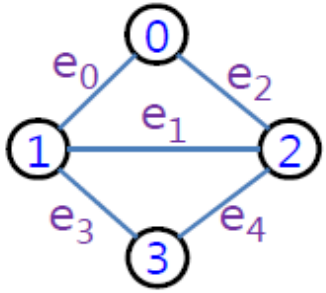
◆ Adjacent List Implementation Using Linked Lists



Vertex Array

V[0] v0 → 2 → 0 → ϕ

V[1] v1 → 3 → 1 → 0 → ϕ

V[2] v2 → 4 → 2 → 1 → ϕ

V[3] v3 → 4 → 3 → ϕ

edge array index

각각 stack으로 구현

vertex and edge name

any information

E[0]	e0		0	1
E[1]	e1		2	1
E[2]	e2		0	2
E[3]	e3		3	1
E[4]	e4		3	2

vertex array index

Corresponding C Code

```
typedef struct ptr_list {
  int i;
  ptr_L *p;
} ptr_L;
typedef struct vertex {
  // any data field on this vertex;
  ptr_L *p;
} vertex;
typedef struct edge {
  int cost;    // may have more information
  int vf, vr; // the two vertex indices
} edge;

vertex *V = NULL;   edge   *E = NULL;
int    V_Num = 0, E_num = 0;
...
get V_Num and E_Num;
V = (vertex *)malloc(V_Num * sizeof(vertex));
E = (edge *)malloc(E_Num * sizeof(edge));
memory allocation error checking;
... // Do whatever we need.
free(V); free(E); // must free adj list in V
```

name
cost etc
flag

name, colur etc
flag

STL list을 사용할 경우
(DBL이라 낭비다)

```
typedef struct vertex {
    . . .
    std::list <int> p;
}
. . .
V[k].p.push_front(10);
```

(A)

(B) Linked list 필요 없음

An Input Format (~.txt)

```
T
VN   EN   src
vi   vj   cost
.  .  .
.  .  .
VN   EN   src
vi   vj   cost
.  .  .
.  .  .
```

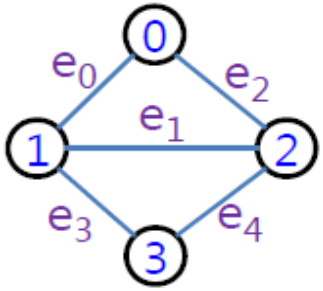
T1

T2

TT

T : number of graphs
VN : number of vertices
EN : number of edges
src : source vertex index (for any application)
vi, vj : vertex indices
cost : cost of the edge

Example




```
1
4  5  0
0  1  1
2  1  1
0  2  1
3  1  1
3  2  1
```

Source vertex index is 0.
All the edge costs are 1.

- DFS and DFS Spanning Tree
  - A Generic DFS Procedure

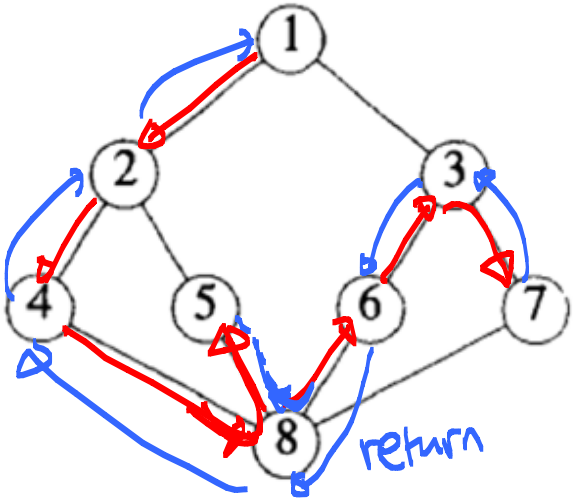
```
void DFS(int v){
     visited[v] = 1;
     for each vertex w adjacent from v {
        if (!visited[w]){DFS(w);}
}
```

