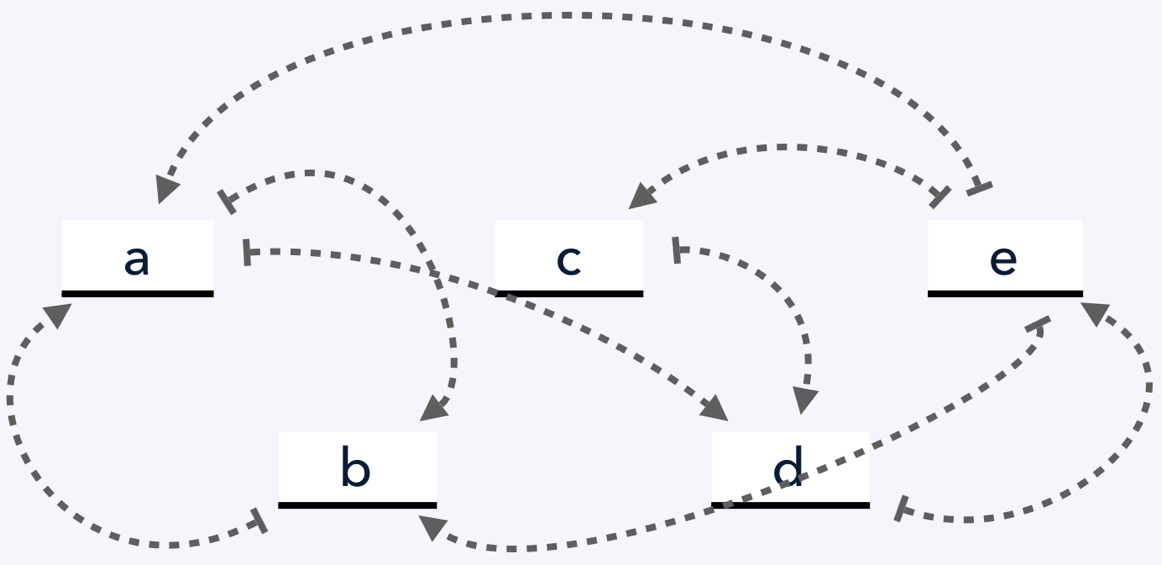
1. 1.度和 = 2 \* 边和

2.入度和 = 出度和

3.n – 1，n \* n

4.n，若两点i，j存在i->j，j->i路径，则两点在同一个环上，而小于n个弧不能成环，n个弧若首尾相连则可行

1. 1.

入度 出度

a 2 2

b 2 1

c 1 1

d 2 1

e 1 3

2.

邻接表

1 a->2->4->^

2 b->1->^

3 c->2->4->^

4 d->5->^

5 e->1->2->3->^

逆邻接表

1. a->2->5->^
2. b->1->5->^
3. c->5->^
4. d->1->3->^
5. e->4->^
6. 1.

邻接矩阵

1 2 3 4 5 6

1 0 9 6 3 inf inf

2 9 0 inf 5 8 inf

3 6 inf 0 2 9 5

4 3 5 2 0 inf 7

5 inf 8 9 inf 0 4

6 inf inf 5 7 4 0

2.

