EDA软件设计-调度算法

2024-2025年春季学期

主讲人：孙静翎

有约束的调度

Constrained Scheduling

- 约束调度
  - 一般情况下是NP完全问题
  - 在面积或资源的约束下最小化延迟（ML-RCS）
  - 使受到延迟约束的资源最小化（MR-LCS）
- 确切解决方法
  - ILP：整数线性规划（Integer linear program）
  - Hu算法：适用于只有一种资源类型的问题
- 启发式算法
  - 列表调度（List scheduling）
  - 力导向调度（Force-directed scheduling）

列表调度算法：ML-RCS

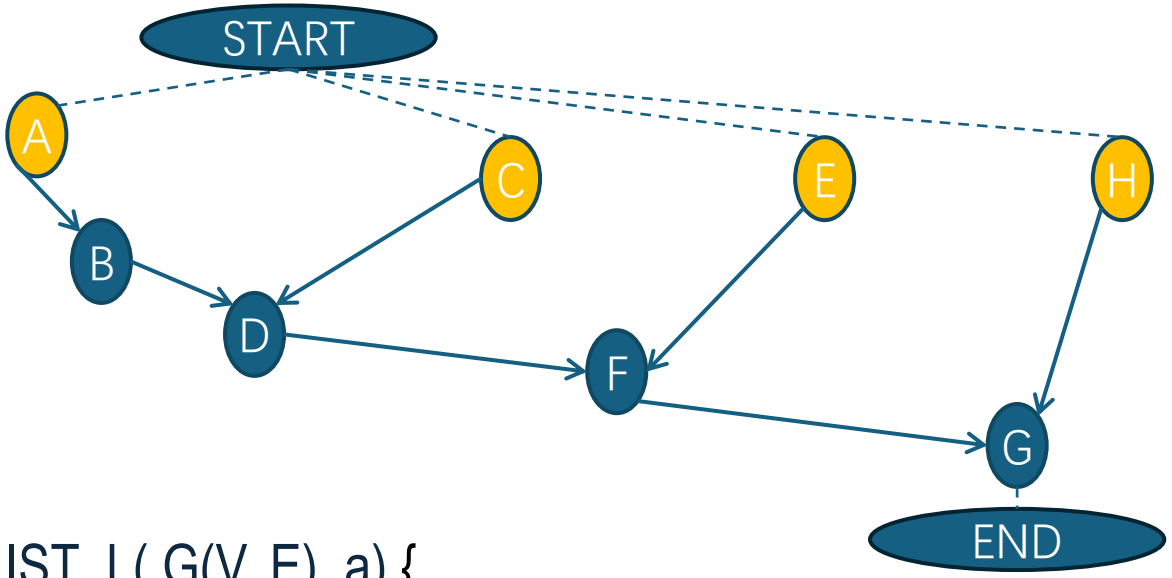
List scheduling algorithm for minimum latency

```
LIST_L( G(V, E), a) {
        t0 = 0;  l = 1;
        repeat {
                for each resource type k = 1, 2, …, n_res {
                    Determine ready operations U_l,k;
                    Determine unfinished operations T_l,k;
                    Select S_k ⊆ U_l,k vertices, s.t. |S_k| + |T_l,k| ≤ a_k;
                    Schedule the S_k operations at step l;
                    }
                l = l + 1;
        }
        until (v_n is scheduled) ;
}
```

列表调度算法：ML-RCS

List scheduling algorithm for minimum latency

1 LIST_L(G(V, E), a) {
2 t0 = 0; l = 1;
3 重复执行以下步骤 {
4 对于每种资源类型 k = 1, 2, …, n_{res} {
5 确定就绪的操作集 $U_{l,k}$;（①还未安排开始周期，②前置结点已在当前周期完成）
6 确定当前正在进行的操作集 $T_{l,k}$;
7 选择一个子集$S_k \subseteq U_{l,k}$,使得 $|S_k| + |T_{l,k}| \le a_k$;
8 在步骤 l 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = l;
9 }
10 $l = l + 1$;
11 }
12 直到 vn 被调度 ;
13 }

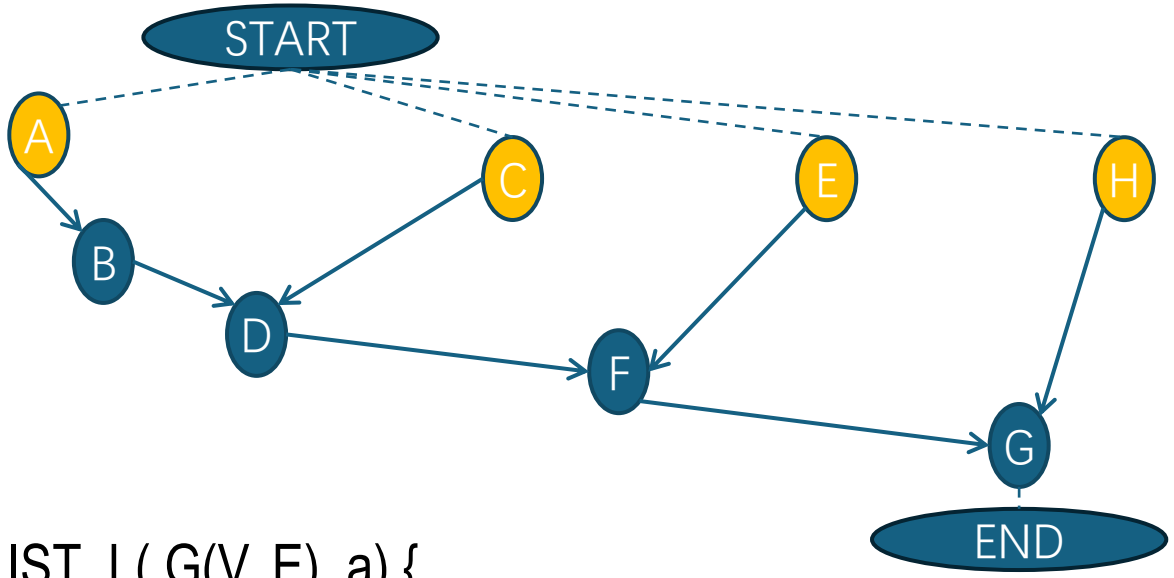


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	1
k	
U	
T	
S	

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;   I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                   确定就绪的操作集 U_I,k;
6                   确定当前正在进行的操作集 T_I,k;
7                   选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                   在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

