Chapter 14 (continues)

Chapter 14 (Sections 14.4-14.5)

We will discuss:

- Depth first search algorithm (Section 14.4)
- Minimum spanning trees algorithms:
  - Kriskal's
  - Prim's (self-study)

Depth First Search algorithm

- The DFS algorithm moves along one path as far as possible before backtracking and examining other paths off the earlier discovered vertices.
- During the DFS execution, each vertex goes through three phases:
  - the vertex has not yet been discovered.
  - the vertex has been discovered, but the algorithm has not completed processing of all the vertices accessible from it.
  - we finished processing the vertex and all the vertices reachable from it

Depth First Search algorithm

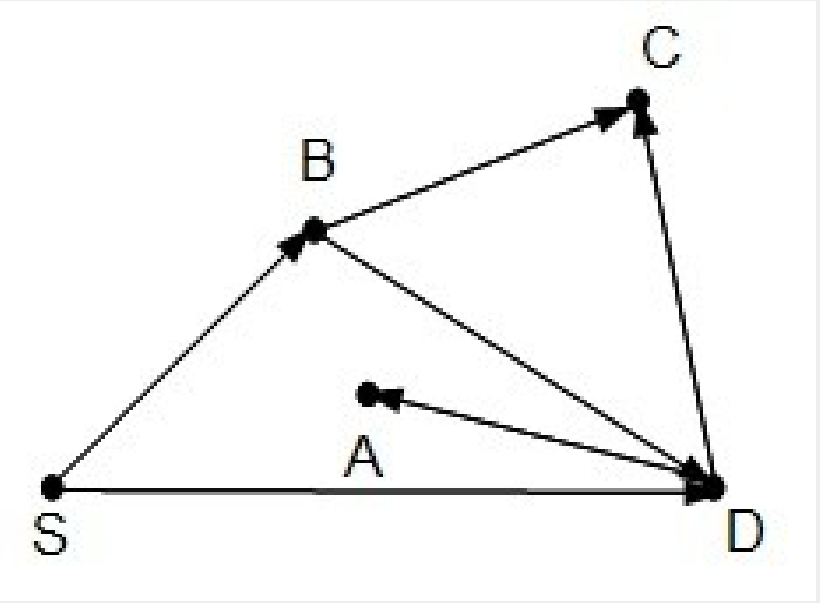
```
dfs(g):
   for each vertex v in g:
      set v's start time to 0
   t=0
   for each vertex v in g:
      if v's start time is 0:
         dfs_traverse(g,v)

dfs_traverse(g,v):
   t += 1
   set v's start time to t
   for each vertex u adjacent to v:
      if u's start time is 0:
         set u's parent to v
         dfs_traverse(g,u)
   t += 1
   set v's end time to t
```

Depth First Search algorithm

```
dfs(g):
   for each vertex v in g:
      set v's start time to 0
   t=0
   for each vertex v in g:
      if v's start time is 0:
         dfs_traverse(g,v)

dfs_traverse(g,v):
   t += 1
   set v's start time to t
   for each vertex u adjacent to v:
      if u's start time is 0:
         set u's parent to v
         dfs_traverse(g,u)
   t += 1
   set v's end time to t
```

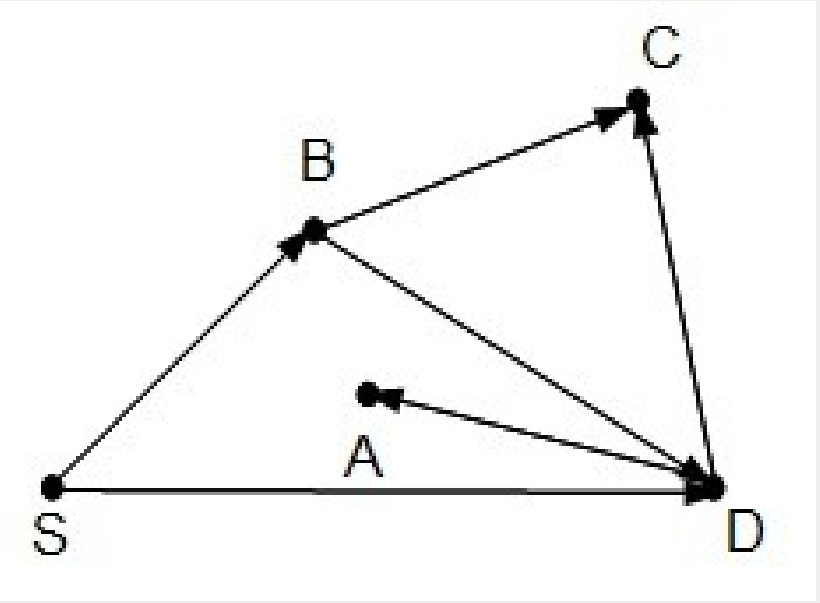


	S	A	B	C	D
par					
st					
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```

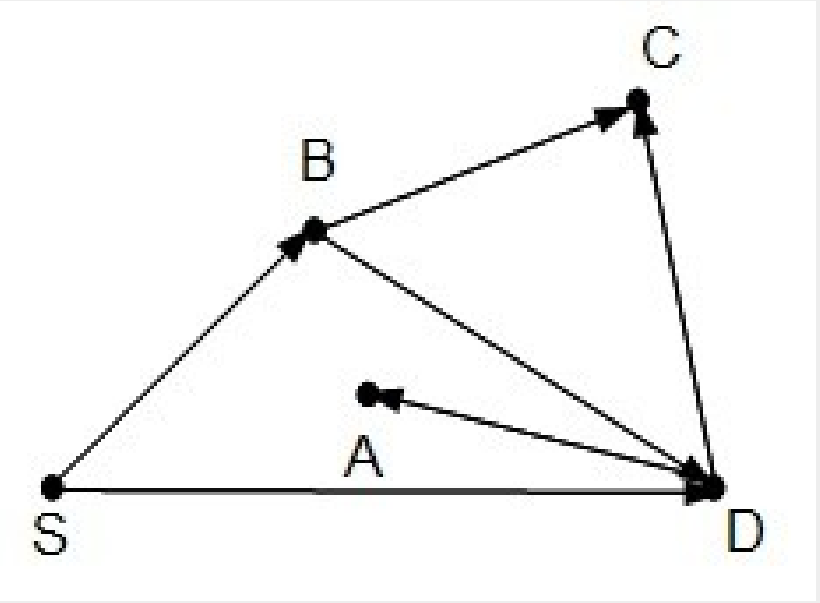


	S	A	B	C	D
par					
st	0	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



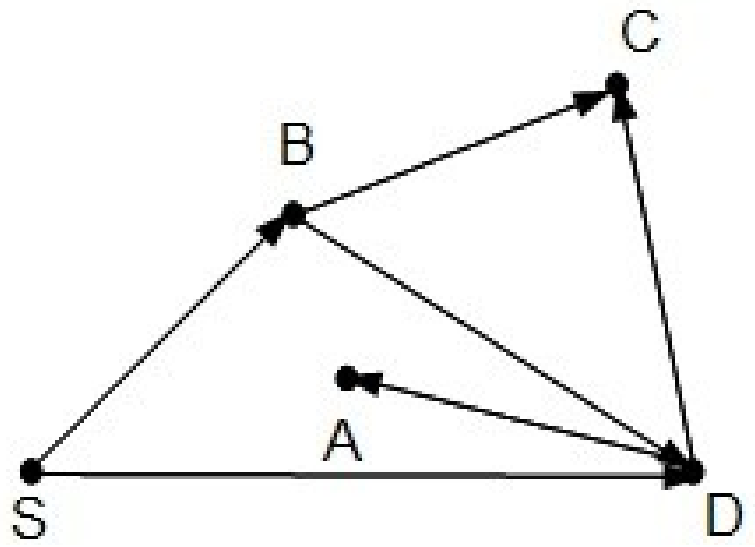
t = 0

	S	A	B	C	D
par					
st	0	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: S

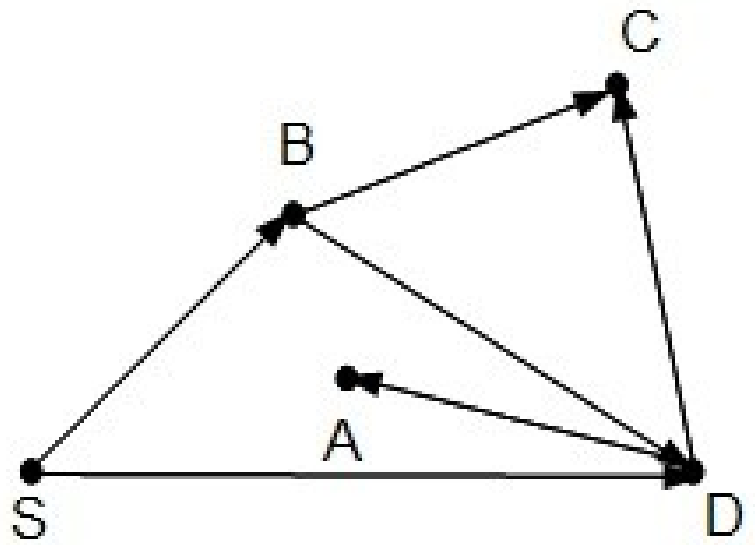
t = 0

	S	A	B	C	D
par					
st	0	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: S's start time is 0

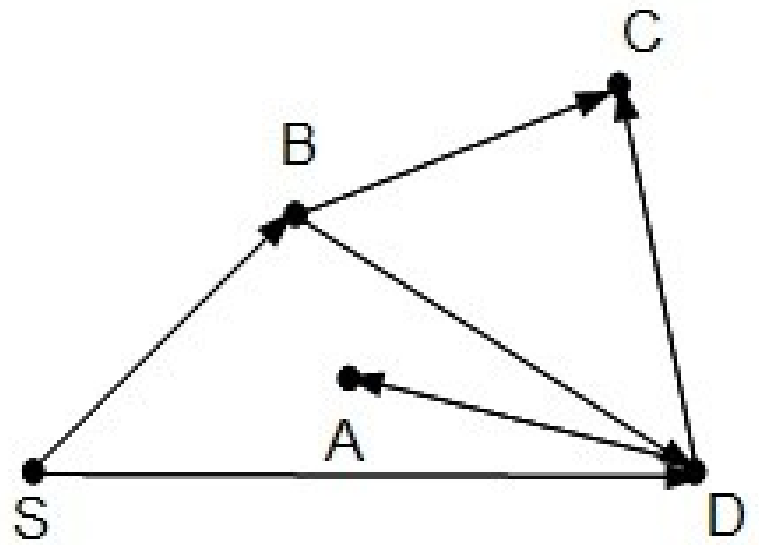
t = 0

	S	A	B	C	D
par					
st	0	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: S

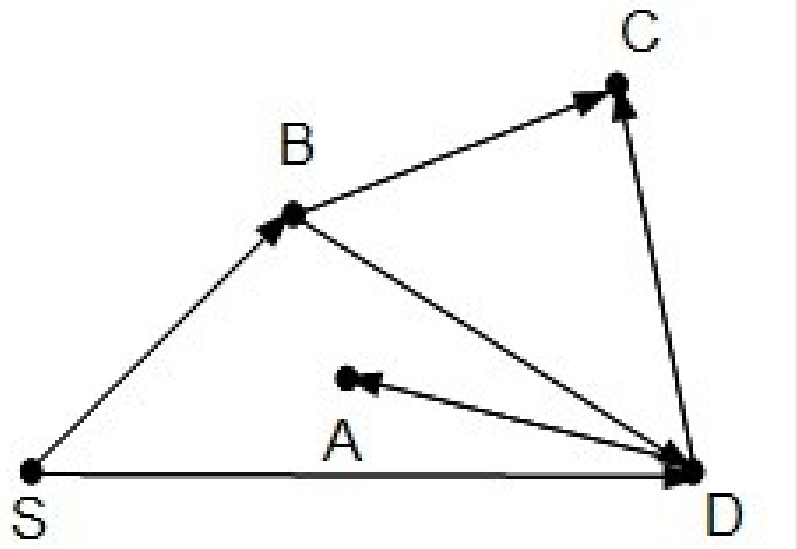
t = 0

	S	A	B	C	D
par					
st	0	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
   for each vertex v in g:
      set v's start time to 0
   t=0
   for each vertex v in g:
      if v's start time is 0:
         dfs_traverse(g,v)