边表

value 9 6 3 5 8 2 9 5 7 4

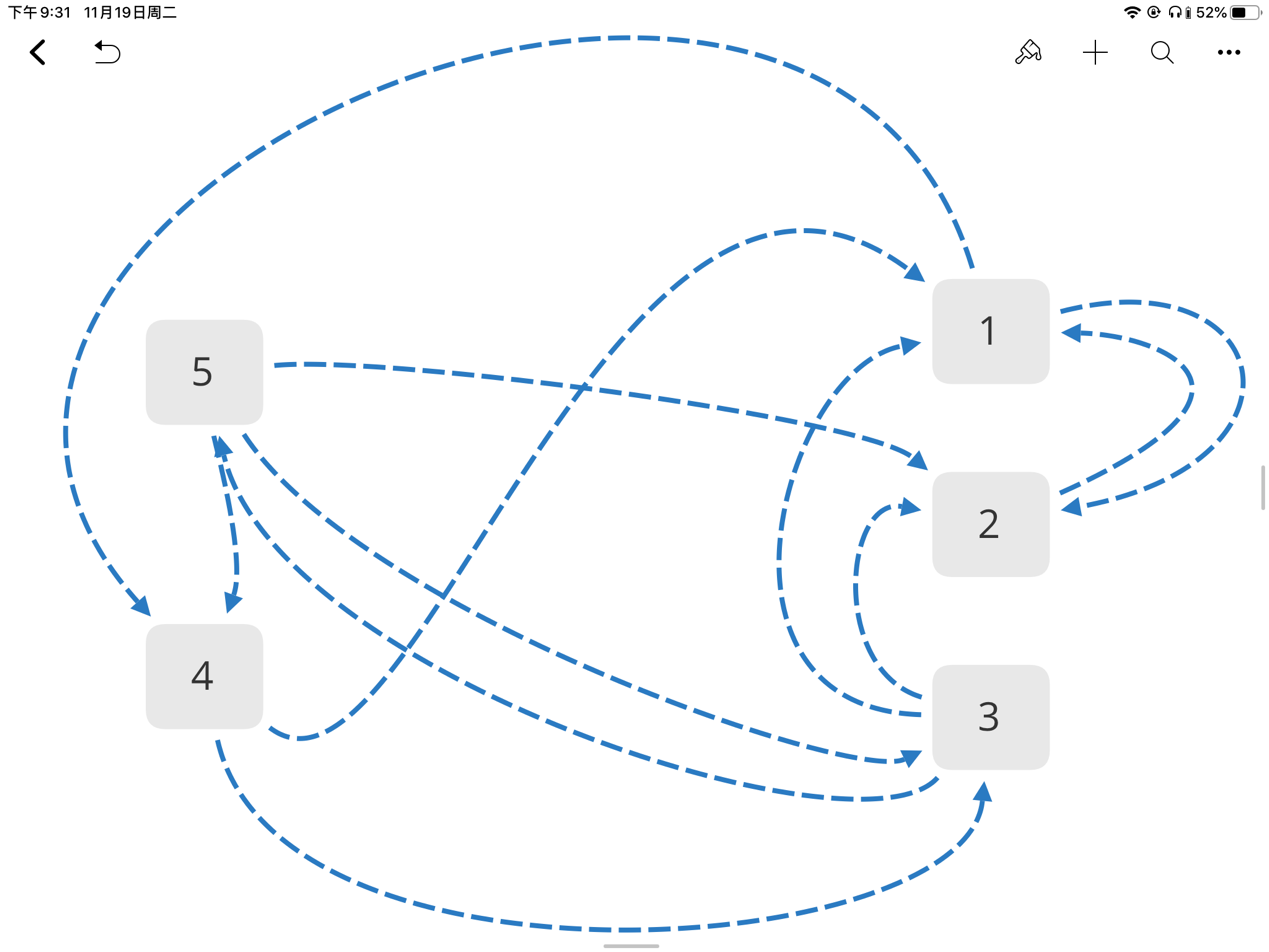
vertex1 1 1 1 2 2 3 3 3 4 5

vertex2 2 3 4 4 5 4 5 6 6 6

3.

v 1 2 3 4 5 6

度 3 3 4 4 3 3

1. 1.

有向图

2.