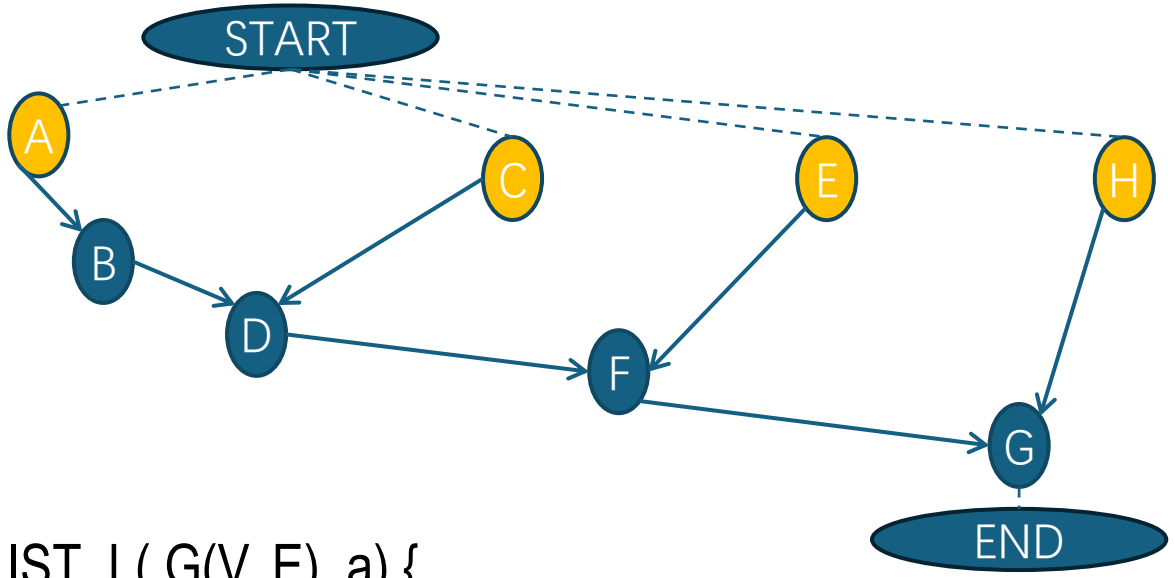


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	1
k	
U	
T	
S	

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

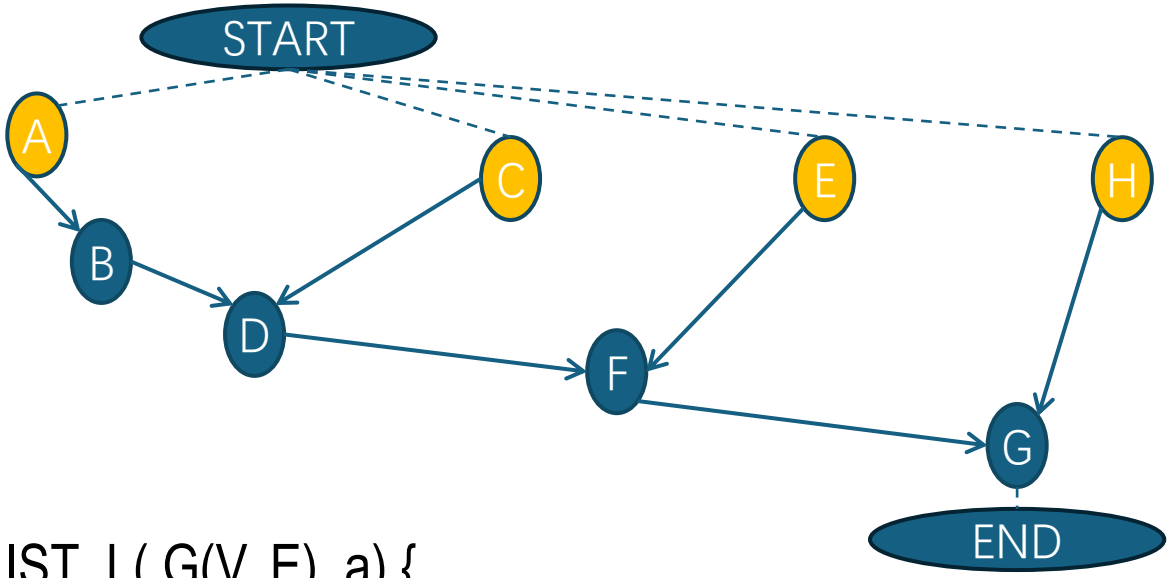


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	1
k	1
U	
T	
S	

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

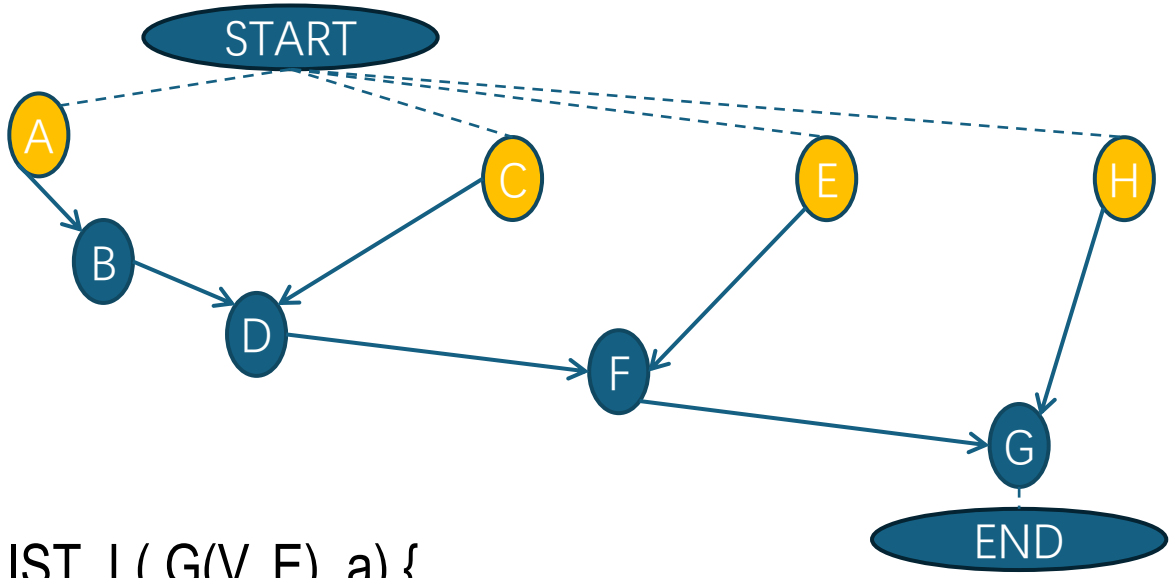


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	1
k	1
U	{}
T	
S	

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

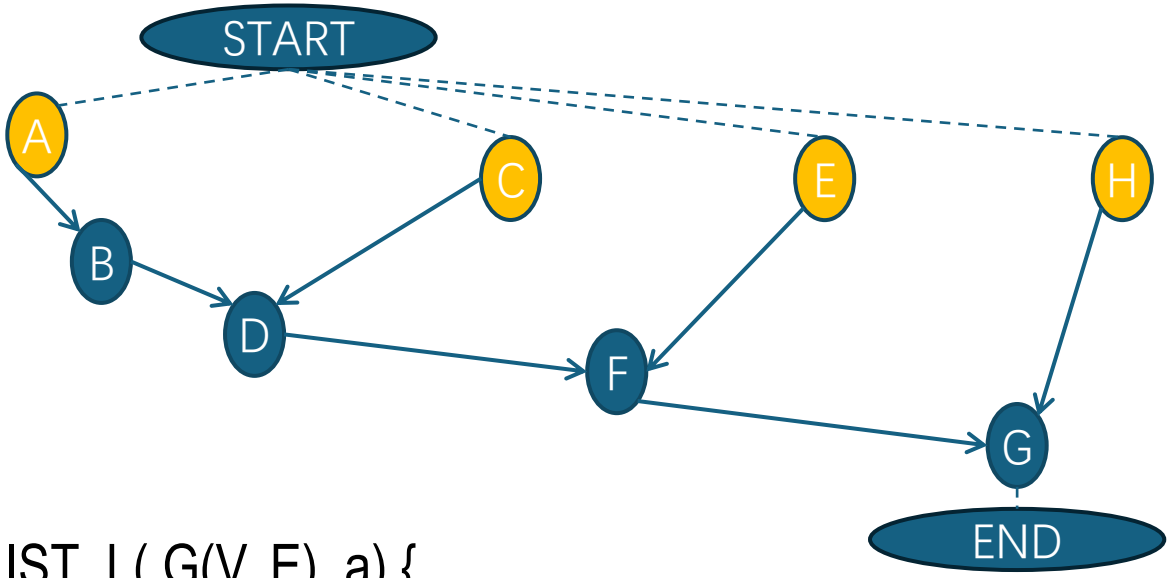


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	1