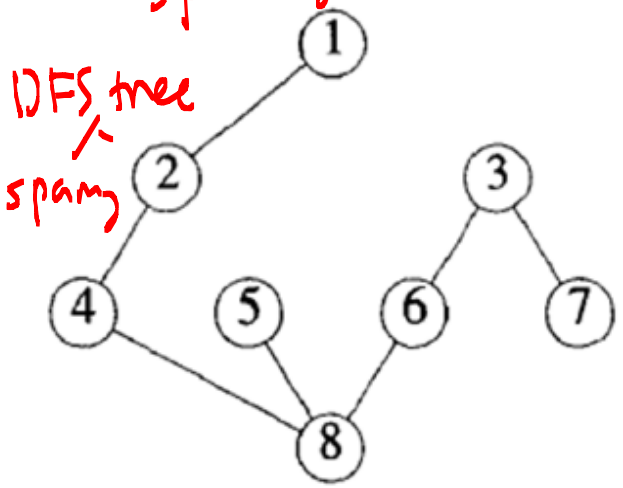
Tree 인지 check
① cycle check
② edge 개수 = |V|-1
tree

{ 3
ECv-w): flag=1

in main
DFS(1);

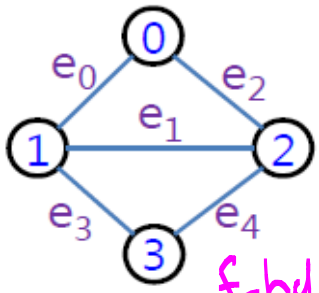
  - DFS Spanning Tree





(a) DFS(1) spanning tree

Adjacent List Implementation Using Arrays



Vertex Array

			f_hd		r_hd
V[0]	v0		e2		-1
V[1]	v1		-1		e3
V[2]	v2		e1		e4
V[3]	v3		e4		-1

header

header

f_hd → V[0] → e2 → e0 → -1

V2

f_hd → e1 → -1

r_hd → e4 → e2 → -1

Linked list adj list 기억.

vertex and edge name

any information

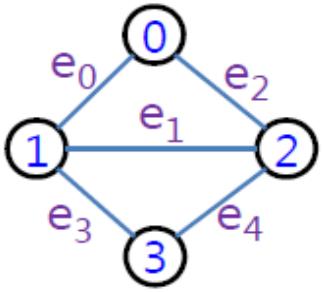
			fp	vf	vr	rp
E[0]	e0		-1	0	1	-1
E[1]	e1		-1	2	1	e0
E[2]	e2		e0	0	2	-1
E[3]	e3		-1	3	1	e1
E[4]	e4		e3	3	2	e2

	1		
	4	5	0
0	0	1	1
1	2	1	1
2	0	2	1
3	3	1	1
4	3	2	1

한줄씩 읽어서 setting.

✦ Step by Step Status

f_hd 또는 r_hd 가 NONE 이면 edge index write




```
1
4  5  0
0  1  1
2  1  1
0  2  1
3  1  1
3  2  1
```

e0=(v0,v1)

e1=(v2,v1)

f_hd r_hd

Vertex Array

			f_hd		r_hd
V[0]	v0		e0		-1
V[1]	v1		-1		e0
V[2]	v2		e1 -1		-1
V[3]	v3		-1		-1

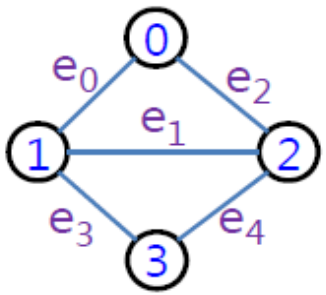
e1 ⑤

			fp	vf	vr	rp
E[0]	e0		-1	0	1	-1
E[1]	e1		-1	-1	-1	-1
E[2]	e2		-1	-1	-1	-1
E[3]	e3		-1	-1	-1	-1
E[4]	e4		-1	-1	-1	-1

