

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	T															
	0															
dfs(1)	T	T														
c=0	0	0														
dfs(2)	T	T	T													
c=0	0	0	0													
dfs(15)	T	T	T													T
c=0	0	0	0													0
dfs(9)	T	T	T							T						T
c=0	0	0	0							0						0
dfs(8)	T	T	T						T	T						T
c=0	0	0	0						0	0						0
dfs(4)	T	T	T		T				T	T						T
c=0	0	0	0		0				0	0						0
dfs(3)	T	T	T	T	T				T	T						T
c=0	0	0	0	0	0				0	0						0
dfs(5)	T	T	T	T	T	T			T	T						T
c=1	0	0	0	0	0	1			0	0						0
dfs(13)	T	T	T	T	T	T			T	T				T		T
c=1	0	0	0	0	0	1			0	0				1		0
dfs(12)	T	T	T	T	T	T			T	T			T	T		T
c=1	0	0	0	0	0	1			0	0			1	1		0
dfs(14)	T	T	T	T	T	T			T	T			T	T	T	T
c=1	0	0	0	0	0	1			0	0			1	1	1	0
dfs(6)	T	T	T	T	T	T	T		T	T			T	T	T	T
c=2	0	0	0	0	0	1	2		0	0			1	1	1	0
dfs(7)	T	T	T	T	T	T	T	T	T	T			T	T	T	T
c=2	0	0	0	0	0	1	2	2	0	0			1	1	1	0
dfs(10)	T	T	T	T	T	T	T	T	T	T	T		T	T	T	T
c=2	0	0	0	0	0	1	2	2	0	0	2		1	1	1	0
dfs(11)	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
c=2	0	0	0	0	0	1	2	2	0	0	2	2	1	1	1	0

1-5)







0 1 2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

dfs(0,0) T

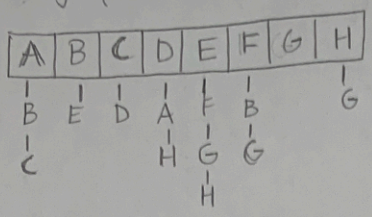
dfs(1,0) T T

dfs(2,1) T T

stops here

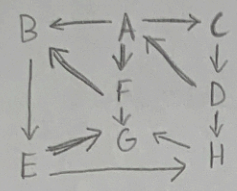
2) Directed graphs

2-1)



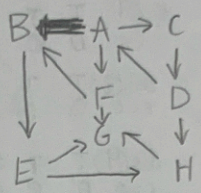
2-2)

1. dfs(A)



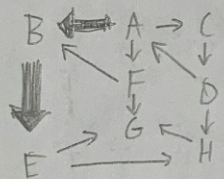
adjacent	marked
[B] - C	A T
E	B T
D	C
A - H	D
F - G - H	E
B - G	F
H	G
	H

2. dfs(B)



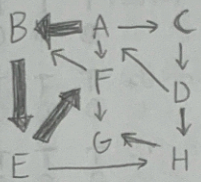
adjacent	marked
B - C	A T
[E]	B T
D	C
A - H	D
F - G - H	E
B - G	F
H	G
	H

3. dfs(E)



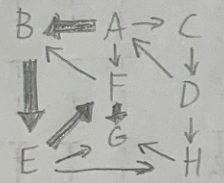
adjacent	marked
B - C	A T
E	B T
D	C
A - H	D
[E] - G - H	E T
B - G	F
H	G
	H

4. dfs(F)



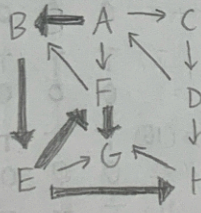
adjacent	marked
B - C	A T
E	B T
D	C
A - H	D
F - G - H	E
B - [G]	F T
H	G
	H

5. dfs(G)



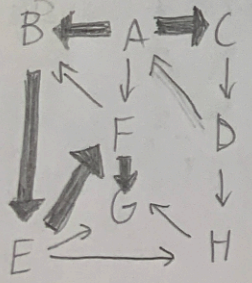
adjacent	marked
B - C	A T
E	B T
D	C
A - H	D
F - G - [H]	E T
B - G	F T
H	G
	H

6. dfs(H)