k	1
U	{}
T	{}
S	

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

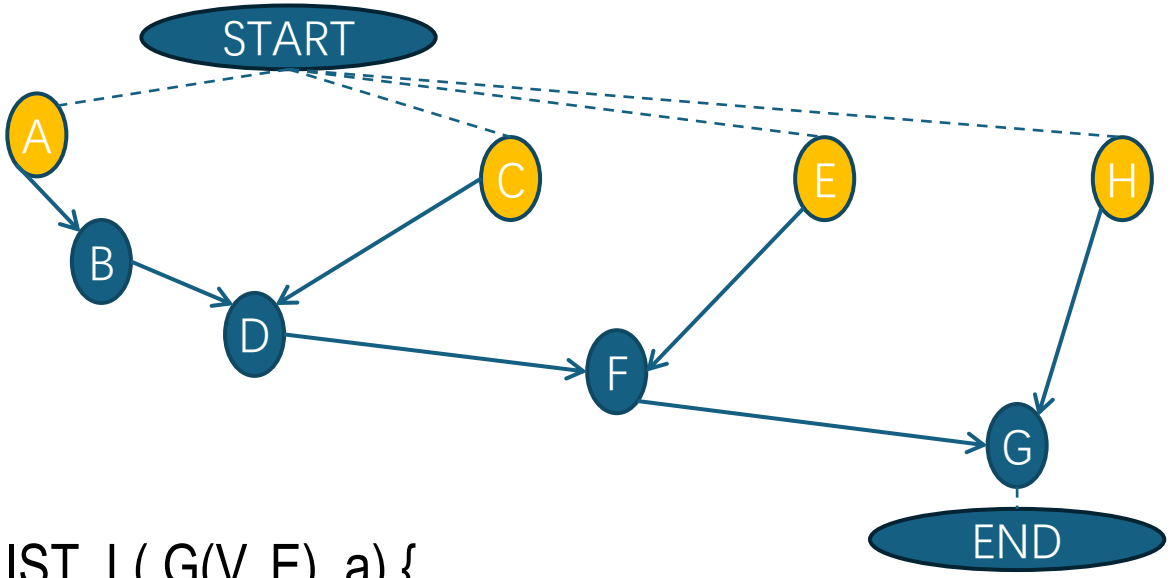


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
l	1
k	1
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  l = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_l,k;
6                 确定当前正在进行的操作集 T_l,k;
7                 选择一个子集S_k ⊆ U_l,k ,使得 |S_k| + |T_l,k| ≤ a_k;
8                 在步骤 l 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = l;
9             }
10            l = l + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