④

이동

✦ Step by Step Status




```
1
4 5 0
0 1 1
2 1 1
0 2 1
3 1 1
3 2 1
```

e2=(v0, v2)

f_hd 에는 e1
r_hd의 e0를 rp에 복사, r_hd에는 e1 write

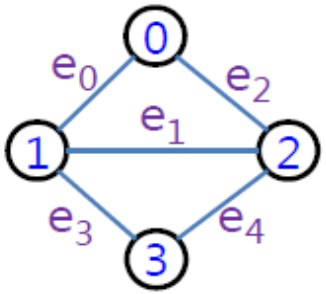
Vertex Array

		f_hd	r_hd
V[0]	v0	e0	-1
V[1]	v1	-1	e1
V[2]	v2	e1	-1
V[3]	v3	-1	-1

		fp	vf	vr	rp
E[0]	e0	-1	0	1	-1
E[1]	e1	-1	2	1	e0
E[2]	e2	-1	-1	-1	-1
E[3]	e3	-1	-1	-1	-1
E[4]	e4	-1	-1	-1	-1

v0 → e0 → x
v1 → x → e1 → e0 → x
v2 → e1 → x

✦ Step by Step Status (edge 2)




```
1
4 5 0
0 1 1
2 1 1
0 2 1
3 1 1
3 2 1
```

e3=(v3, v1)

index 2

Vertex Array

			f_hd		r_hd
V[0]	v0		**e2**		-1
V[1]	v1		-1		**e1**
V[2]	v2		**e1**		**e2**
V[3]	v3		-1		-1

e3

v0 → e2 → e0 → x
↳ x

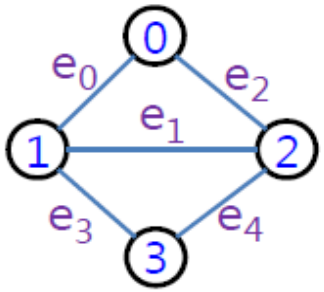
v1 → x
↳ e1 → e0 → x

v2 → e1 → x
↳ e2 → x

v3 → x
↳ x

			fp	vf	vr	rp
E[0]	e0		-1	**0**	**1**	-1
E[1]	e1		-1	**2**	**1**	**e0**
E[2]	e2		**e0**	**0**	**2**	-1
E[3]	e3		-1	3 -1	1 -1	-1 e1
E[4]	e4		-1	-1	-1	-1

✦ Step by Step Status (edge 3)




```
1
4 5 0
0 1 1
2 1 1
0 2 1
3 1 1
3 2 1
```

(3, 2)

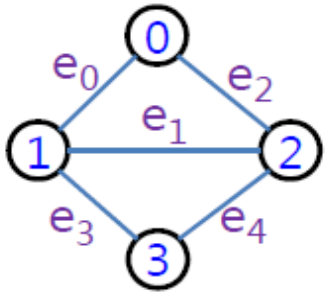
e4 :

Vertex Array

			f_hd		r_hd
V[0]	v0		**e2**		-1
V[1]	v1		-1		**e3**
V[2]	v2		**e1**		**e2** e4
V[3]	v3		e4 **e3**		-1

			fp	vf	vr	rp
E[0]	e0		-1	**0**	**1**	-1
E[1]	e1		-1	**2**	**1**	**e0**
E[2]	e2		**e0**	**0**	**2**	-1
E[3]	e3		-1	**3**	**1**	**e1**
E[4]	e4		e3 -1	~~1~~ 3	~~1~~ 1	-1 e2

✦ Step by Step Status (edge 4)




```
1
4  5  0
0  1  1
2  1  1
0  2  1
3  1  1
3  2  1
```

Vertex Array

			f_hd		r_hd
V[0]	v0		**e2**		-1
V[1]	v1		-1		**e3**
V[2]	v2		**e1**		**e4**
V[3]	v3		**e4**		-1

			fp	vf	vr	rp
E[0]	e0		-1	**0**	**1**	-1
E[1]	e1		-1	**2**	**1**	**e0**
E[2]	e2		**e0**	**0**	**2**	-1
E[3]	e3		-1	**3**	**1**	**e1**
E[4]	e4		**e3**	**3**	**2**	**e2**