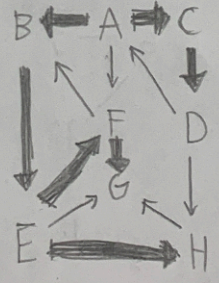
adjacent	marked
B - [C]	A T
E	B T
D	C
A - H	D
F - G - H	E T
B - G	F T
H	G T
	H T

7. dfs(C)



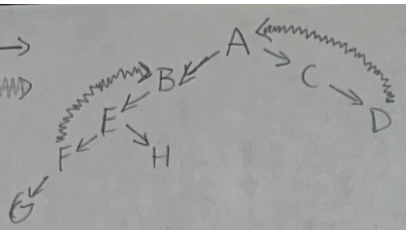
adjacent	marked
B - C	A T
E	B T
[D]	C T
A - H	D
F - G - H	E T
B - G	F T
H	G T
	H T

8. dfs(D)

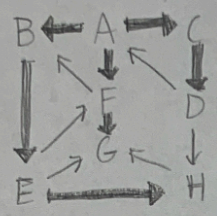


adjacent	marked
B - C	A T
E	B T
D	C T
A - H	D T
F - G - H	E T
B - G	F T
H	G T
	H T

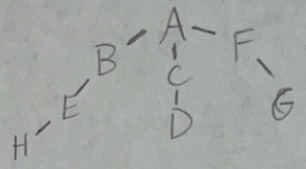
So tree edge: →
back edge: ⇝



2-3)

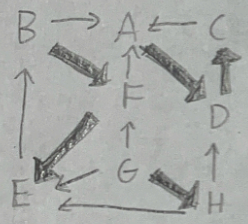


queue: A B C F E D G H



2-4)

1. G^R:



2. dfs for an unmarked V on G^R, going A-Z:

dfs(A)
dfs(D)
dfs(C) → C D A

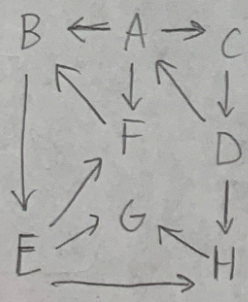
dfs(B)
dfs(F)
dfs(E) → E F B

dfs(G)
dfs(H) → H G

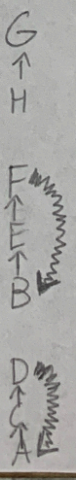
final:
C D A E F B H G

marked:	A	B	C	D	E	F	G
	T	T	T	T	T	T	T

3. dfs in order of reversed stack, so GHBFEADC for G^{nr}



dfs(G) — G
dfs(H) — H
dfs(B) —
dfs(E)
dfs(F) — FEB
dfs(A)
dfs(C)
dfs(D) — DCA



strongly-connected components

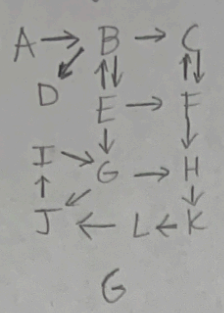
{G} {H}

{B E F}

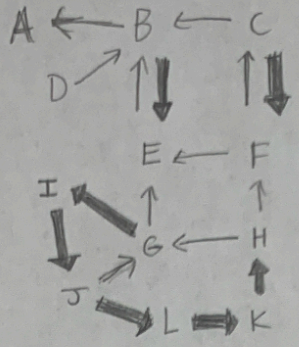
{A C D}

3 Strongly connected components

3-1)



reversed becomes →



1. dfs in order from A-Z on G^R:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
marked:	T	T	T	T	T	T	T	T	T	T	T	T				

dfs(A) — A
dfs(B)
dfs(E) — E
B
A

dfs(C)
dfs(F) — F
C

dfs(D) — D

dfs(G)
dfs(I)
dfs(J)
dfs(L)
dfs(K)
dfs(H) — H
K
L
J
I
G

So our stack is
G I J L K H D C F B E A

2. dfs in order of the stack on G^{rr}:

	G	I	J	L	K	H	D	C	F	B	E	A
marked:	T	T	T	T	T	T	T	T	T			

dfs(G)
dfs(J)
dfs(I) — I
J

dfs(D) — D
dfs(C) —
dfs(F) — F
C

dfs(H)
dfs(K)
dfs(L) — L
K
H
G

dfs(B)
dfs(E) — E
dfs(A) B
dfs(A) — A

I
J
L
K
H
G
D
F
C

E
B
A

Strongly connected components are:

{G J I H K L}
{D} {C F}
{B E} {A}