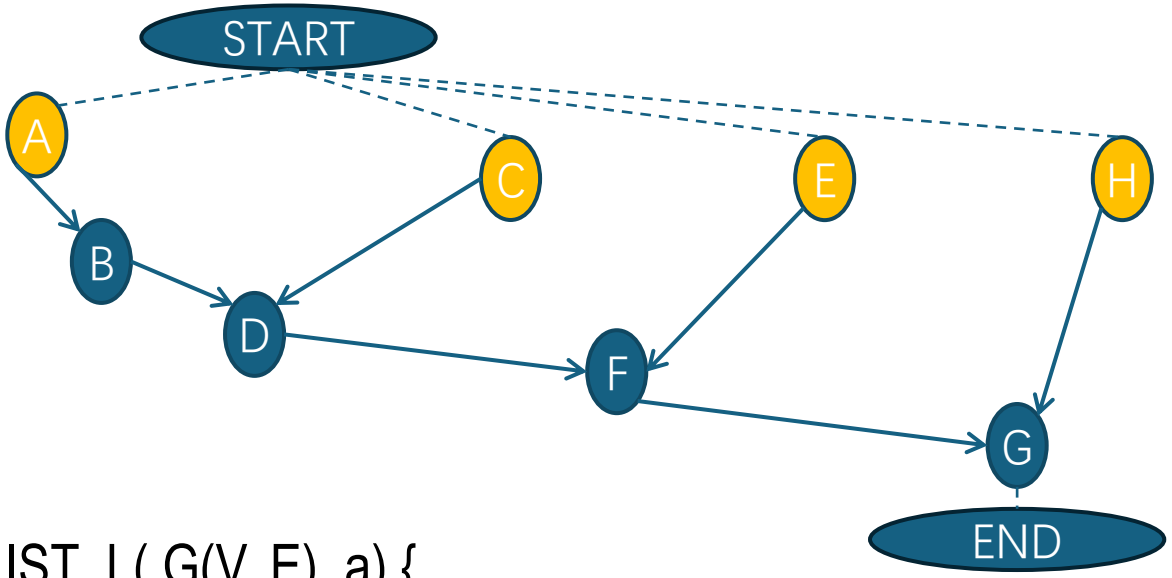


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
l	1
k	1
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  l = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_l,k;
6                 确定当前正在进行的操作集 T_l,k;
7                 选择一个子集S_k ⊆ U_l,k ,使得 |S_k| + |T_l,k| ≤ a_k;
8                 在步骤 l 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = l;
9             }
10            l = l + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

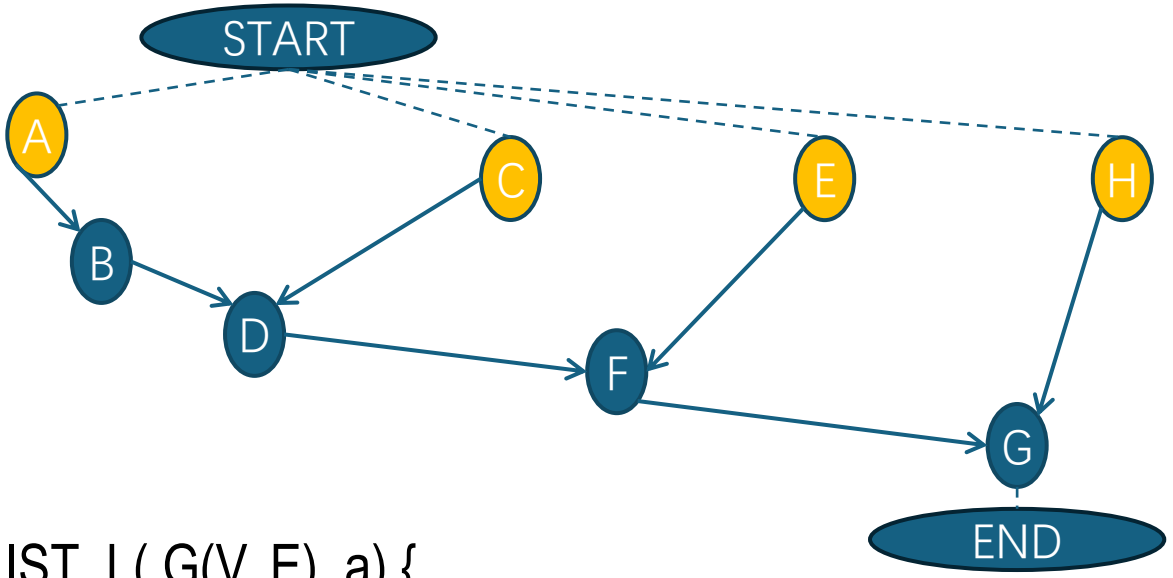


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
l	1
k	2
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  l = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_l,k;
6                 确定当前正在进行的操作集 T_l,k;
7                 选择一个子集S_k ⊆ U_l,k ,使得 |S_k| + |T_l,k| ≤ a_k;
8                 在步骤 l 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = l;
9             }
10            l = l + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

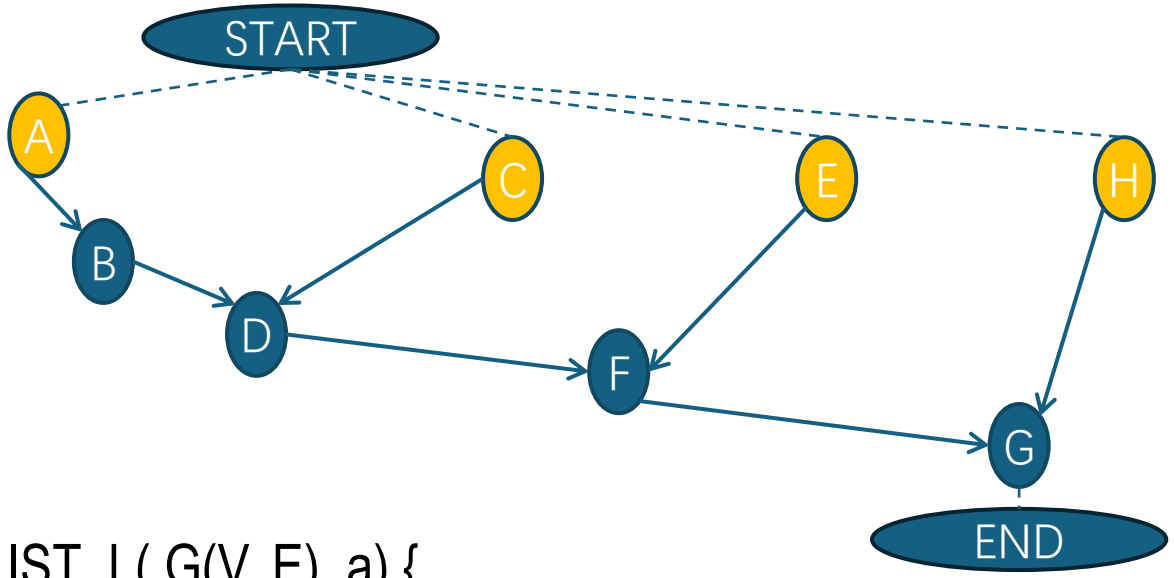


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	1
k	2
U	{A,C,E,H}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