邻接矩阵（0代表无，1代表有

u1 2 3 4 5

v1 0 1 1 1 0

2 1 0 1 0 1

3 0 0 0 1 1

4 1 0 0 0 1

5 0 0 1 0 0

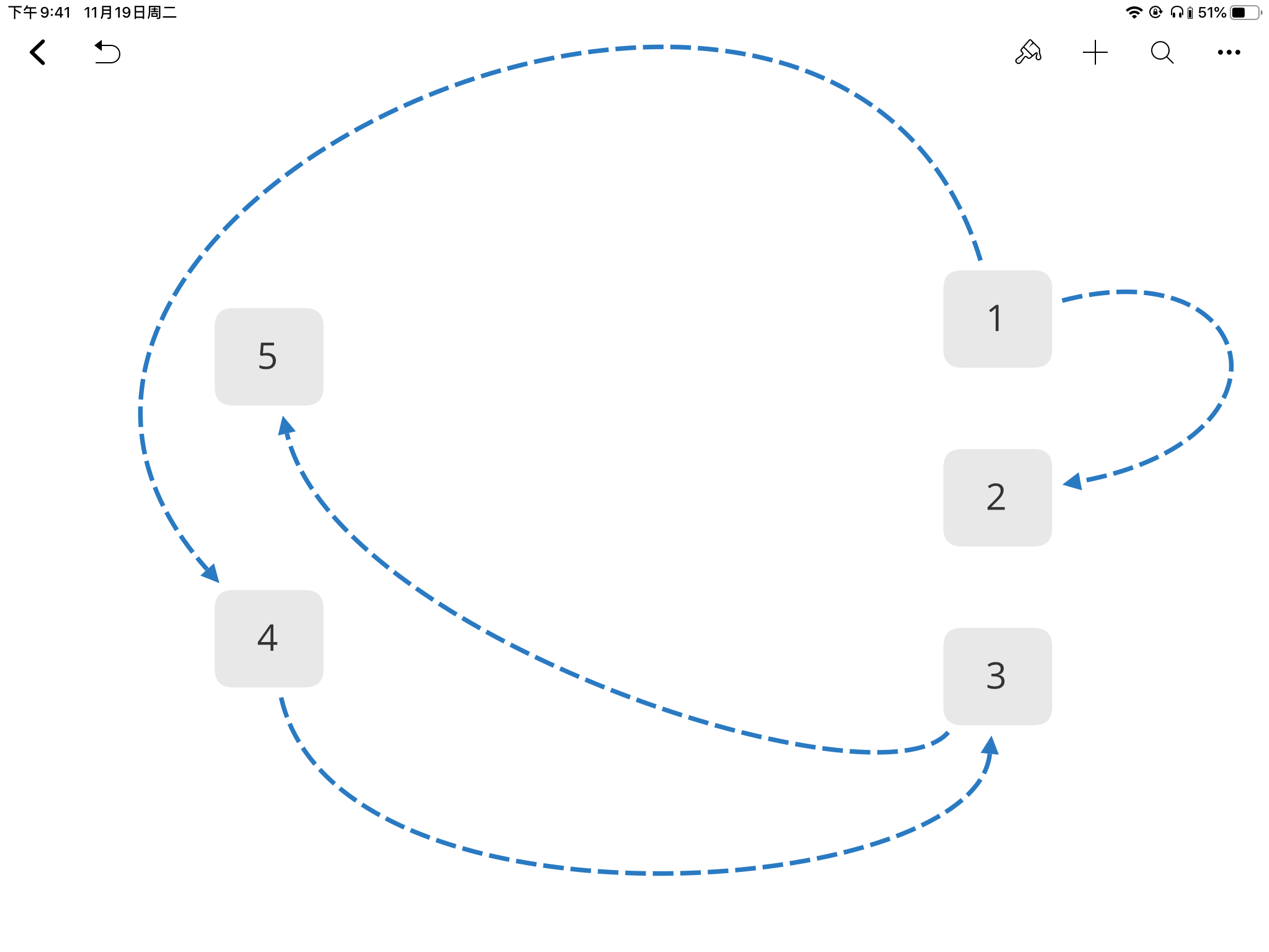
3.