dfs_traverse(g,v):
→  t += 1
   set v's start time to t
   for each vertex u adjacent to v:
      if u's start time is 0:
         set u's parent to v
         dfs_traverse(g,u)
   t += 1
   set v's end time to t
```



v: S

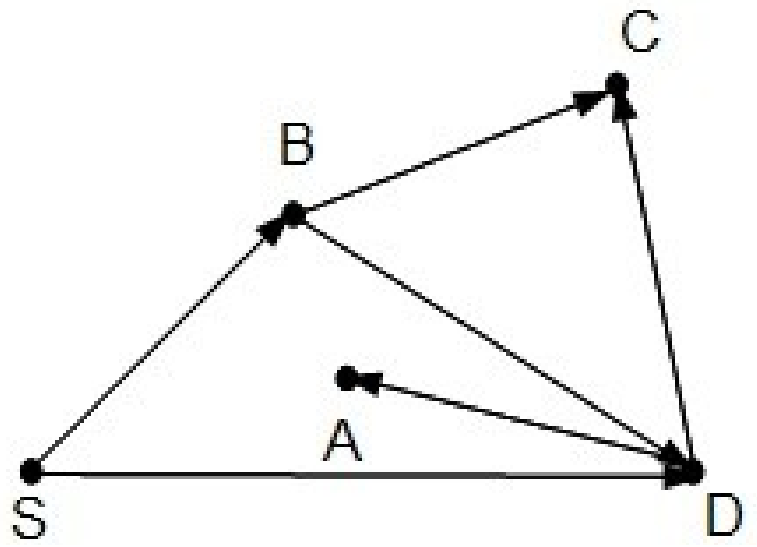
t = 1

	S	A	B	C	D
par					
st	0	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: S

t = 1

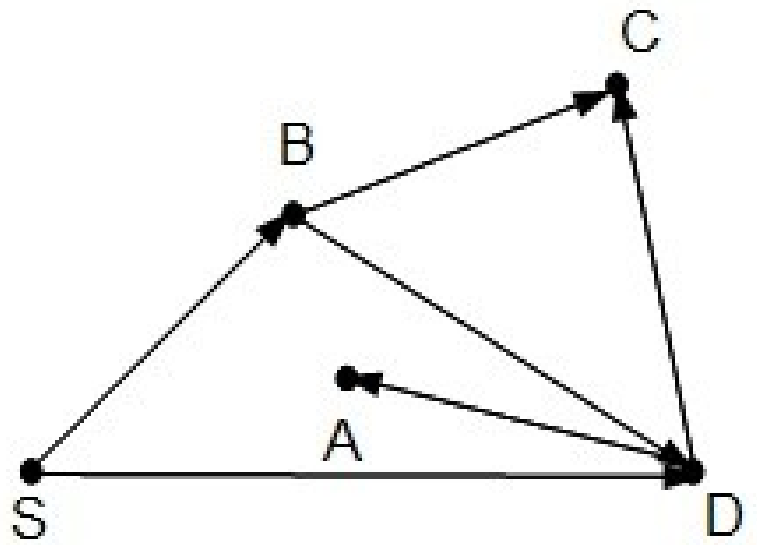
	S	A	B	C	D
par					
st	1	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
→   for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```

t = 1



v: S u: B,D

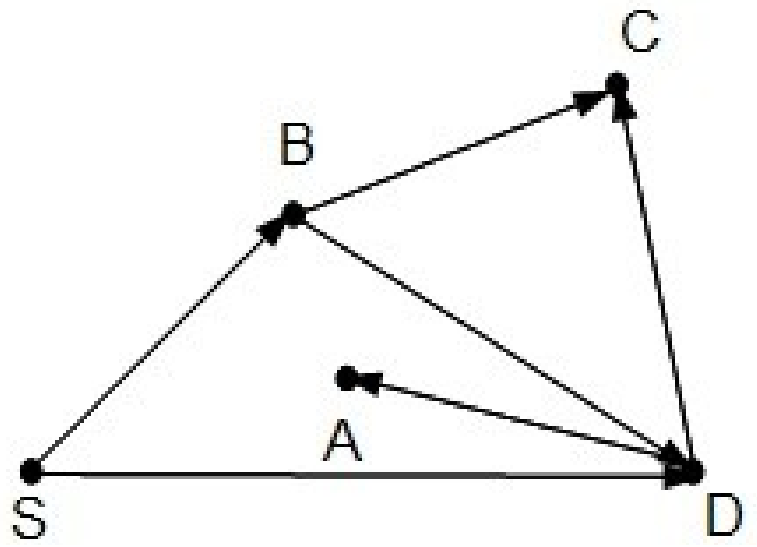
	S	A	B	C	D
par					
st	1	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
   for each vertex v in g:
      set v's start time to 0
   t=0
   for each vertex v in g:
      if v's start time is 0:
         dfs_traverse(g,v)

dfs traverse(g,v):
   t += 1
   set v's start time to t
→  for each vertex u adjacent to v:
      if u's start time is 0:
         set u's parent to v
         dfs_traverse(g,u)
   t += 1
   set v's end time to t
```

t = 1



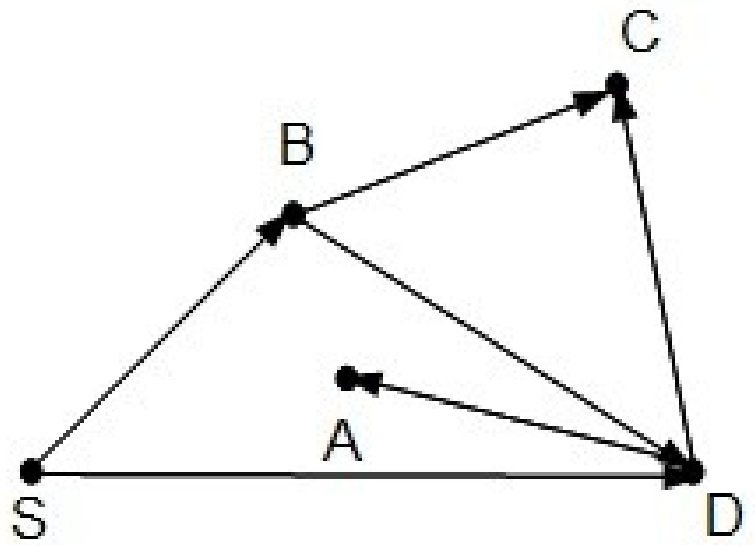
v: S u: B

	S	A	B	C	D
par					
st	1	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: S u: B

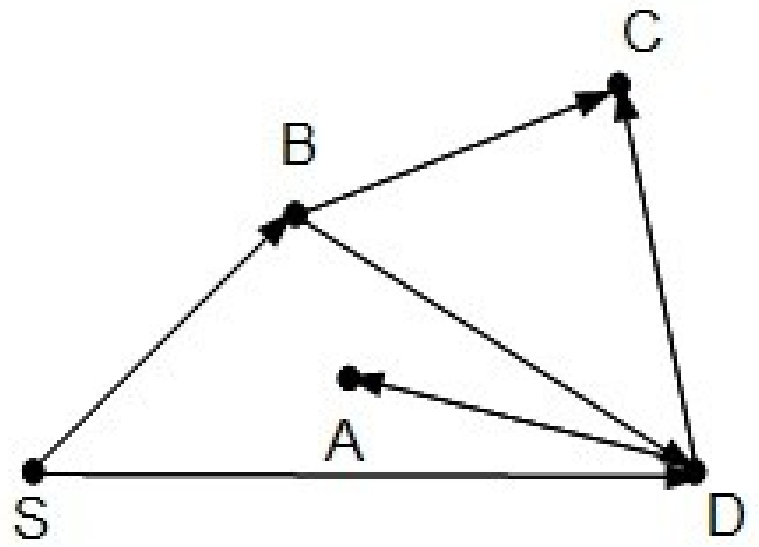
t = 1

	S	A	B	C	D
par					
st	1	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
   for each vertex v in g:
      set v's start time to 0
   t=0
   for each vertex v in g:
      if v's start time is 0:
         dfs_traverse(g,v)