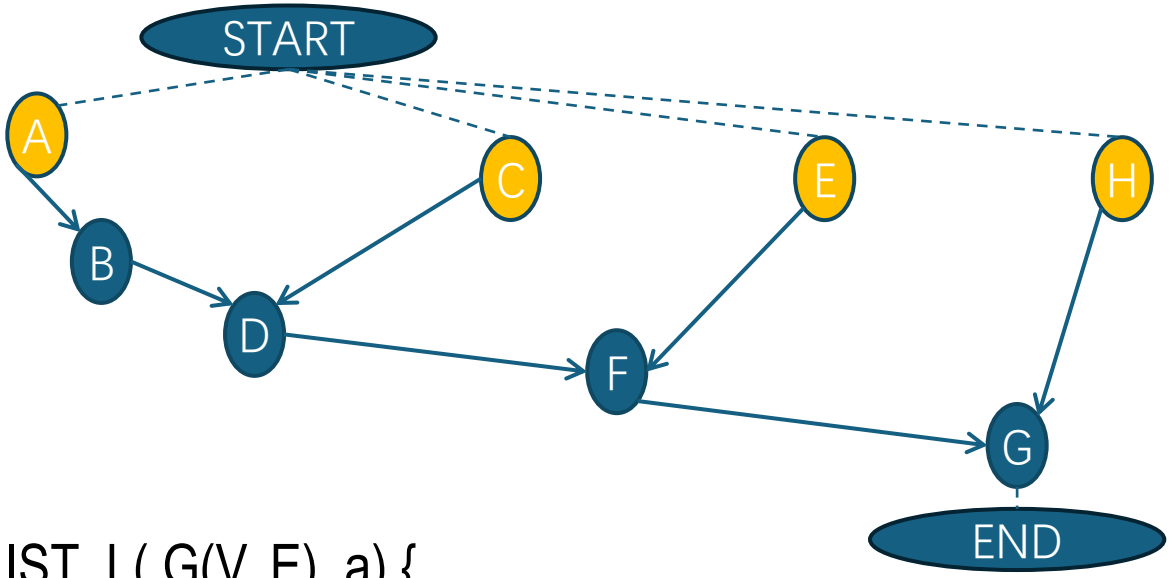


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	1
k	2
U	{A,C,E,H}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

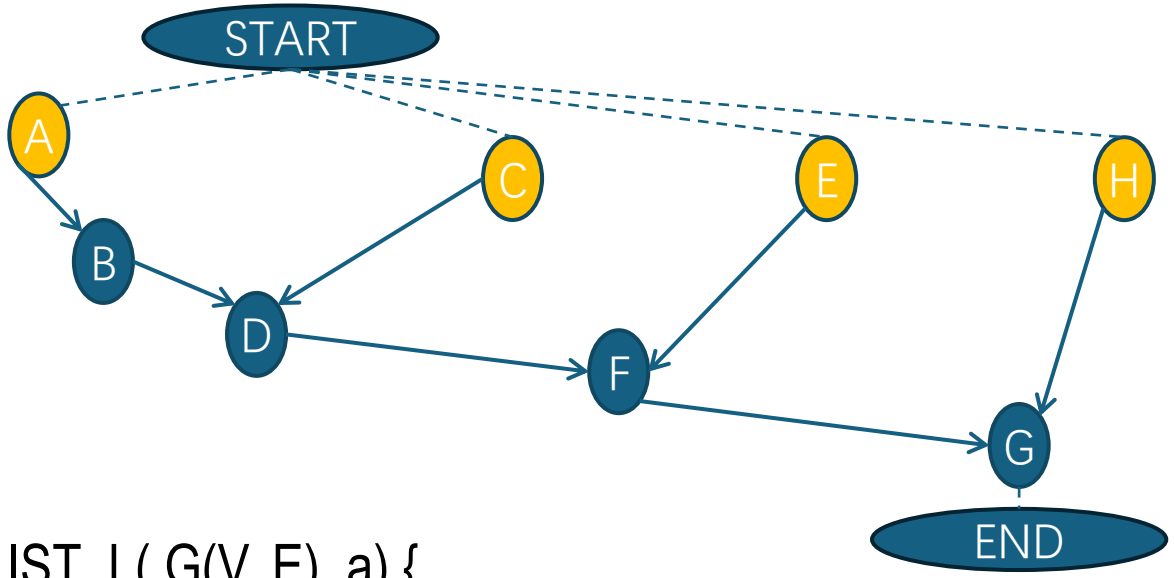


START 0

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	1
k	2
U	{A,C,E,H}
T	{}
S	{A,C}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

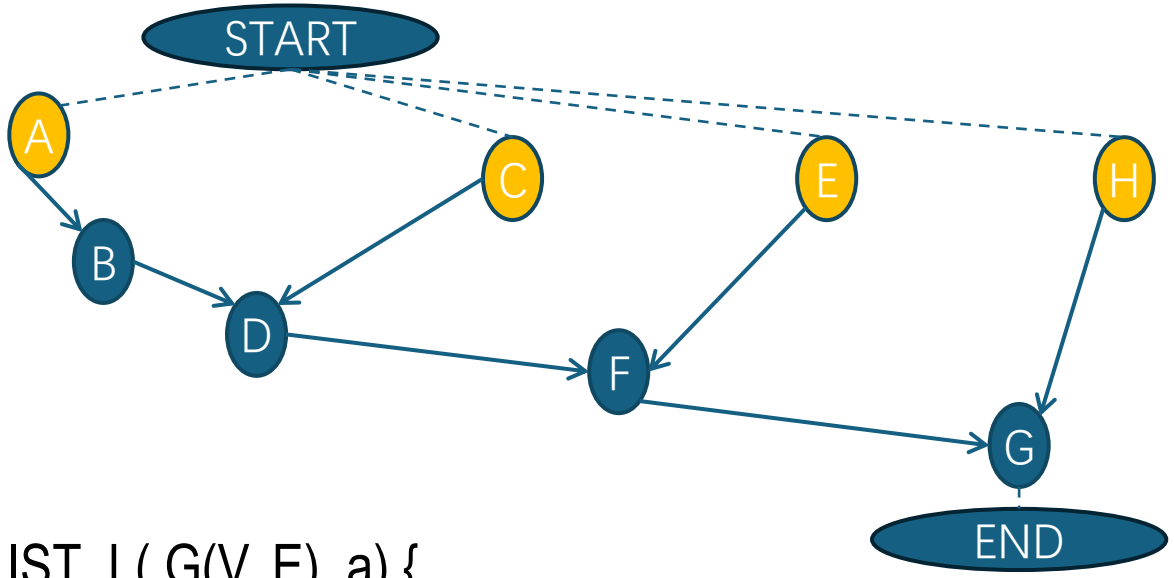
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	1
k	2
U	{A,C,E,H}
T	{}
S	{A,C}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                   确定就绪的操作集 U_I,k;
6                   确定当前正在进行的操作集 T_I,k;
7                   选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                   在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