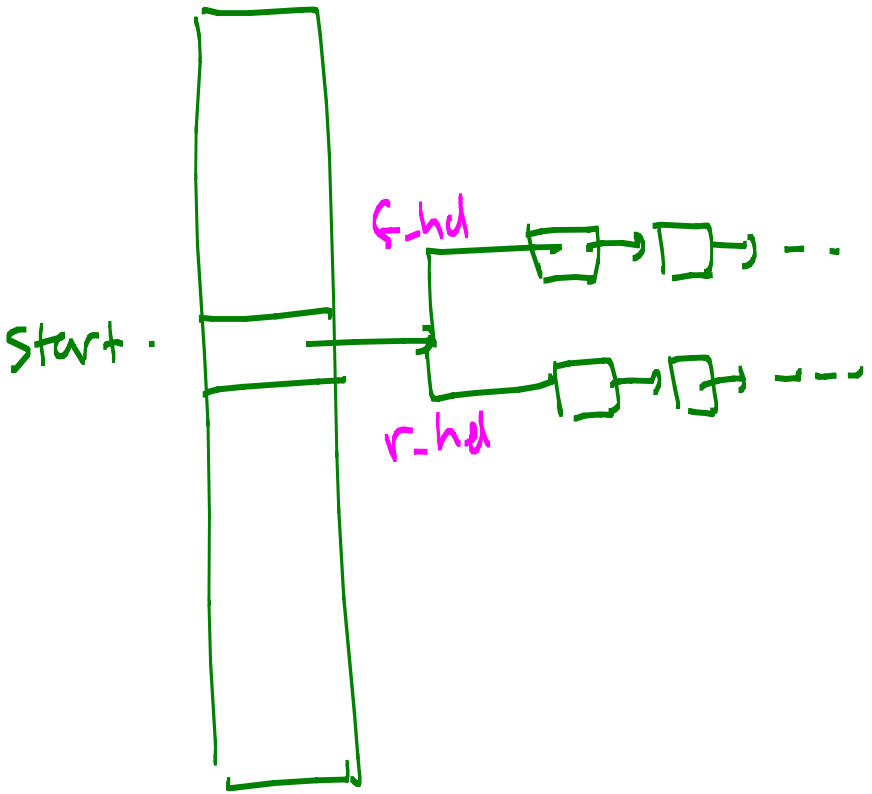
$v_0 \rightarrow e_2 \rightarrow e_0 \rightarrow \times$
$\quad\quad \rightarrow \times$

$v_1 \rightarrow \times$
$\quad\quad \rightarrow e_3 \rightarrow e_1 \rightarrow e_0 \rightarrow$

$v_2 \rightarrow e_1 \rightarrow \times$
$\quad\quad \rightarrow e_4 \rightarrow e_2 \rightarrow \times$

$v_3 \rightarrow e_4 \rightarrow e_3 \rightarrow \times$

DFS




```
DFS(v)
    visited[v] = 1
    for ( w ~ f_hd )
        if( not visited ) DFS(w)
    for ( w ~ r_hd )
        if( not visited ) DFS(w)
```

✦ Another Example

f_hd

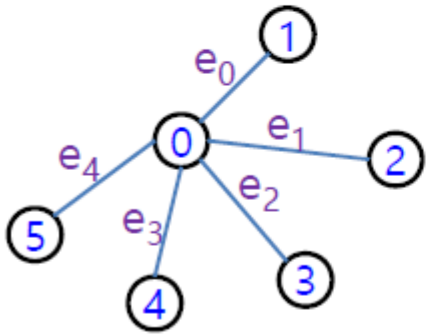
v0->e0

	1		
	6	5	0
e0	0	1	1
e1	0	2	1
e2	0	3	1
e3	0	4	1
e4	0	5	1

방금 읽은 에지는
항상 -1 -1이다

	f_hd	r_hd
V[0]	e0	-1
V[1]	-1	e0
V[2]	-1	-1
V[3]	-1	-1
V[4]	-1	-1
V[5]	-1	-1

	fp	vf	vr	rp
E[0]	-1	0	1	-1
E[1]	-1	-1	-1	-1
E[2]	-1	-1	-1	-1
E[3]	-1	-1	-1	-1
E[4]	-1	-1	-1	-1