dfs_traverse(g,v):
   t += 1
   set v's start time to t
   for each vertex u adjacent to v:
      if u's start time is 0:
         set u's parent to v
         dfs_traverse(g,u)
   t += 1
   set v's end time to t
```



v: S u: B

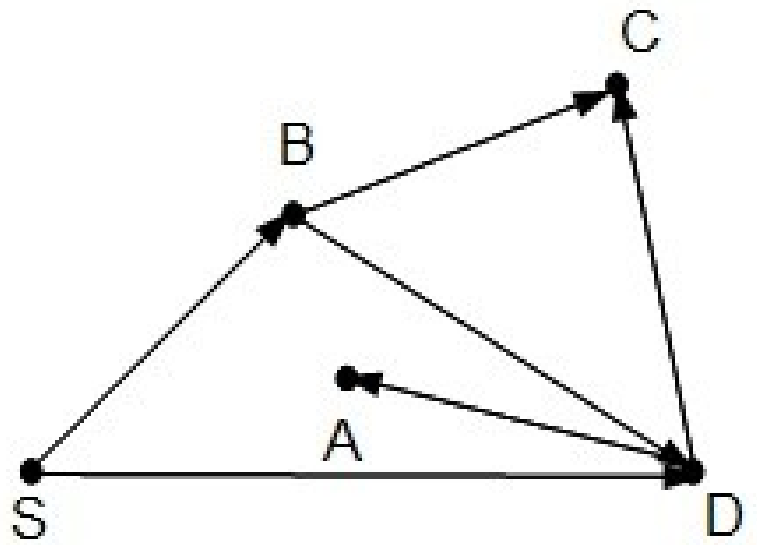
t = 1

	S	A	B	C	D
par			S		
st	1	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u:

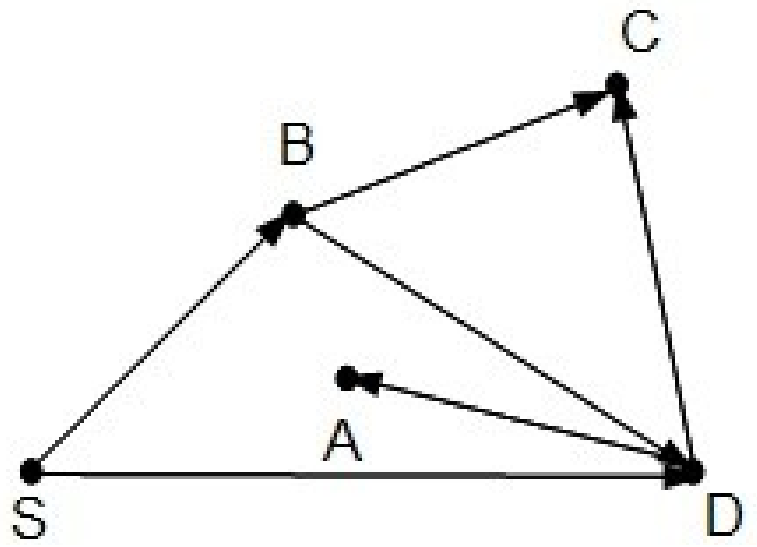
t = 1

	S	A	B	C	D
par			S		
st	1	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u:

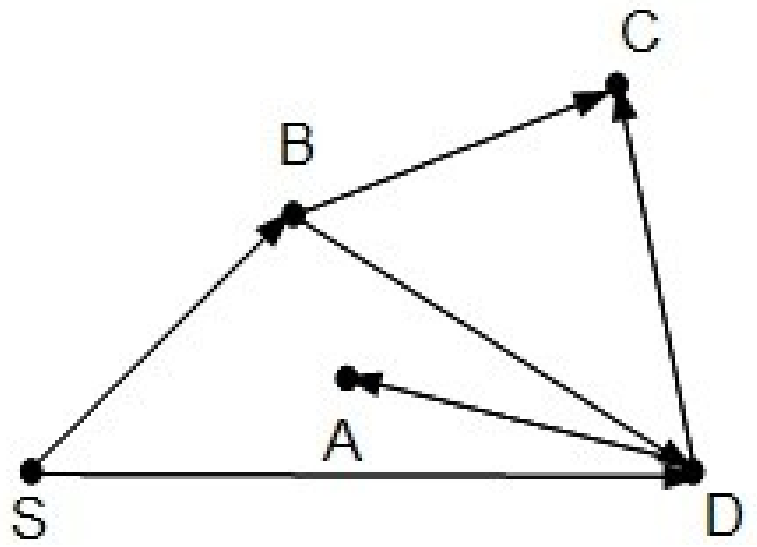
t = 2

	S	A	B	C	D
par			S		
st	1	0	0	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u:

t = 2

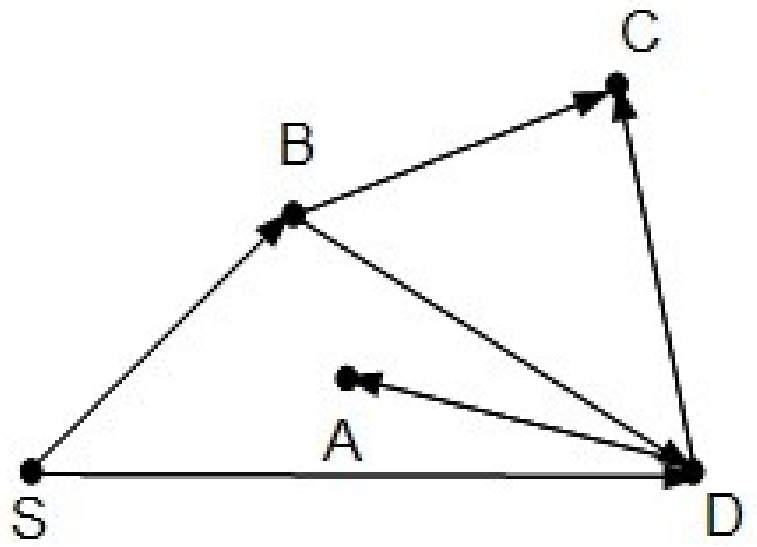
	S	A	B	C	D
par			S		
st	1	0	2	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```

t = 2



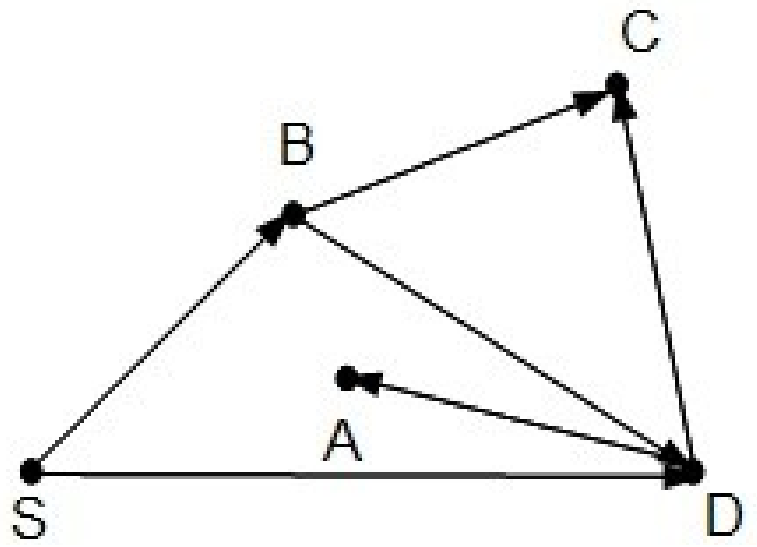
v: B u: C, D

	S	A	B	C	D
par			S		
st	1	0	2	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u: C

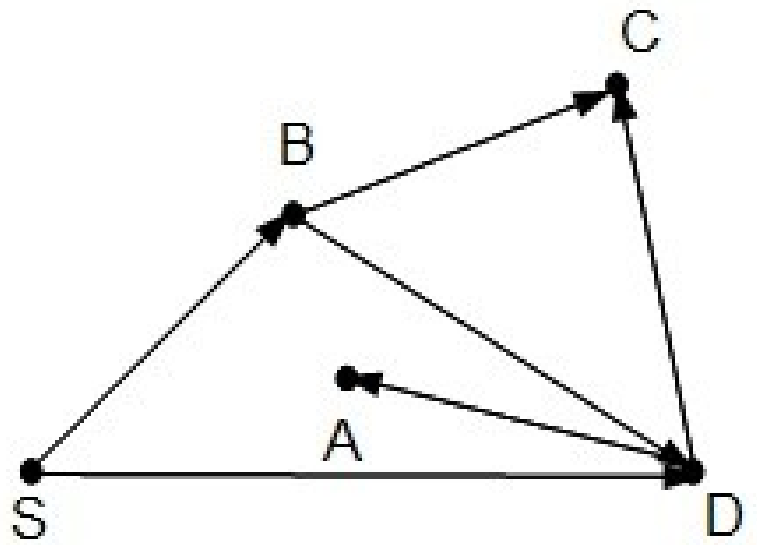
t = 2

	S	A	B	C	D
par			S		
st	1	0	2	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u: C

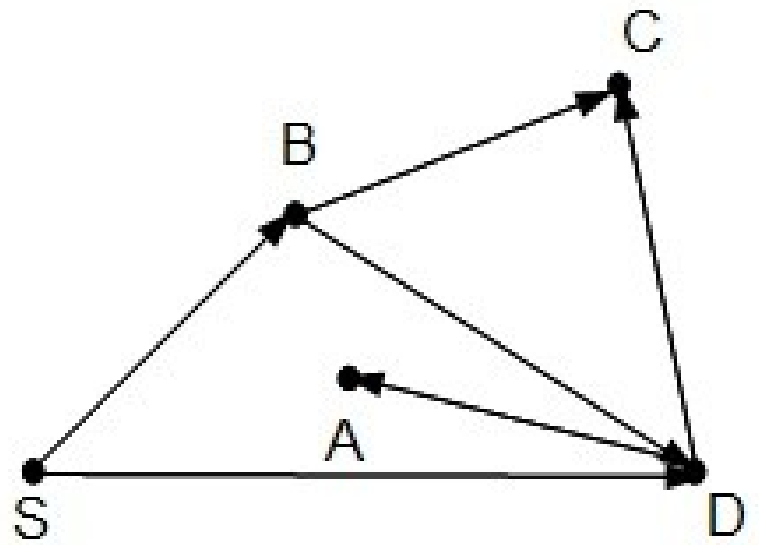
t = 2

	S	A	B	C	D
par			S	B	
st	1	0	2	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: C u:

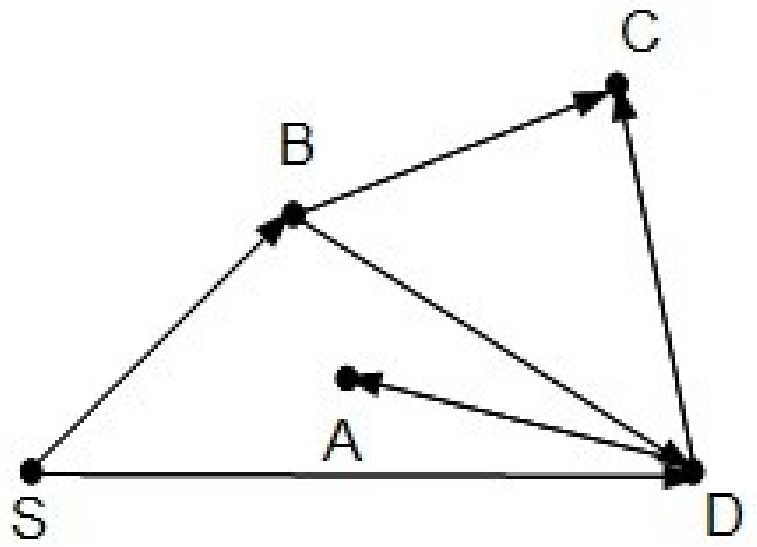
t = 2

	S	A	B	C	D
par			S	B	
st	1	0	2	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: C u:

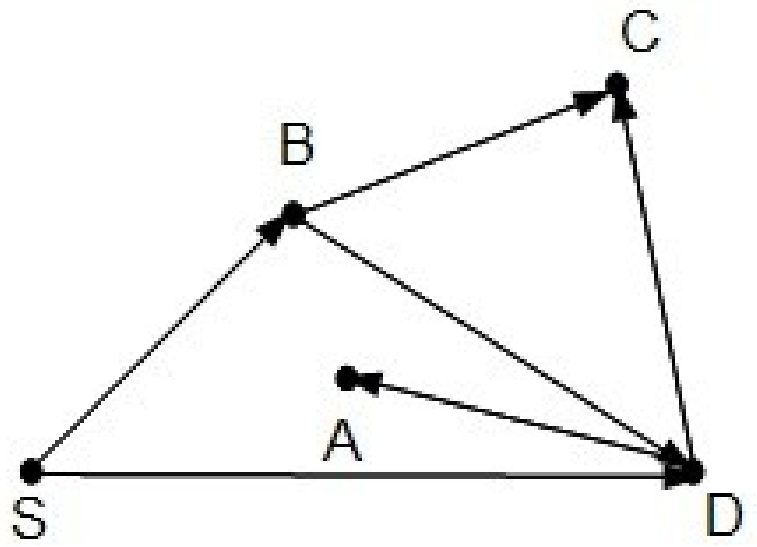
t = 3

	S	A	B	C	D
par			S	B	
st	1	0	2	0	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: C u:

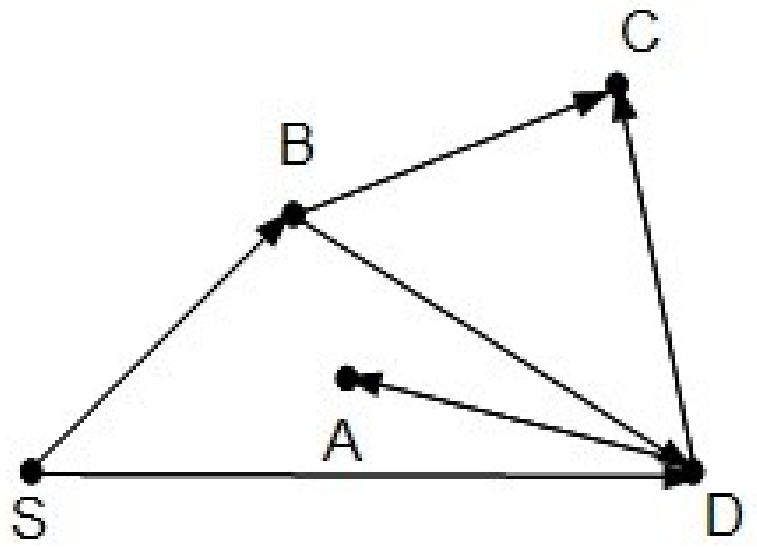
t = 3

	S	A	B	C	D
par			S	B	
st	1	0	2	3	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: C u: none

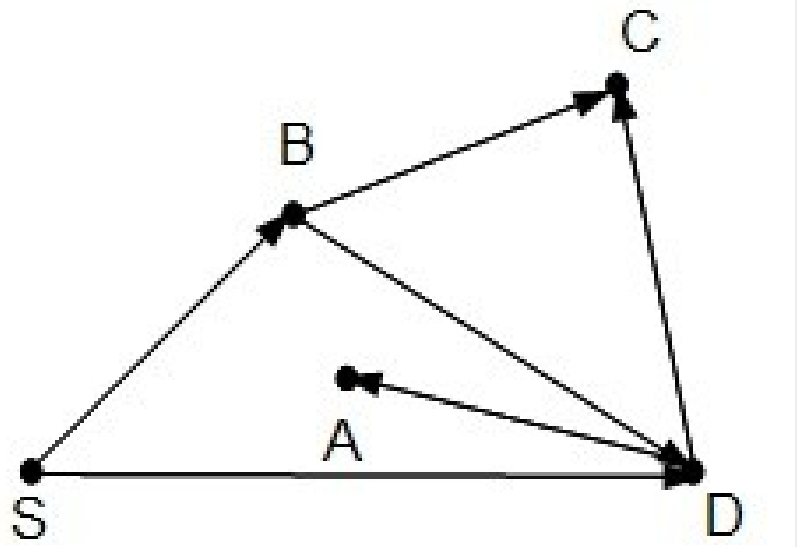
t = 3

	S	A	B	C	D
par			S	B	
st	1	0	2	3	0
et					

Depth First Search algorithm