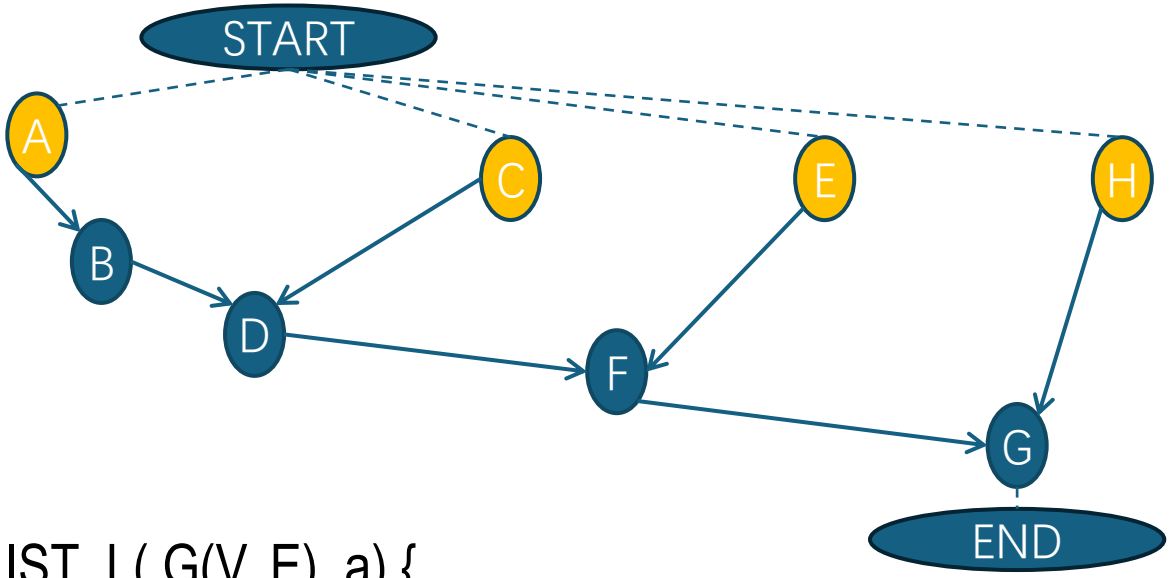
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	2
U	{A,C,E,H}
T	{}
S	{A,C}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

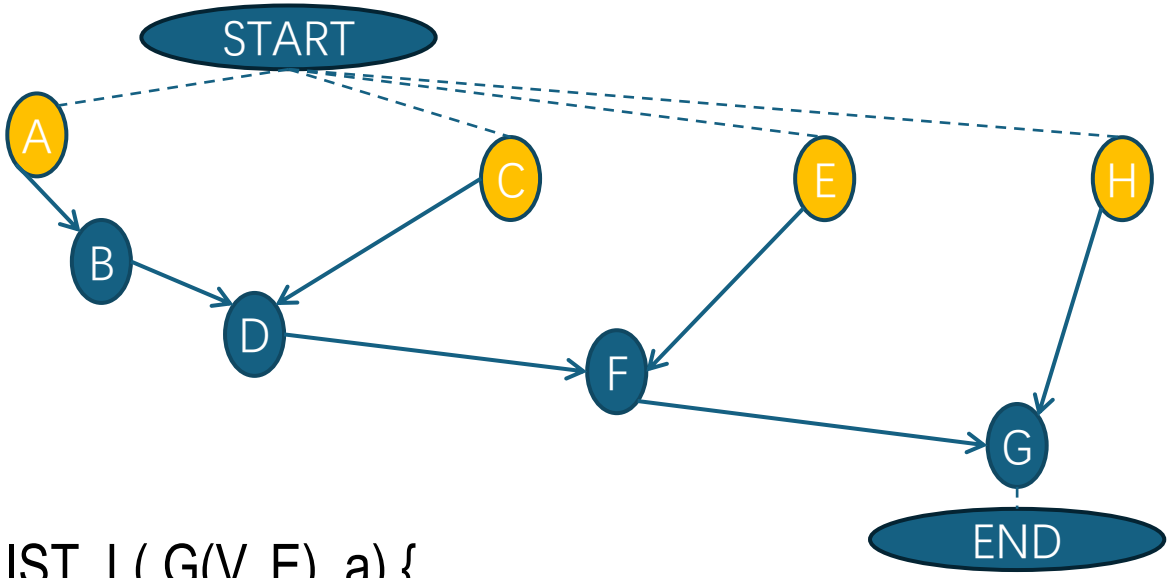
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	2
U	{A,C,E,H}
T	{}
S	{A,C}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

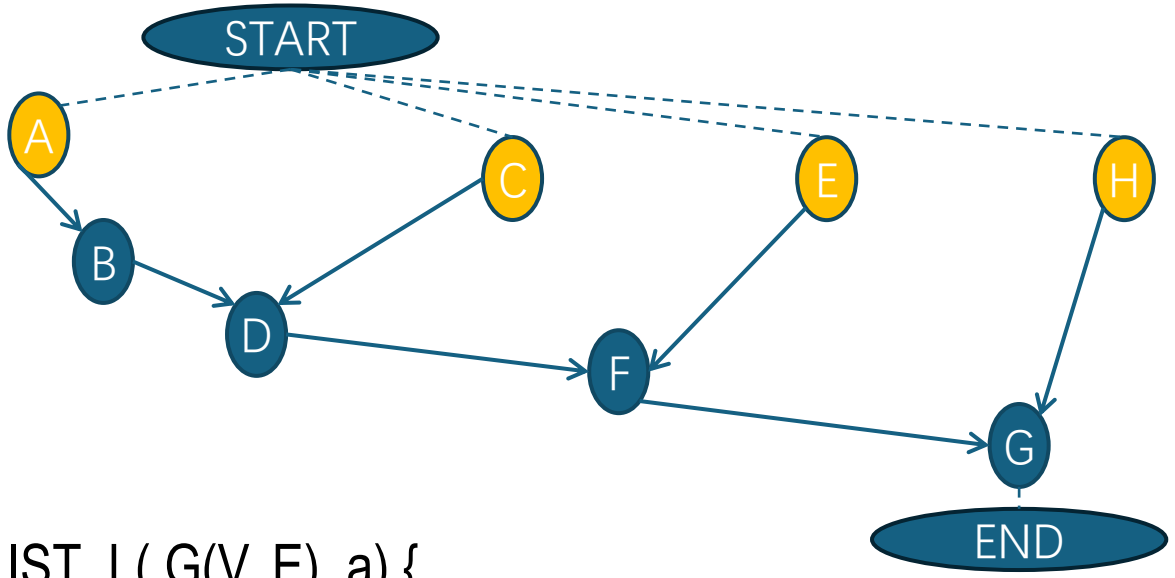
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	1
U	{A,C,E,H}
T	{}
S	{A,C}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