v0->e1->e0

	f_hd	r_hd
V[0]	e1	-1
V[1]	-1	e0
V[2]	-1	e1
V[3]	-1	-1
V[4]	-1	-1
V[5]	-1	-1

	fp	vf	vr	rp
E[0]	-1	0	1	-1
E[1]	e0	0	2	-1
E[2]	-1	-1	-1	-1
E[3]	-1	-1	-1	-1
E[4]	-1	-1	-1	-1

✦ Another Example

	1		
	6	5	0
e0	0	1	1
e1	0	2	1
e2	0	3	1
e3	0	4	1
e4	0	5	1

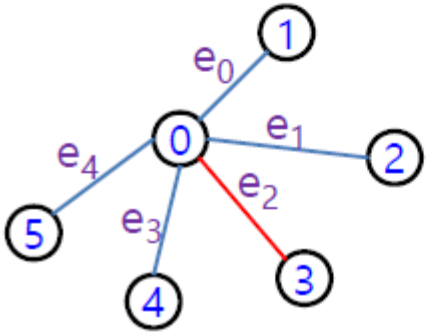
방금 읽은 에지는
항상 -1 -1이다

v0->e1->e0

	f_hd	r_hd
V[0]	e1	-1
V[1]	-1	e0
V[2]	-1	e1
V[3]	-1	-1
V[4]	-1	-1
V[5]	-1	-1

	fp	vf	vr	rp
E[0]	-1	0	1	-1
E[1]	e0	0	2	-1
E[2]	-1	-1	-1	-1
E[3]	-1	-1	-1	-1
E[4]	-1	-1	-1	-1

v0->e2->e1->e0



	f_hd	r_hd
V[0]	e2	-1
V[1]	-1	e0
V[2]	-1	e1
V[3]	-1	e2
V[4]	-1	-1
V[5]	-1	-1

	fp	vf	vr	rp
E[0]	-1	0	1	-1
E[1]	e0	0	2	-1
E[2]	e1	0	3	-1
E[3]	-1	-1	-1	-1
E[4]	-1	-1	-1	-1

✦ Another Example

	1		
	6	5	0
e0	0	1	1
e1	0	2	1
e2	0	3	1
e3	0	4	1
e4	0	5	1

v0->e2->e1->e0

	f_hd	r_hd
V[0]	e2	-1
V[1]	-1	e0
V[2]	-1	e1
V[3]	-1	e2
V[4]	-1	-1
V[5]	-1	-1

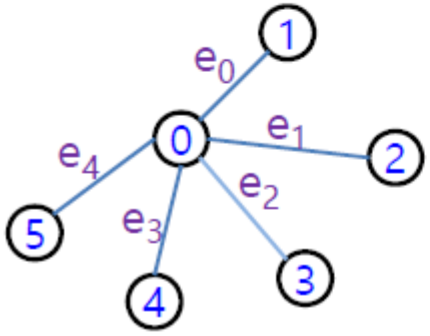
	fp	vf	vr	rp
E[0]	-1	0	1	-1
E[1]	e0	0	2	-1
E[2]	e1	0	3	-1
E[3]	-1	-1	-1	-1
E[4]	-1	-1	-1	-1

방금 읽은 에지의 fp로
V[0]의 f_hd를 이동
V[0]의 f_hd에 새로운 에지 index 저장

v0->e3->e2->e1->e0

	f_hd	r_hd
V[0]	e3	-1
V[1]	-1	e0
V[2]	-1	e1
V[3]	-1	e2
V[4]	-1	e3
V[5]	-1	-1

	fp	vf	vr	rp
E[0]	-1	0	1	-1
E[1]	e0	0	2	-1
E[2]	e1	0	3	-1
E[3]	e2	0	4	-1
E[4]	-1	-1	-1	-1