```
dfs(g):
   for each vertex v in g:
      set v's start time to 0
   t=0
   for each vertex v in g:
      if v's start time is 0:
         dfs_traverse(g,v)

dfs_traverse(g,v):
   t += 1
   set v's start time to t
   for each vertex u adjacent to v:
      if u's start time is 0:
         set u's parent to v
         dfs_traverse(g,u)
   t += 1
   set v's end time to t
```



v: C u: none

t = 4

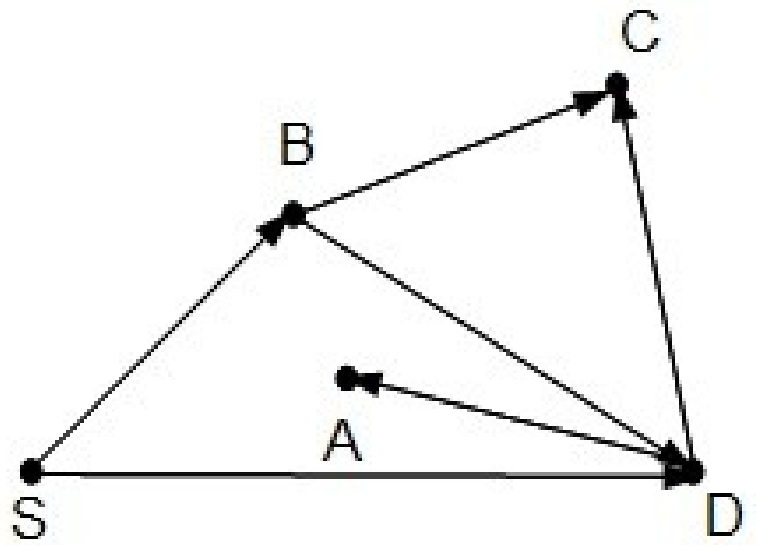
	S	A	B	C	D
par			S	B	
st	1	0	2	3	0
et					

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
→   set v's end time to t
```

t = 4



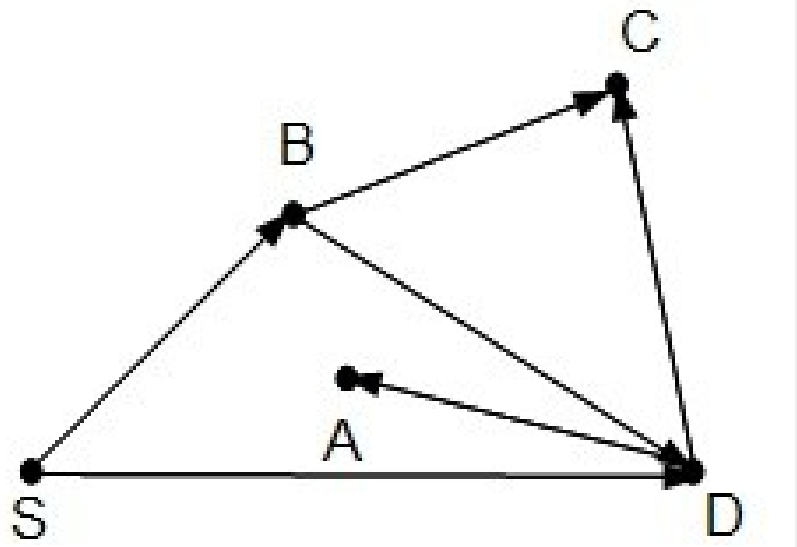
v: C u: none

	S	A	B	C	D
par			S	B	
st	1	0	2	3	0
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u: C is done, D

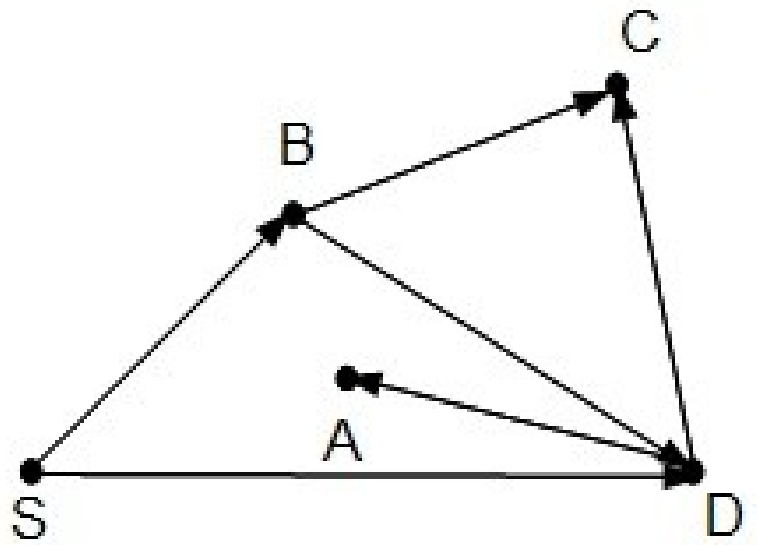
t = 4

	S	A	B	C	D
par			S	B	
st	1	0	2	3	0
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u: D

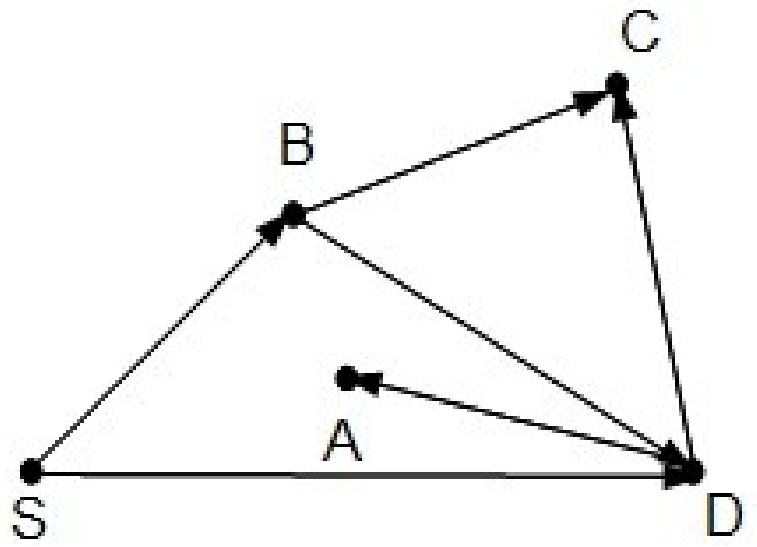
t = 4

	S	A	B	C	D
par			S	B	
st	1	0	2	3	0
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u: D

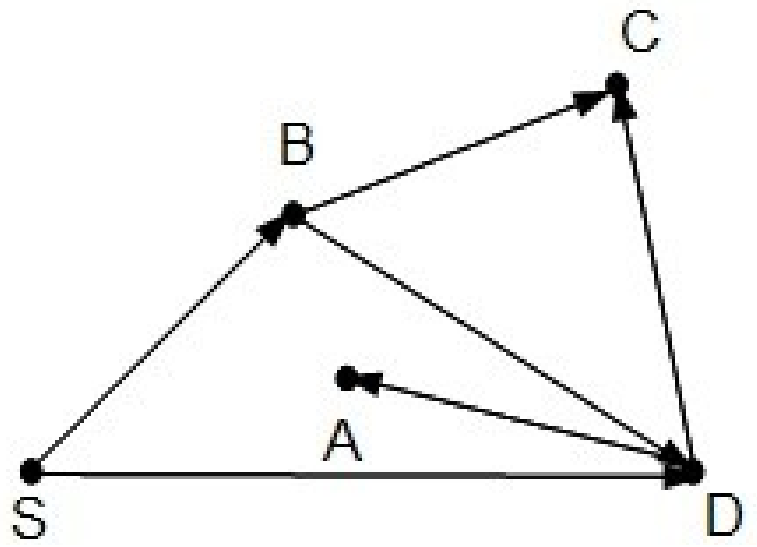
t = 4

	S	A	B	C	D
par			S	B	B
st	1	0	2	3	0
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: D u:

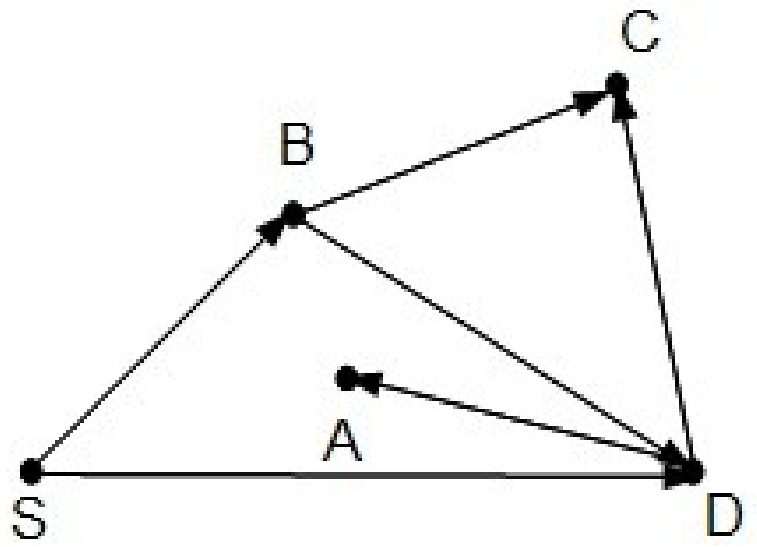
t = 4

	S	A	B	C	D
par			S	B	B
st	1	0	2	3	0
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: D u:

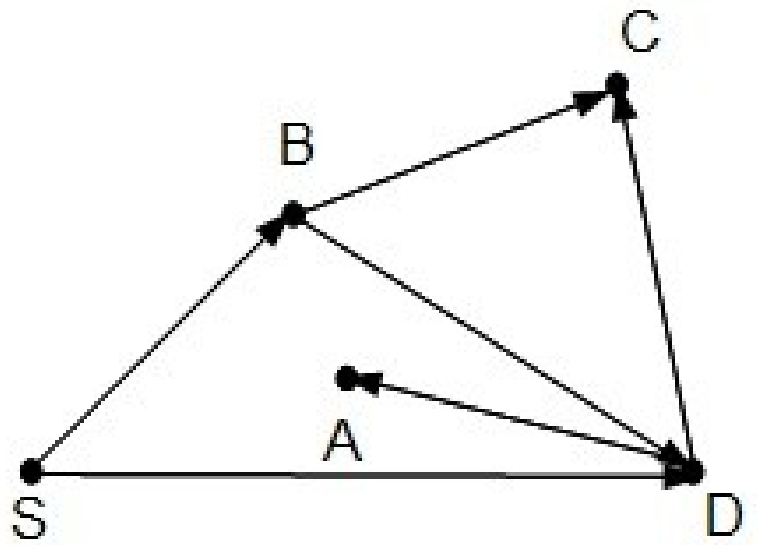
t = 5

	S	A	B	C	D
par			S	B	B
st	1	0	2	3	0
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: D u:

t = 5

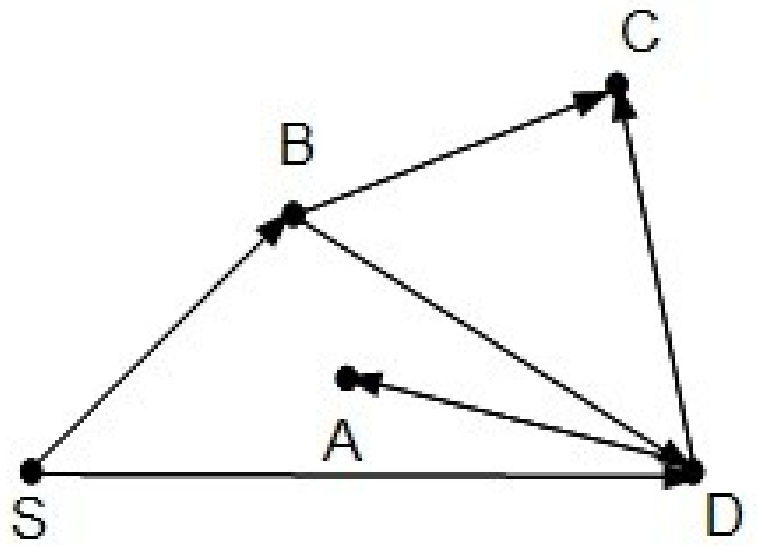
	S	A	B	C	D
par			S	B	B
st	1	0	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```

t = 5



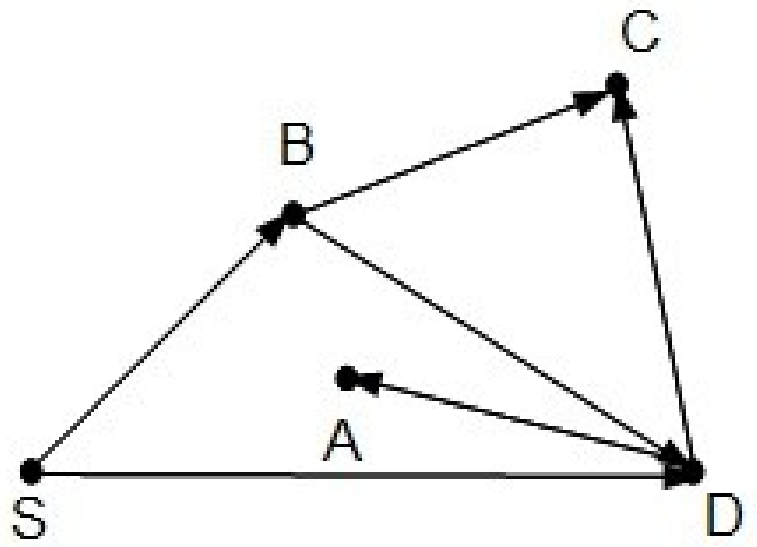
v: D u: A, C

	S	A	B	C	D
par			S	B	B
st	1	0	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: D u: A

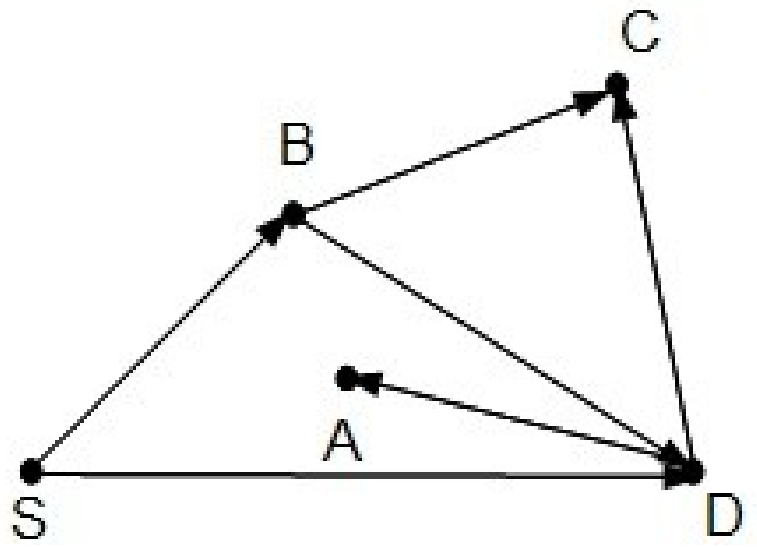
t = 5

	S	A	B	C	D
par			S	B	B
st	1	0	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: D u: A

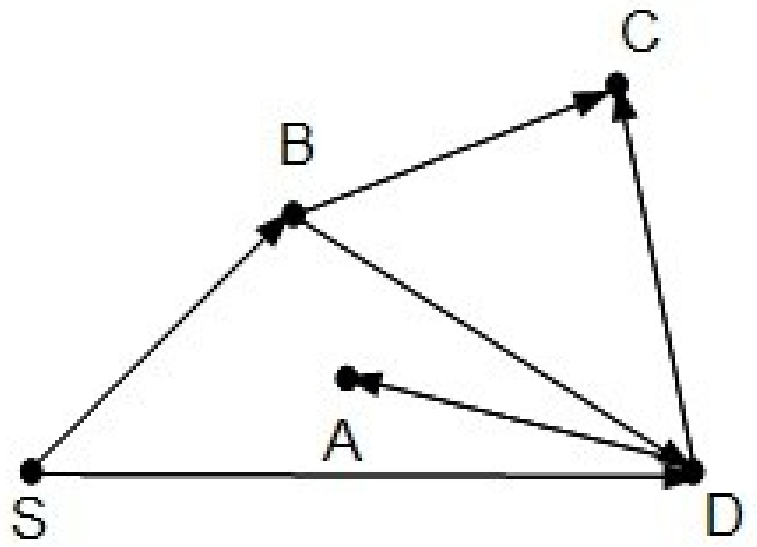
t = 5

	S	A	B	C	D
par			S	B	B
st	1	0	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: D u: A

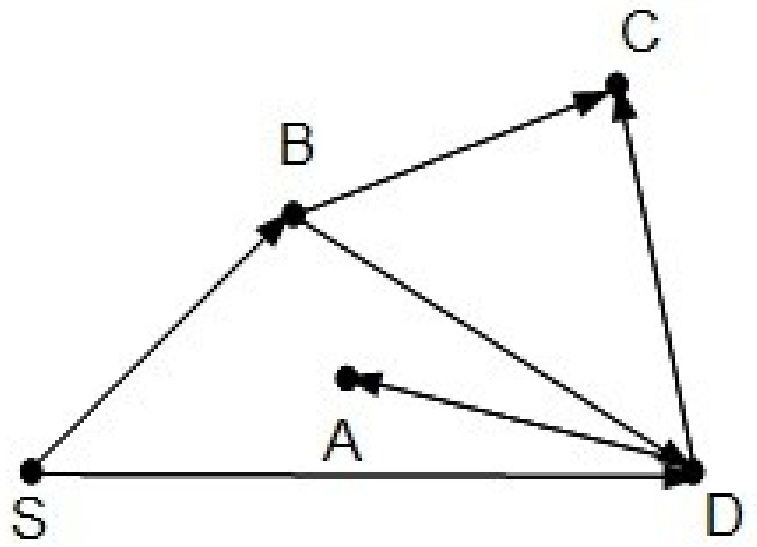
t = 5

	S	A	B	C	D
par		D	S	B	B
st	1	0	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: A u:

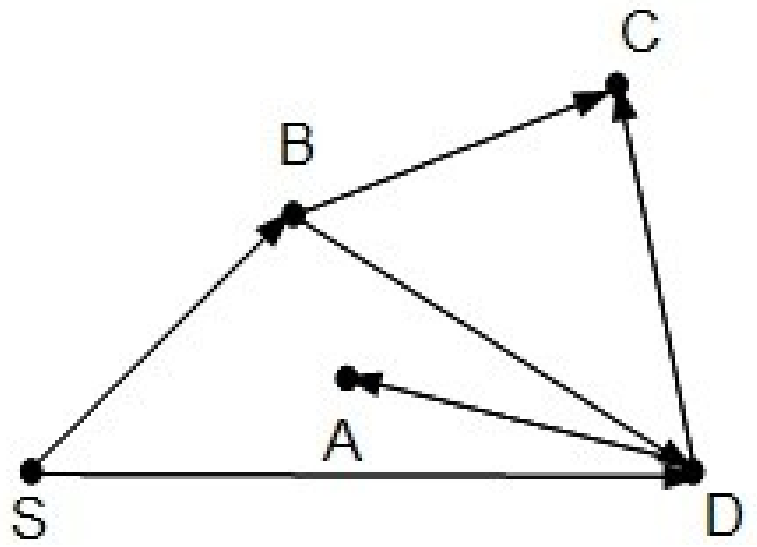
t = 5

	S	A	B	C	D
par		D	S	B	B
st	1	0	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: A u:

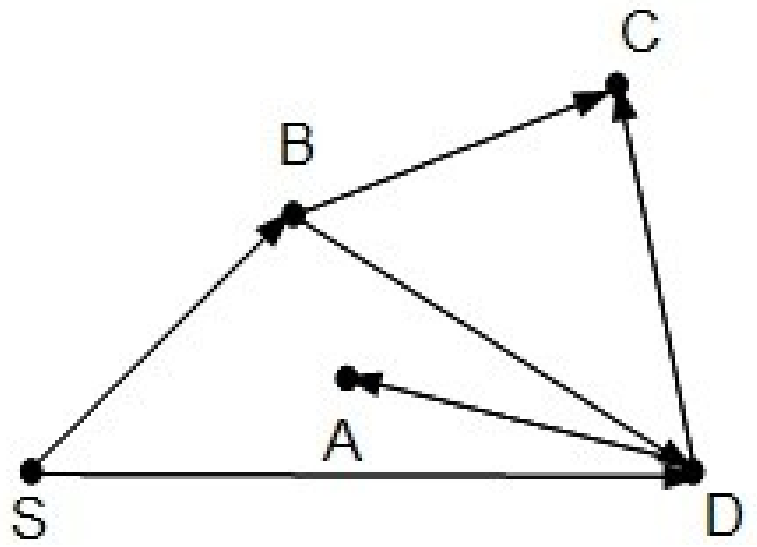
t = 6

	S	A	B	C	D
par		D	S	B	B
st	1	0	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: A u:

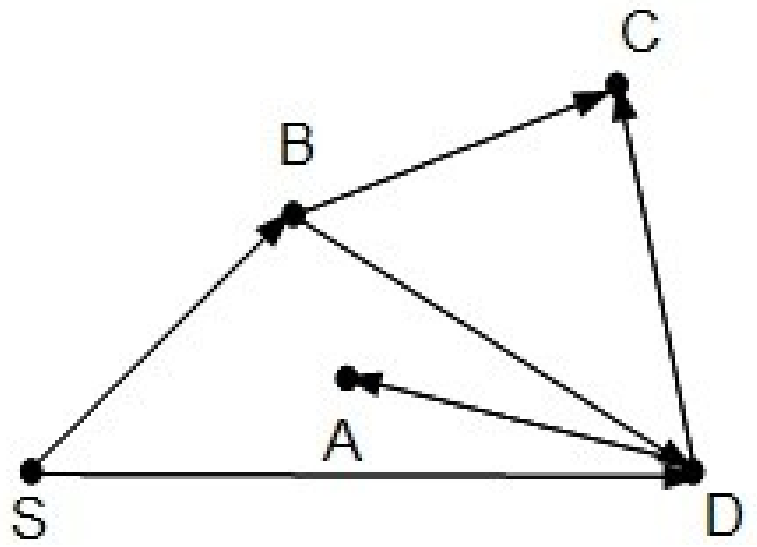
t = 6

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: A u: none

t = 6

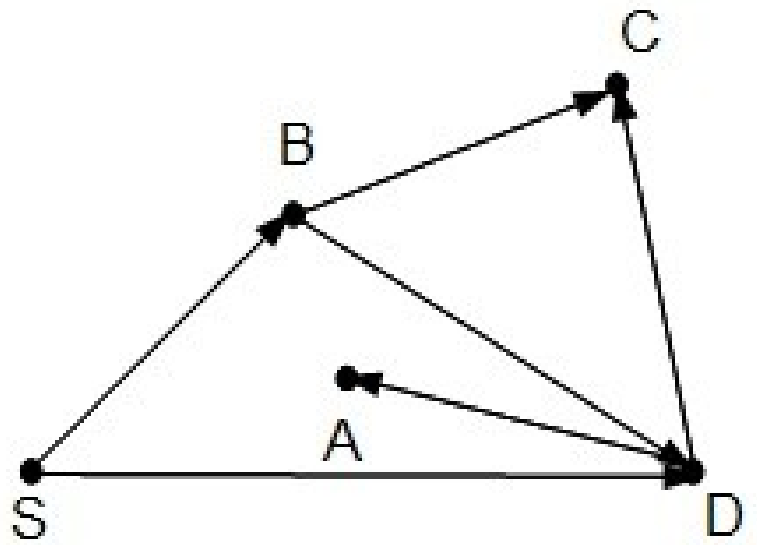
	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
→   t += 1
    set v's end time to t
```

t = 7



v: A u: none

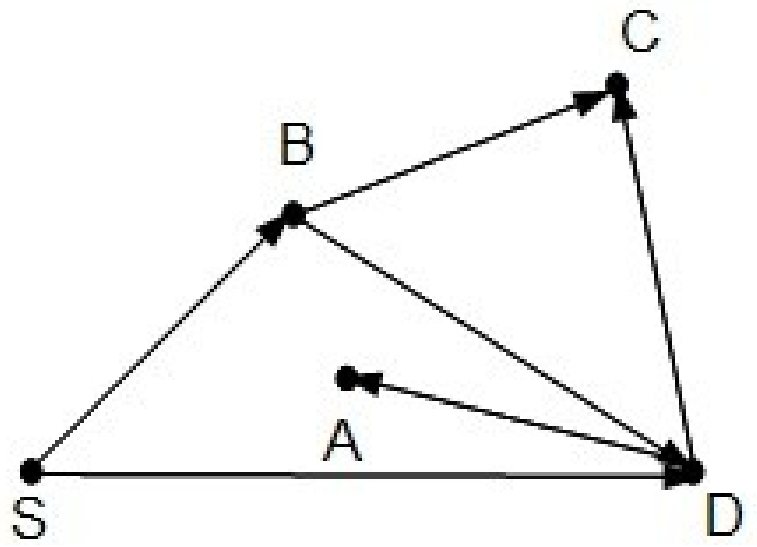
	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et				4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```

t = 7



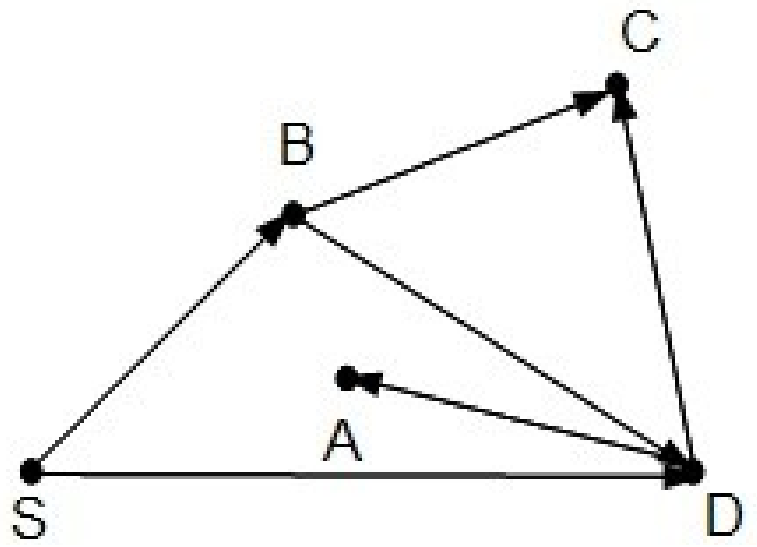
v: A u: none

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7		4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: D u: C

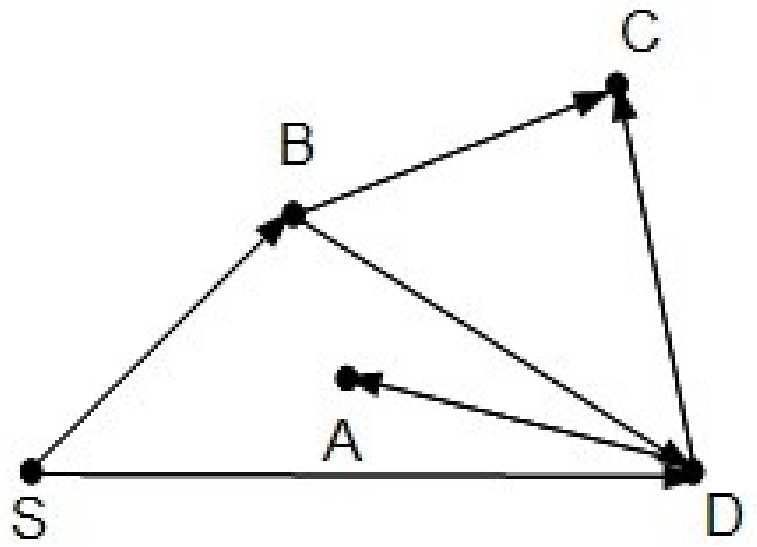
t = 7

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7		4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: D u: C

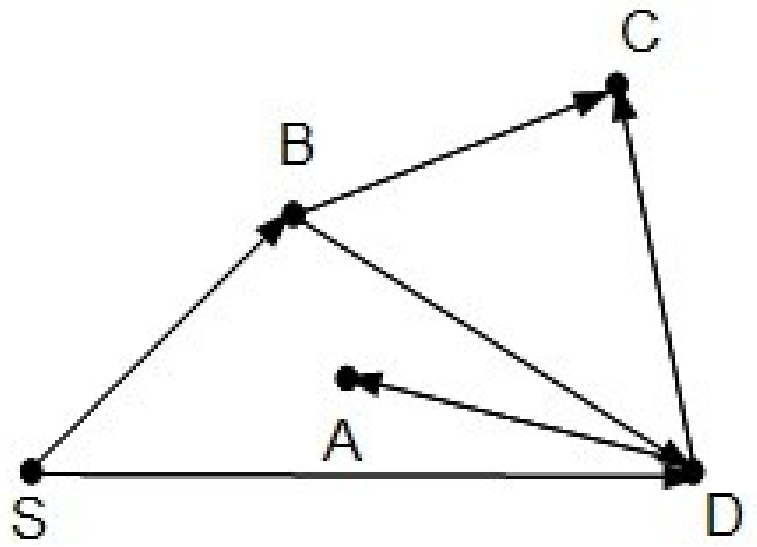
t = 7

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7		4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: D u: none left

t = 7

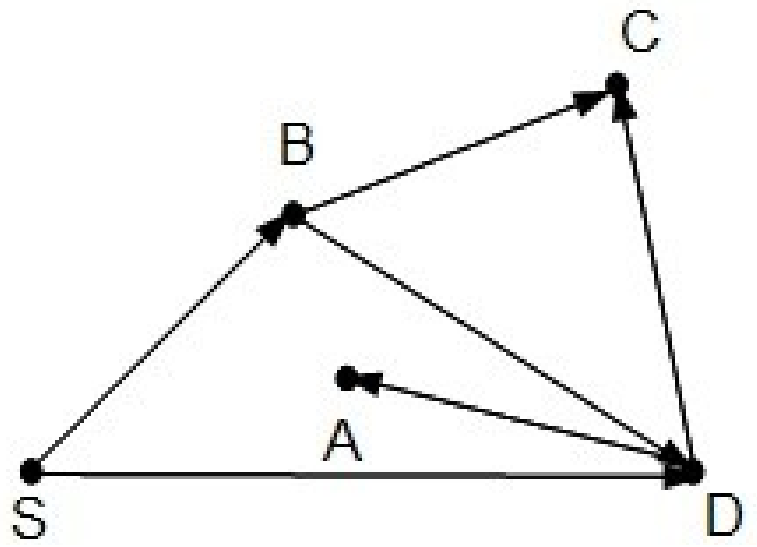
	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7		4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
→   t += 1
    set v's end time to t
```

t = 8



v: D u: none left

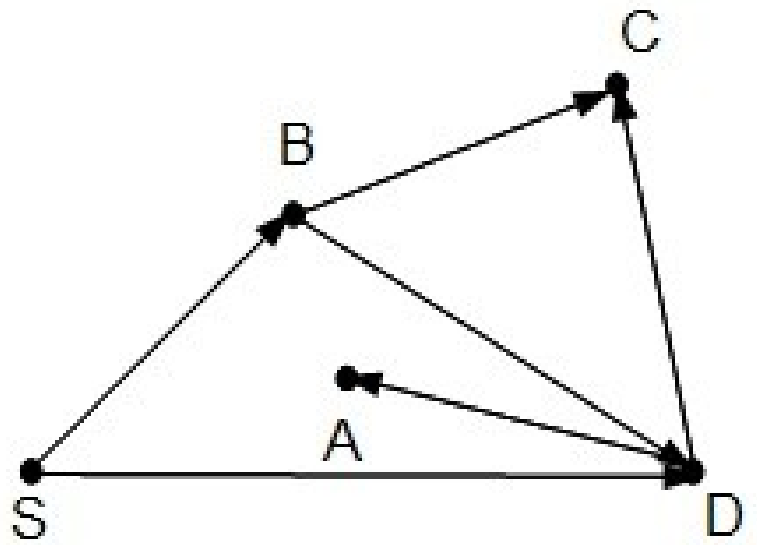
	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7		4	