C 1

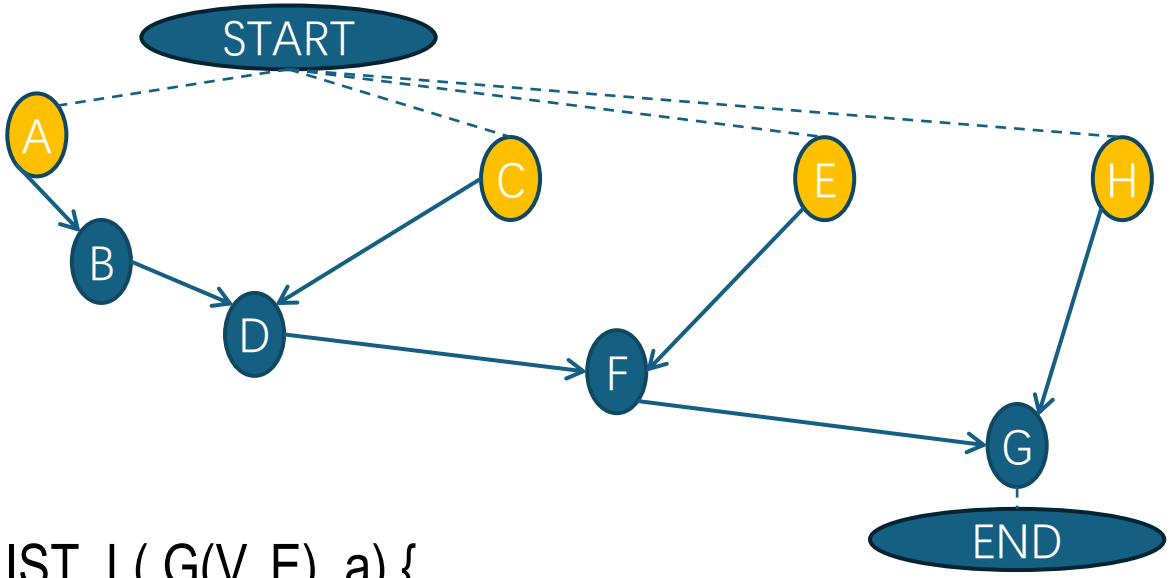
k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	1
U	{}
T	{}
S	{A,C}

B没有就绪，因为B要等待A执行完成，即周期1+2=3才能开工

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

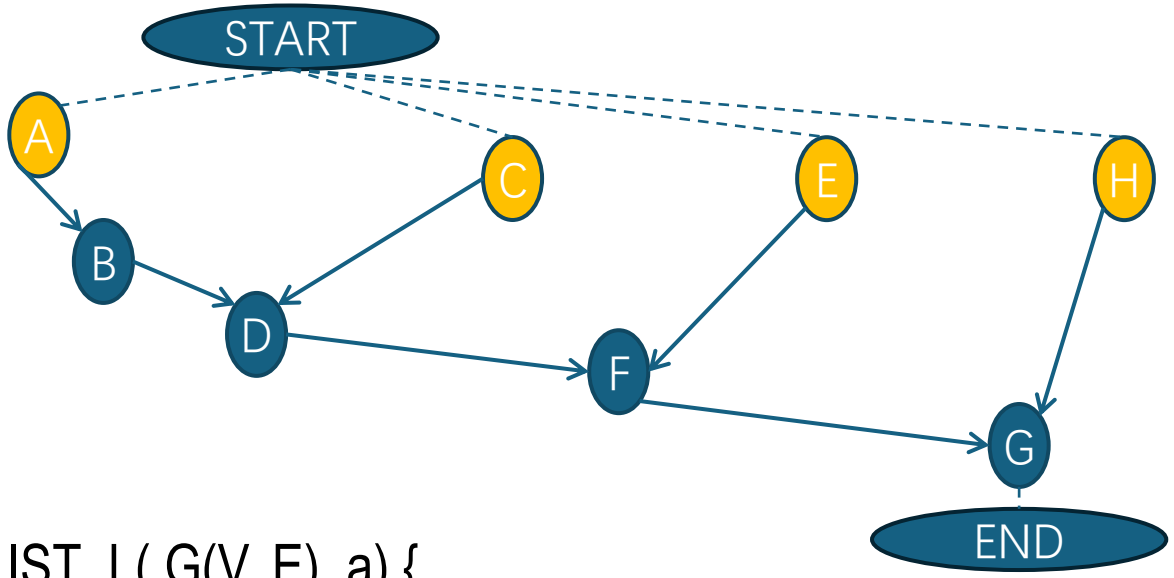
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	1
U	{}
T	{}
S	{A,C}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