✦ Another Example

	1		
	6	5	0
e0	0	1	1
e1	0	2	1
e2	0	3	1
e3	0	4	1
e4	0	5	1

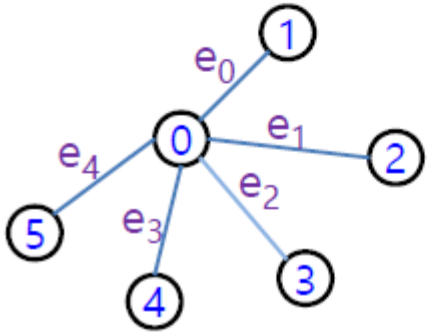
v0->e3->e2->e1->e0

방금 읽은 에지의 fp로
V[0]의 f_hd를 이동
V[0]의 f_hd에 새로운 에지 index 저장

	f_hd	r_hd
V[0]	e3	-1
V[1]	-1	e0
V[2]	-1	e1
V[3]	-1	e2
V[4]	-1	e3
V[5]	-1	-1

	fp	vf	vr	rp
E[0]	-1	0	1	-1
E[1]	e0	0	2	-1
E[2]	e1	0	3	-1
E[3]	e2	0	4	-1
E[4]	-1	-1	-1	-1

v0->e4->e3->e2->e1->e0



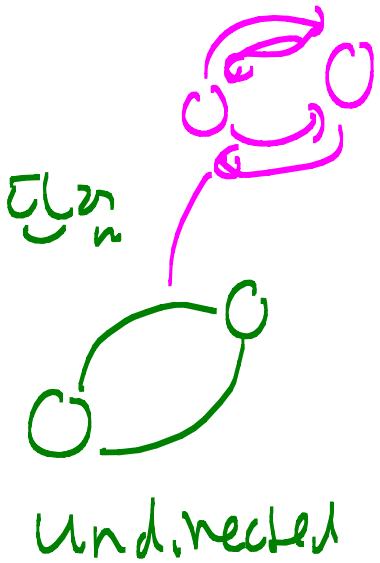
	f_hd	r_hd
V[0]	e4	-1
V[1]	-1	e0
V[2]	-1	e1
V[3]	-1	e2
V[4]	-1	e3
V[5]	-1	e4

	fp	vf	vr	rp
E[0]	-1	0	1	-1
E[1]	e0	0	2	-1
E[2]	e1	0	3	-1
E[3]	e2	0	4	-1
E[4]	e3	0	5	-1

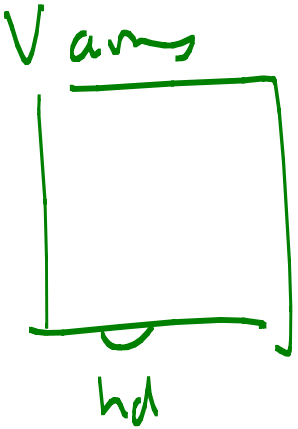
Directed Graph

header가 1개이면 충분

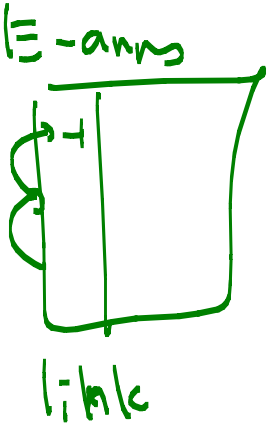
e1 3 → 2



undirected

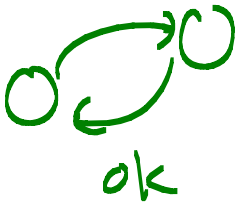


hd



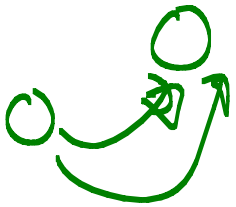
link

directed



ok

directed



모두 가능