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
→   set v's end time to t
```

t = 8



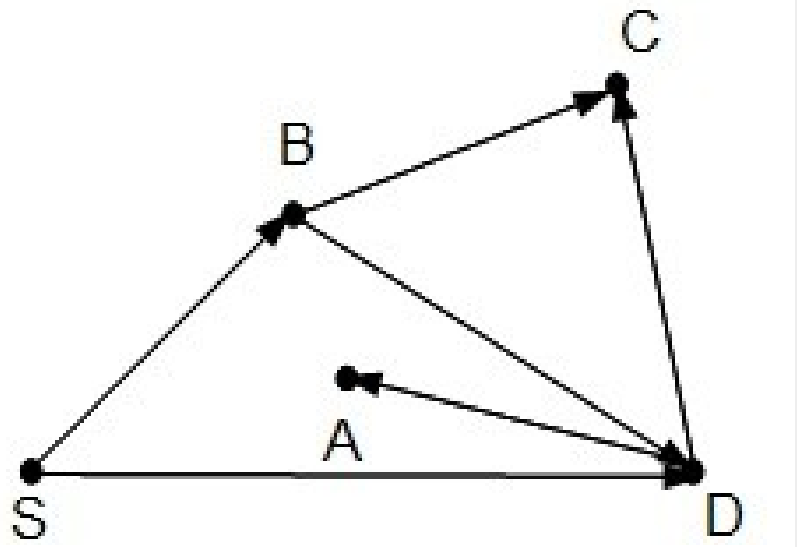
v: D u: none left

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7		4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u: D

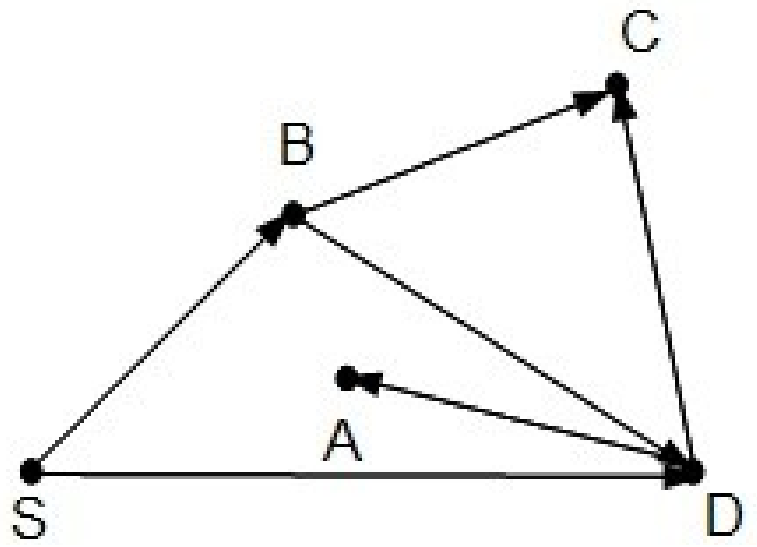
t = 8

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7		4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: B u: none left

t = 8

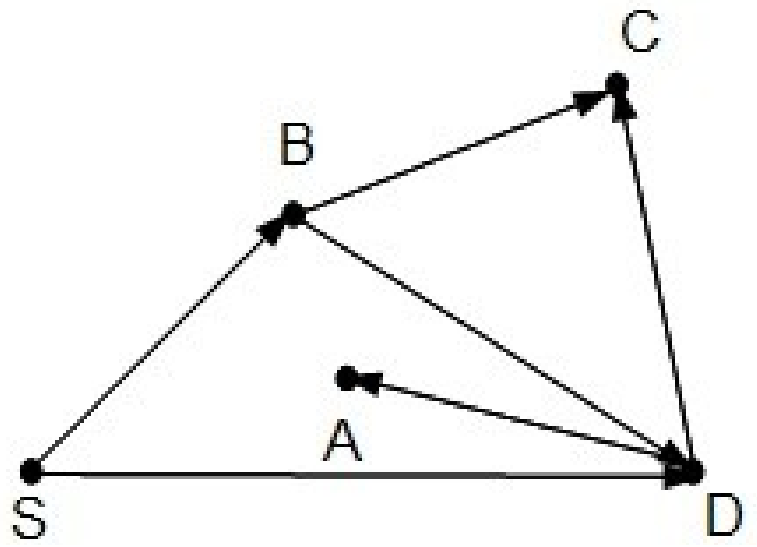
	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7		4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
→   t += 1
    set v's end time to t
```

t = 9



v: B u: none left

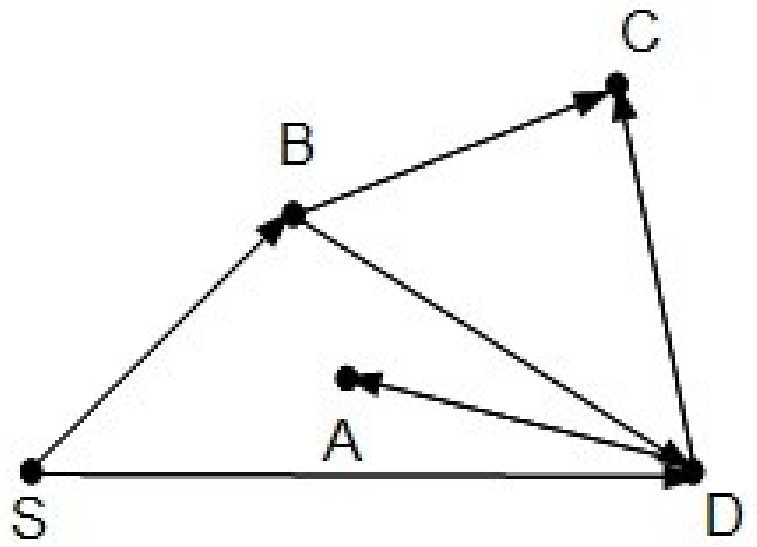
	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7		4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
→   set v's end time to t
```

t = 9



v: B u: none left

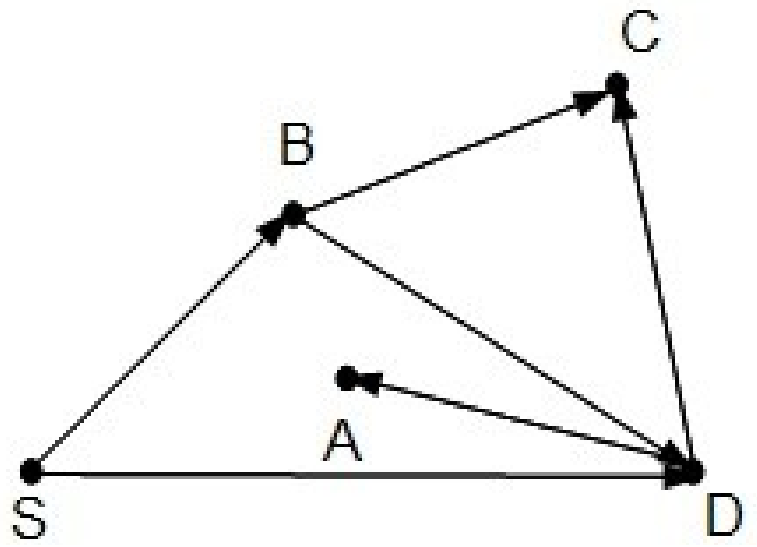
	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7	9	4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```

t = 9



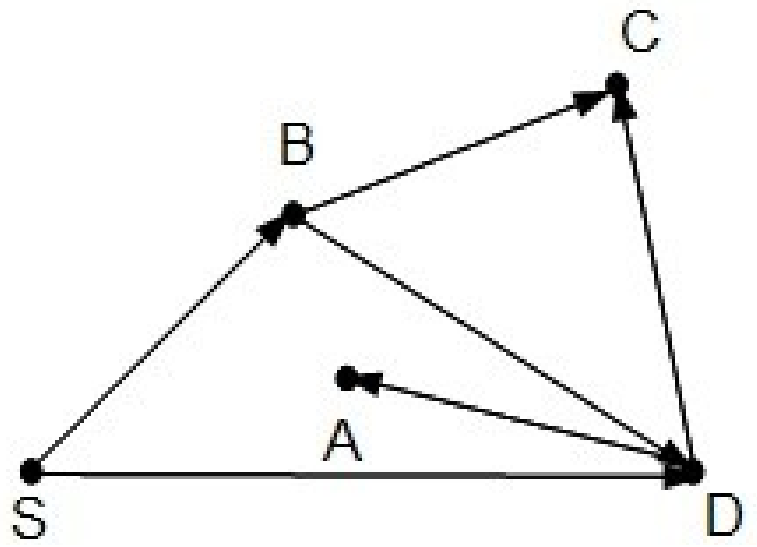
v: S u: B

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7	9	4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: S u: D

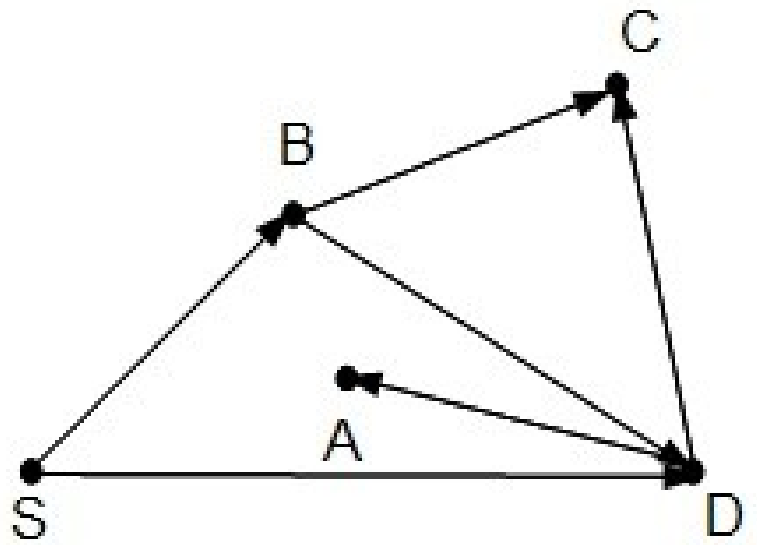
t = 9

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7	9	4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: S u: D

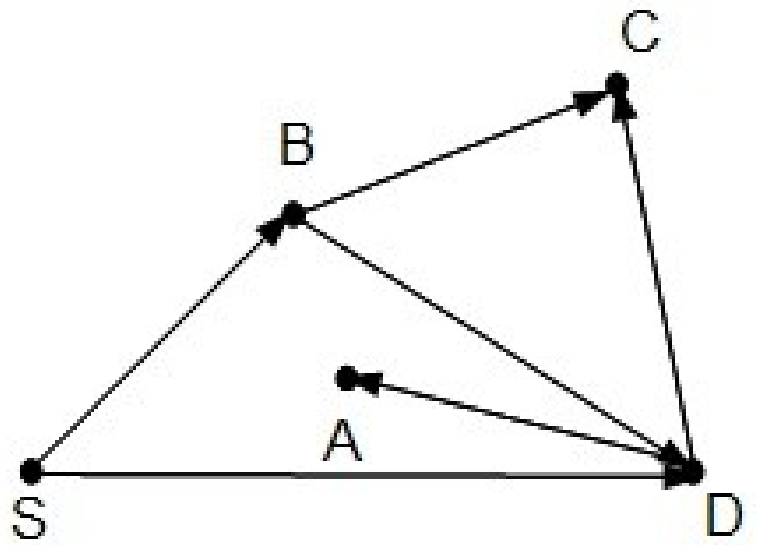
t = 9

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7	9	4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: S u: none left

t = 9

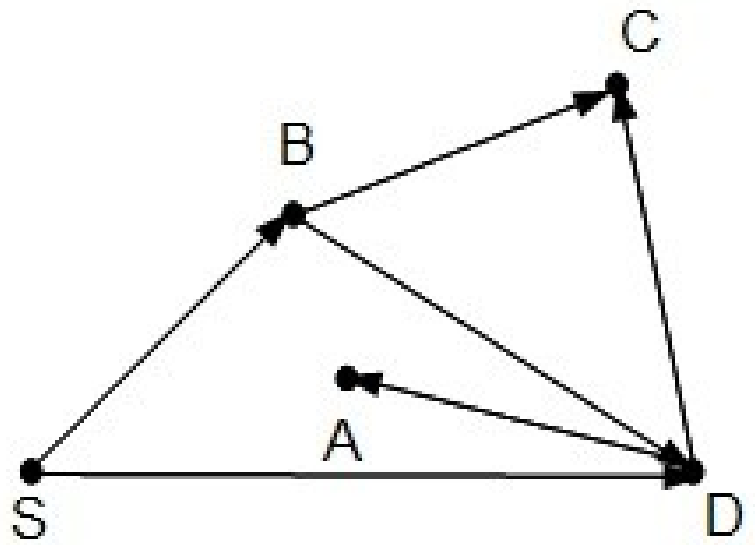
	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7	9	4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
→   t += 1
    set v's end time to t
```

t = 10



v: S u:

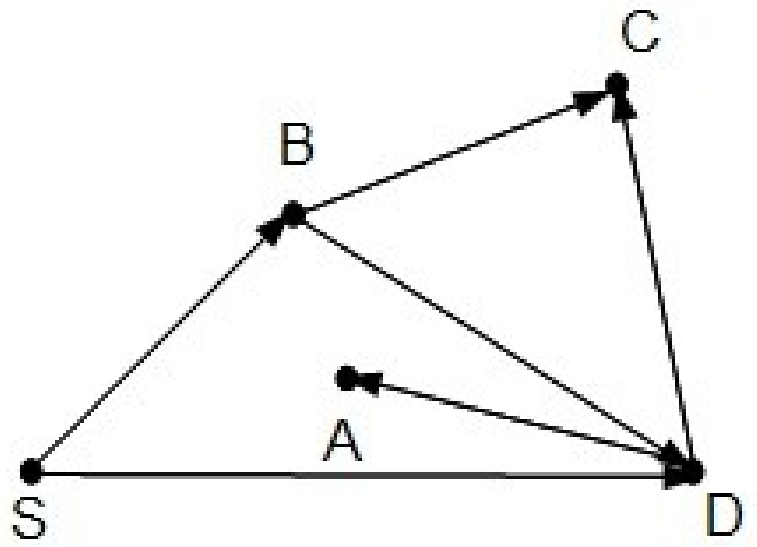
	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et		7	9	4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```

t = 10



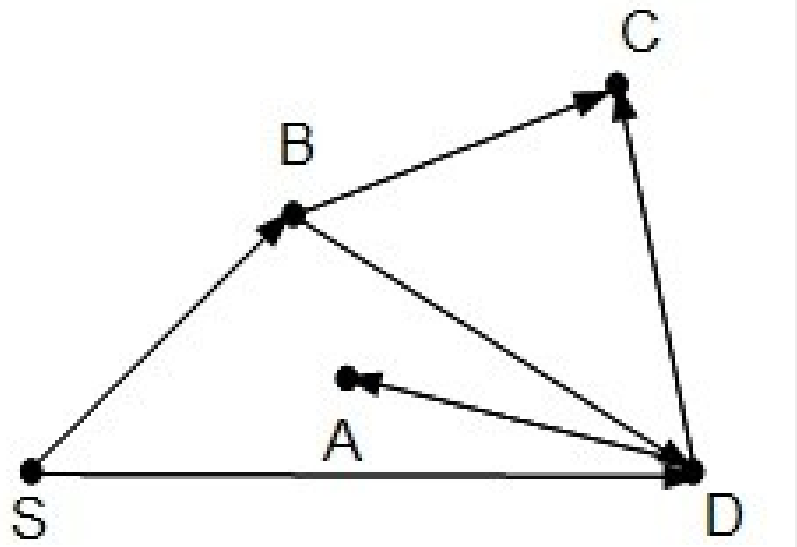
v: S u:

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et	10	7	9	4	8

Depth First Search algorithm

```
dfs(g):
   for each vertex v in g:
      set v's start time to 0
   t=0
   for each vertex v in g:
      if v's start time is 0:
         dfs_traverse(g,v)

dfs_traverse(g,v):
   t += 1
   set v's start time to t
   for each vertex u adjacent to v:
      if u's start time is 0:
         set u's parent to v
         dfs_traverse(g,u)
   t += 1
   set v's end time to t
```



v: S u:

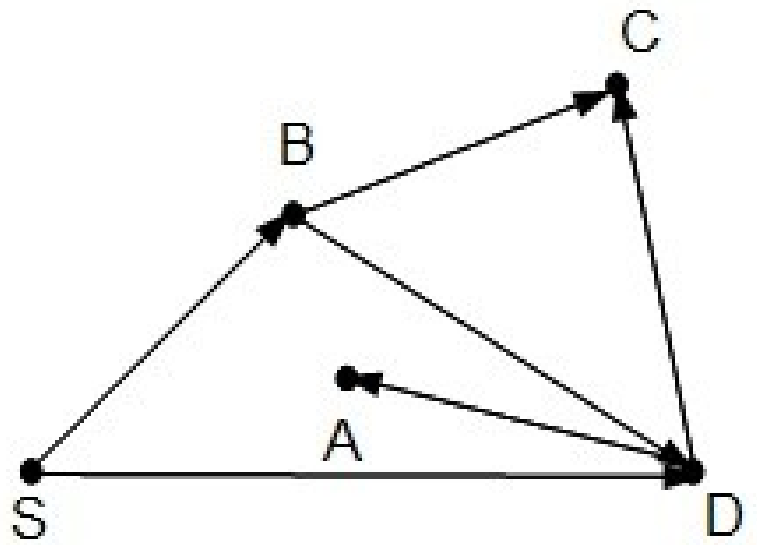
t = 10

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et	10	7	9	4	8

Depth First Search algorithm

```
dfs(g):
    for each vertex v in g:
        set v's start time to 0
    t=0
    for each vertex v in g:
        if v's start time is 0:
            dfs_traverse(g,v)

dfs_traverse(g,v):
    t += 1
    set v's start time to t
    for each vertex u adjacent to v:
        if u's start time is 0:
            set u's parent to v
            dfs_traverse(g,u)
    t += 1
    set v's end time to t
```



v: A,B,C,D

t = 10

	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et	10	7	9	4	8

Depth First Search algorithm

```
dfs(g):
   for each vertex v in g:
      set v's start time to 0
   t=0
   for each vertex v in g:
      if v's start time is 0:
         dfs_traverse(g,v)

dfs_traverse(g,v):
   t += 1
   set v's start time to t
   for each vertex u adjacent to v:
      if u's start time is 0:
         set u's parent to v
         dfs_traverse(g,u)
   t += 1
   set v's end time to t
```

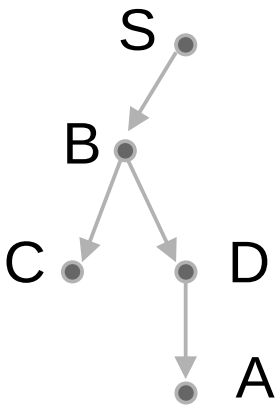
Running time analysis:

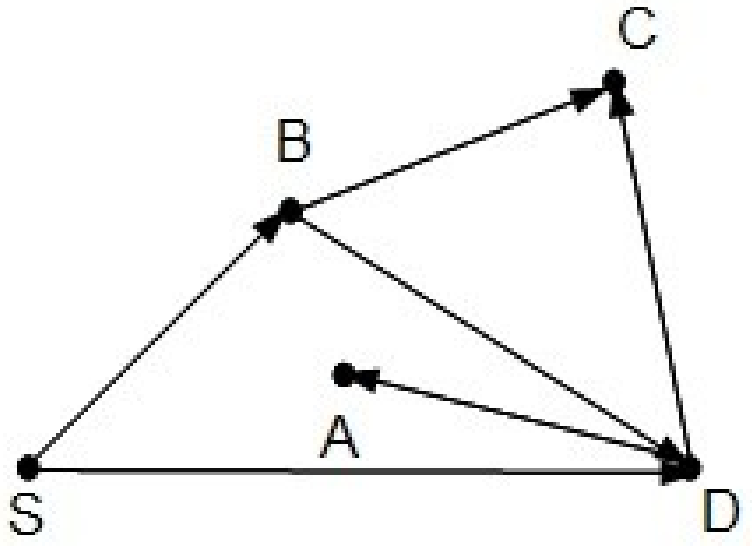
- dfs function processes each vertex in a constant number of times.
- dfs_traverse processes each edge once and performs a constant number of operations as it processes each edge.
- Hence, the overall time is $\Theta(V + E)$.

Depth First Search algorithm

DFS algorithm is similar to tree traversals.

We can view calls from dfs function to dfs_traverse as producing a separate tree.





	S	A	B	C	D
par		D	S	B	B
st	1	6	2	3	5
et	10	7	9	4	8

Depth First Search algorithm

Implementing DFS algorithm

- What container(s) to use to represent graph?
  - adjacency list?
  - adjacency matrix?
- How to represent a table?
- ...

Topological sort

- Self-development:
  - Read about topological sort, using DFS

Minimum Spanning Trees

- A *minimum spanning tree* in a connected weighted undirected graph is a spanning tree that has the smallest possible sum of weights of its edges.
  - no cycles!
  - the minimum spanning tree for a graph with V vertices should have V-1 edges

Minimum Spanning Trees

- A *minimum spanning tree* in a connected weighted undirected graph is a spanning tree that has the smallest possible sum of weights of its edges.
  - no cycles!
  - the minimum spanning tree for a graph with V vertices should have V-1 edges
  - Kruskal's algorithm (1956)
  - Prim's algorithm (1957)

Kruskal's algorithm

procedure *Kruskal*(*G*: weighted connected undirected graph with *n* vertices)
T:= empty graph
sort edges by weight (n edges)
for *i* := *1* to *n-1*
    e:= any edge in *G* with smallest weight that does not form a cycle when added to T
    add *e* to T
return T

Kruskal's algorithm

procedure *Kruskal*(*G*: weighted connected undirected graph with *n* vertices)
T:= empty graph
sort edges by weight (n edges)
for *i* := *1* to *n-1*
    e:= any edge in *G* with smallest weight that does not form a cycle when added to T
    add *e* to T
return T

difficult to implement;
We can use DFS algorithm to determine if an undirected graph has a cycle (think about it!)

Kruskal's algorithm

procedure *Kruskal*(*G*: weighted connected undirected graph with *n* vertices)
T:= empty graph
sort edges by weight (n edges)
for *i* := *1* to *n-1*
    e:= any edge in *G* with smallest weight that does not form a cycle when added to T
    add *e* to T
return T

difficult to implement;
Textbook explores a data structure known as `disjoint set`

Disjoint Set

```
class DisjointSet :

def __init__(self):
    self.sets = {}

def make_set(self, x):
'''post: adds a set to the group of sets for the single
element x; raises KeyError if already a set containing x'''

    # check if set for this item already exists
    if x in self.sets:
        raise KeyError(f"{x} already in DisjointSet")

    # map element to the set/list containing it
    self.sets[x] = [x]
```

DisjointSet data structure is a group of sets that do not contain any elements in common

Disjoint Set

```
def find(self, x):
'''post : returns set/list containing x
raises KeyError if there is not a set containing x;
for efficiency use the "is" operator to determine if
two elements are in the same set by making two calls
to find
(e.g., if dj.find(x) is dj.find(y): )'''

return self.sets[x]
```

Union method

- The union(x,y) method joins the set that contains x with the set that contains y
- the precondition for the union method is that the two parameters are not in the same set .
- The union method decreases the number of sets in the group by one.

Union method

```
def union( self, x, y) :
    '''post: the sets containing x and y are merged/joined
    raises KeyError if the two sets are already the same'''

    if self.sets[x] is self.sets[y]:
        raise KeyError(f"{x} and {y} are in the same set")
```

Union method

```
#determine smaller list, to add fewer items to the
# existing list
if len(self.sets[x]) > len(self.sets[y] ) :
   # save list of elements in smaller set
   temp = self.sets [y]
   # for each element in smaller set,
   # map it to the larger list
   for k in self.sets[y]:
      self.sets[k] = self.sets[x]
      # add all elements in smaller set/list to larger one
      self.sets[x].extend (temp)
```

Union method

```
#if len(self.sets[x]) <= len(self.sets[y] )
else:
    #save elements in smaller set
    temp = self.sets[x]
    # for each element in smaller set ,
    # map it to the larger list
    for k in self.sets[x] :
        self.sets[k] = self.sets [y]
        # add all elements in smaller set /list to larger one
        self.sets[y].extend(temp)
```

Kruskal's algorithm

- Form a set for each vertex
  - Disjoint set will have V sets, with one element
- Checking edge:
  - if the two vertices are in the same set, do not add
  - Otherwise, add the edge and join the two sets
- As we do this, each set corresponds to the vertices that are connected by the edges we have added
- Continue until we have one set with V elements