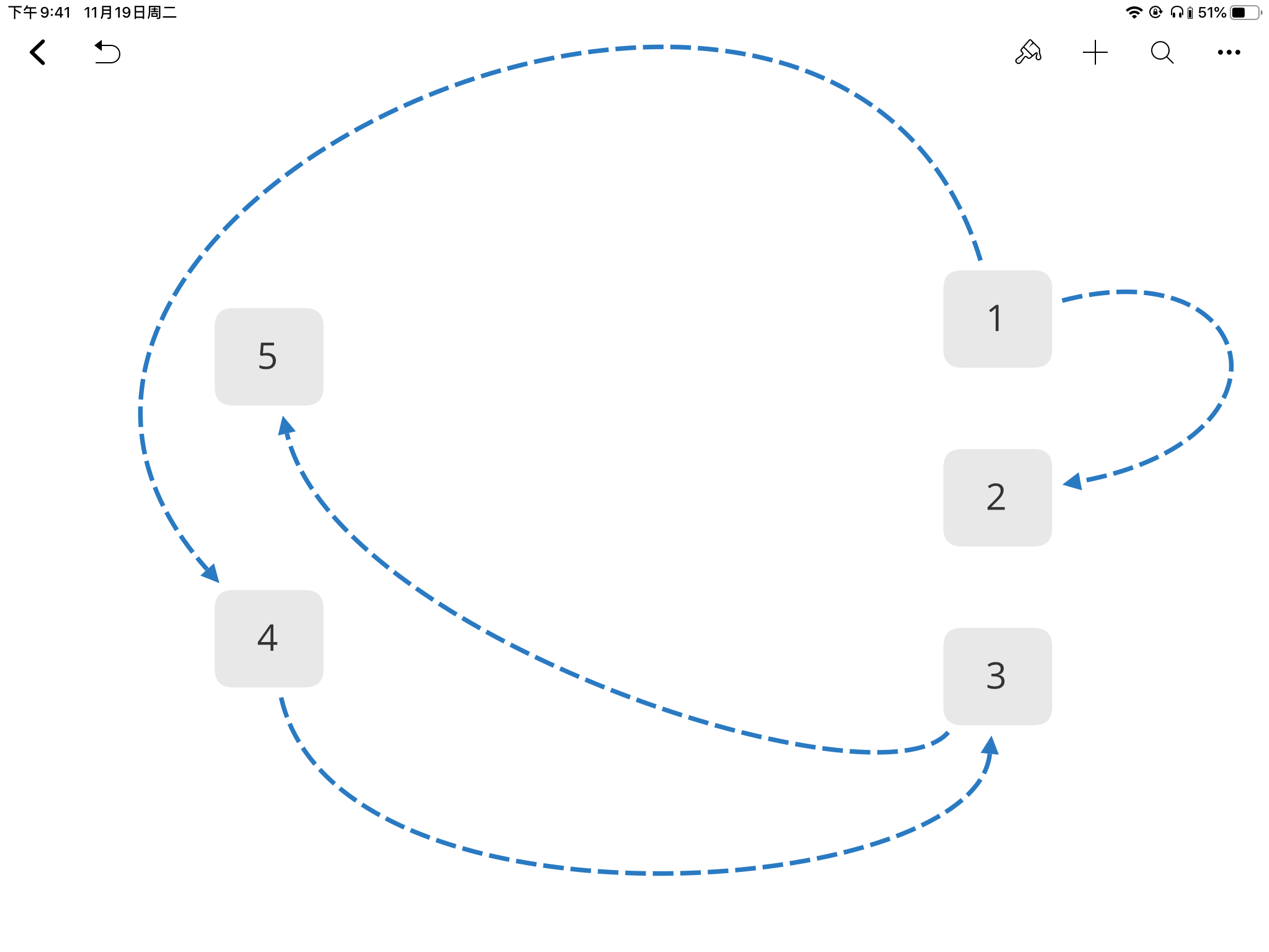
dfs序：1，2，4，3，5

bfs序：1，2，4，3，5

4.

dfs生成树：



bfs生成树：

1. 否，有边权相同
2. 1.prim：

1 2 3 4 5 6

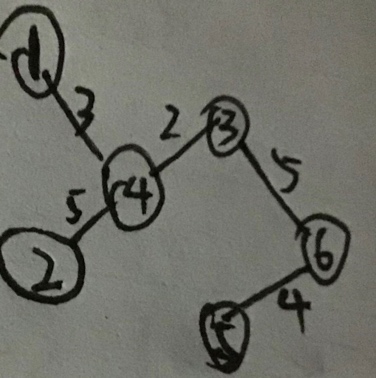
9 -1 inf 5 8 inf

3 -1 2 -1 8 7

3 -1 -1 -1 8 5

-1 -1 -1 -1 8 5

-1 -1 -1 -1 4 -1

-1 -1 -1 -1 -1 -1

2.kruskal

<3, 4> <1, 4> <5, 6> <2, 4> <3, 6>



1 inf

2 <4,2> 20

3 inf

4 0

5 inf

6 <4,6> 15

1 inf

2 <4,2> 20

3 inf

4 0

5 inf

6 <4,6> 15

1 <4,2,1> 30

2 <4,2> 20

3 inf

4 0

5 <4,2,5> 50

6 <4,6> 15

1 <4,2,1> 30

2 <4,2> 20

3 <4,2,1,3> 45

4 0

5 <4,2,5> 50

6 <4,6> 15

1 <4,2,1> 30

2 <4,2> 20

3 <4,2,1,3> 45

4 0

5 <4,2,5> 50

6 <4,6> 15

1 <4,2,1> 30

2 <4,2> 20

3 <4,2,1,3> 45

4 0

5 <4,2,5> 50

6 <4,6> 15



Initial:

a b c d e f

a 0 5<a,b> 3<a,c> inf inf inf

b inf 0 inf inf 9<b,e> 4<b,f>

c inf inf 0 2<c,d> 5<c,e> inf

d inf inf inf 0 4<d,e> inf

e inf inf inf Inf 0 inf

f inf inf inf inf 3<f,e> inf

a:

a b c d e f

a 0 5<a,b> 3<a,c> inf inf inf

b inf 0 inf inf 9<b,e> 4<b,f>

c inf inf 0 2<c,d> 5<c,e> inf

d inf inf inf 0 4<d,e> inf

e inf inf inf Inf 0 inf

f inf inf inf inf 3<f,e> inf

b:

a b c d e f

a 0 5<a,b> 3<a,c> inf 14<a,b,e> 9<a,b,f>

b inf 0 inf inf 9<b,e> 4<b,f>

c inf inf 0 2<c,d> 5<c,e> inf

d inf inf inf 0 4<d,e> inf

e inf inf inf inf 0 inf

f inf inf inf inf 3<f,e> inf

c:

a b c d e f

a 0 5<a,b> 3<a,c> 5<a,c,d> 8<a,c,e> 9<a,b,f>

b inf 0 inf inf 9<b,e> 4<b,f>

c inf inf 0 2<c,d> 5<c,e> inf

d inf inf inf 0 4<d,e> inf

e inf inf inf inf 0 inf

f inf inf inf inf 3<f,e> inf

d:

a b c d e f

a 0 5<a,b> 3<a,c> 5<a,c,d> 8<a,c,e> 9<a,b,f>

b inf 0 inf inf 9<b,e> 4<b,f>

c inf inf 0 2<c,d> 5<c,e> inf

d inf inf inf 0 4<d,e> inf

e inf inf inf inf 0 inf

f inf inf inf inf 3<f,e> inf

e:

a b c d e f

a 0 5<a,b> 3<a,c> 5<a,c,d> 8<a,c,e> 9<a,b,f>

b inf 0 inf inf 9<b,e> 4<b,f>

c inf inf 0 2<c,d> 5<c,e> inf

d inf inf inf 0 4<d,e> inf

e inf inf inf inf 0 inf

f inf inf inf inf 3<f,e> inf

f:

a b c d e f

a 0 5<a,b> 3<a,c> 5<a,c,d> 8<a,c,e> 9<a,b,f>

b inf 0 inf inf 7<b,f,e> 4<b,f>

c inf inf 0 2<c,d> 5<c,e> inf

d inf inf inf 0 4<d,e> inf

e inf inf inf inf 0 inf

f inf inf inf inf 3<f,e> inf



output:{}

indegree:{0,1,1,1,2,1,3}

stack:{0}

output:{0}

indegree:{0,0,0,1,2,1,3}

stack:{1,2}

output:{0,2}

indegree:{0,0,0,1,1,1,2}

stack:{1}

output:{0,2,1}

indegree:{0,0,0,0,0,0,2}

stack:{3,4,5}

output:{0,2,1,5}

indegree:{0,0,0,0,0,0,1}

stack:{3,4}

output:{0,2,1,5,4}

indegree:{0,0,0,0,0,0,0}

stack:{3,6}

output:{0,2,1,5,4,6}

indegree:{0,0,0,0,0,0,0}

stack:{3}

output:{0,2,1,5,4,6,3}

indegree:{0,0,0,0,0,0,0}

stack:{}



1,2,4,3,9,7,8,6,5

1,2,4,3,9,8,7,6,5

1,2,4,9,3,7,8,6,5

1,2,4,9,3,8,7,6,5

1,2,4,9,7,3,8,6,5

1,2,4,9,7,8,3,6,5

1,2,4,9,7,8,6,3,5

1,2,4,9,8,3,7,6,5

1,2,4,9,8,7,3,6,5

1,2,4,9,8,7,6,3,5

1,2,9,4,3,7,8,6,5

1,2,9,4,3,8,7,6,5

1,2,9,4,7,3,8,6,5

1,2,9,4,7,8,3,6,5

1,2,9,4,7,8,6,3,5

1,2,9,4,8,3,7,6,5

1,2,9,4,8,7,3,6,5

1,2,9,4,8,7,6,3,5

1,2,9,8,4,3,7,6,5

1,2,9,8,4,7,3,6,5

1,2,9,8,4,7,6,3,5

1,9,2,4,3,7,8,6,5

1,9,2,4,3,8,7,6,5

1,9,2,4,7,3,8,6,5

1,9,2,4,7,8,3,6,5

1,9,2,4,7,8,6,3,5

1,9,2,4,8,3,7,6,5

1,9,2,4,8,7,3,6,5

1,9,2,4,8,7,6,3,5

1,9,2,8,4,3,7,6,5

1,9,2,8,4,7,3,6,5

1,9,2,8,4,7,6,3,5

1,9,8,2,4,3,7,6,5

1,9,8,2,4,7,3,6,5

1,9,8,2,4,7,6,3,5

9,1,2,4,3,7,8,6,5

9,1,2,4,3,8,7,6,5

9,1,2,4,7,3,8,6,5

9,1,2,4,7,8,3,6,5

9,1,2,4,7,8,6,3,5

9,1,2,4,8,3,7,6,5

9,1,2,4,8,7,3,6,5

9,1,2,4,8,7,6,3,5

9,1,2,8,4,3,7,6,5

9,1,2,8,4,7,3,6,5

9,1,2,8,4,7,6,3,5

9,1,8,2,4,3,7,6,5

9,1,8,2,4,7,3,6,5

9,1,8,2,4,7,6,3,5

9,8,1,2,4,3,7,6,5

9,8,1,2,4,7,3,6,5

9,8,1,2,4,7,6,3,5



void topological\_sort() {

int i, u, v;

for (v = 1; v <= n; v++) {

for (u = 1; u <= n; u++) {

indegree[v] += E[u][v];

}

if (!indegree[v]) {

stack.push\_back(v);

}

}

for (i = 1; i <= n; i++) {

u = stack.back();

stack.pop\_back();

vis[u] = 1;

cout << u - 1 << ' ';

for (v = 1; v <= n; v++) {

if (!vis[v]) {

indegree[v] -= E[u][v];

if (!indegree[v]) {

stack.push\_back(v);

}

}

}

}

}

12．

void dfs(int S) {

int i, j, u, vis[MAX\_NUM];

vector<int> u\_stack, path, S\_stack;

memset(vis, 0, sizeof(vis));

for (u = 1; u <= n; u++) {

u\_stack.push\_back(u);

S\_stack.push\_back(S);

}

while (!u\_stack.empty()) {

u = u\_stack.back();

S = S\_stack.back();

if (vis[u]) {

vis[u] = 0;

path.pop\_back();

S\_stack.pop\_back();

u\_stack.pop\_back();

continue;

}

vis[u] = 1;

path.push\_back(u);

for (i = 0; i < G[u].size(); i++) {

if (!vis[G[u][i].v]) {

if (!(S - G[u][i].w)) {

for (j = 0; j < path.size(); j++) {

cout << path[j] << ' ';

}

cout << G[u][i].v << endl;

}

else if (S - G[u][i].w > 0) {

u\_stack.push\_back(G[u][i].v);

S\_stack.push\_back(S - G[u][i].w);

}

}

}

}

}

13．

1.最早发生时间：{0, 5, 6, 18, 21, 21, 23, 25, 28, 30}

最晚发生时间：{0, 15, 6, 18, 22, 26, 23, 26, 28, 30}

2.至少时间：30

3.关键路径：0, 2, 3, 6, 8, 9

关键活动：{0, 2, 3, 6, 8, 9}