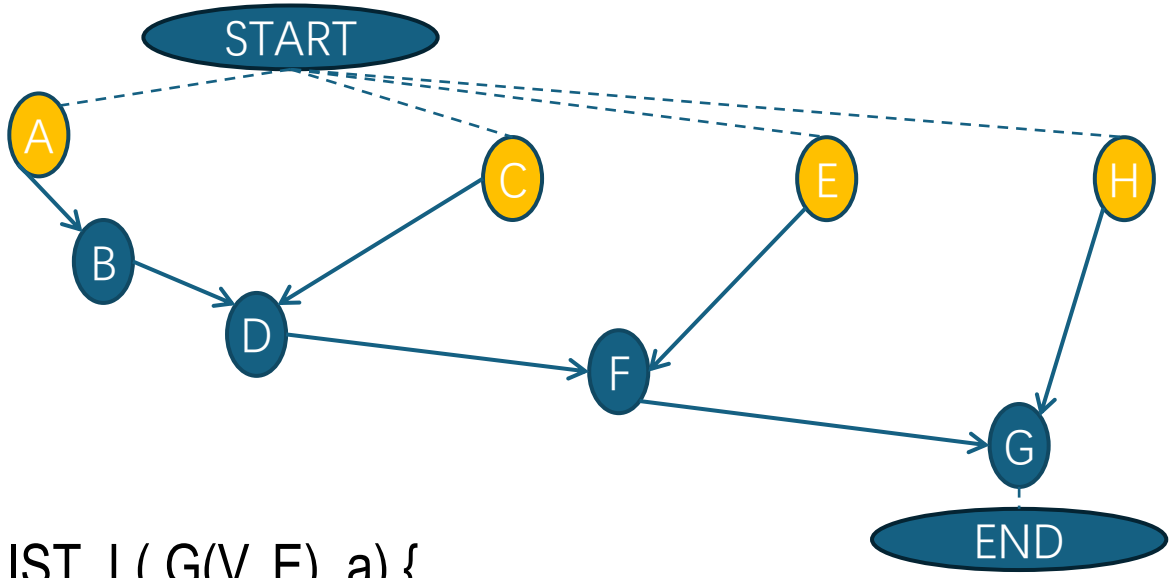
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	1
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

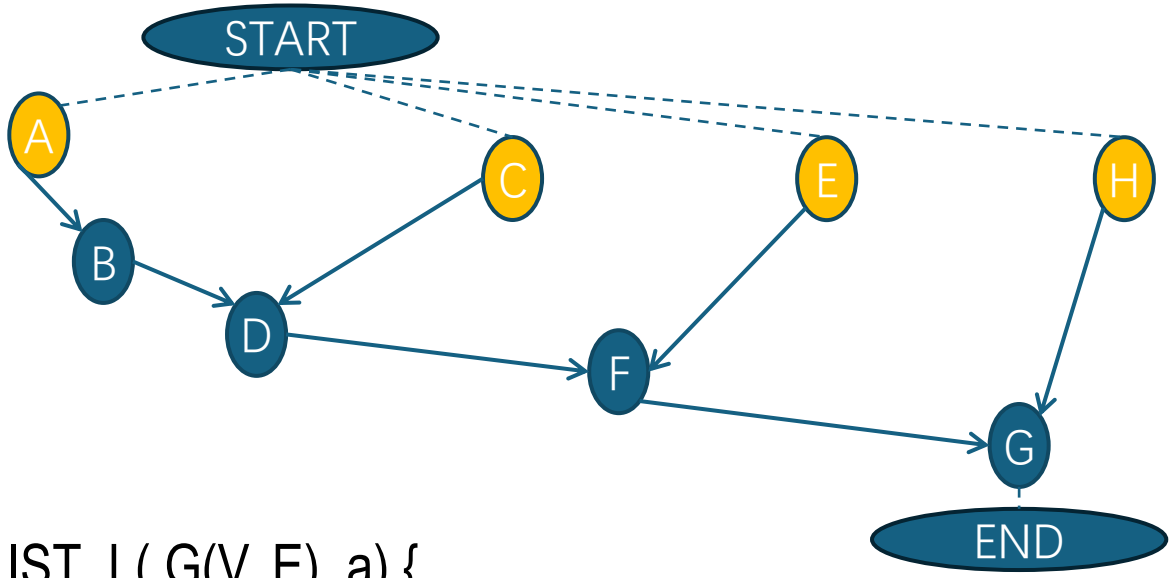
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	1
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

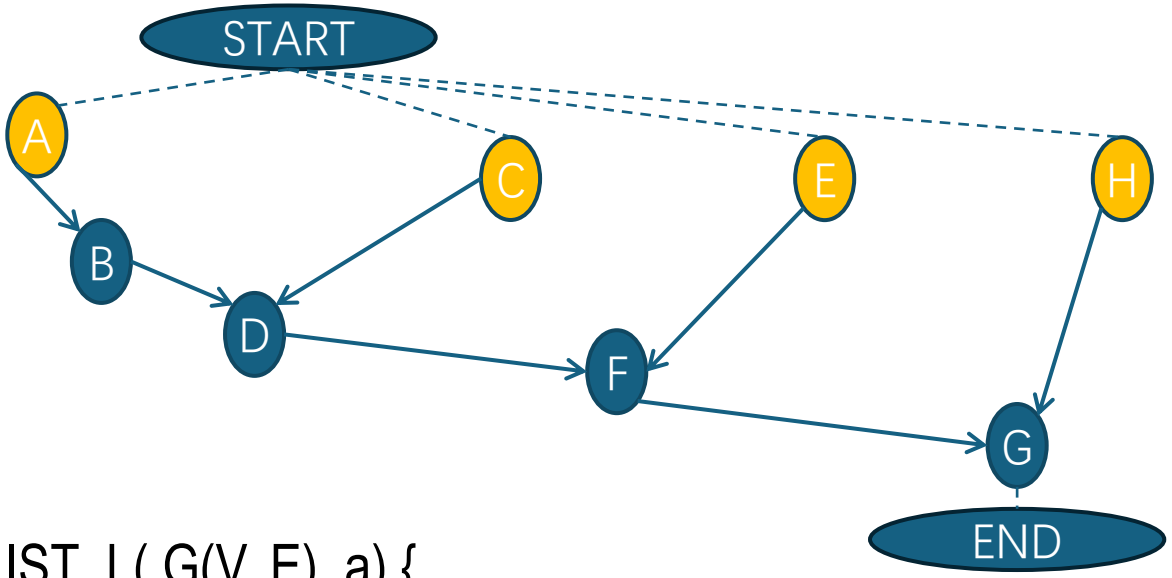
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
l	2
k	2
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  l = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_l,k;
6                 确定当前正在进行的操作集 T_l,k;
7                 选择一个子集S_k ⊆ U_l,k ,使得 |S_k| + |T_l,k| ≤ a_k;
8                 在步骤 l 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = l;
9             }
10            l = l + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

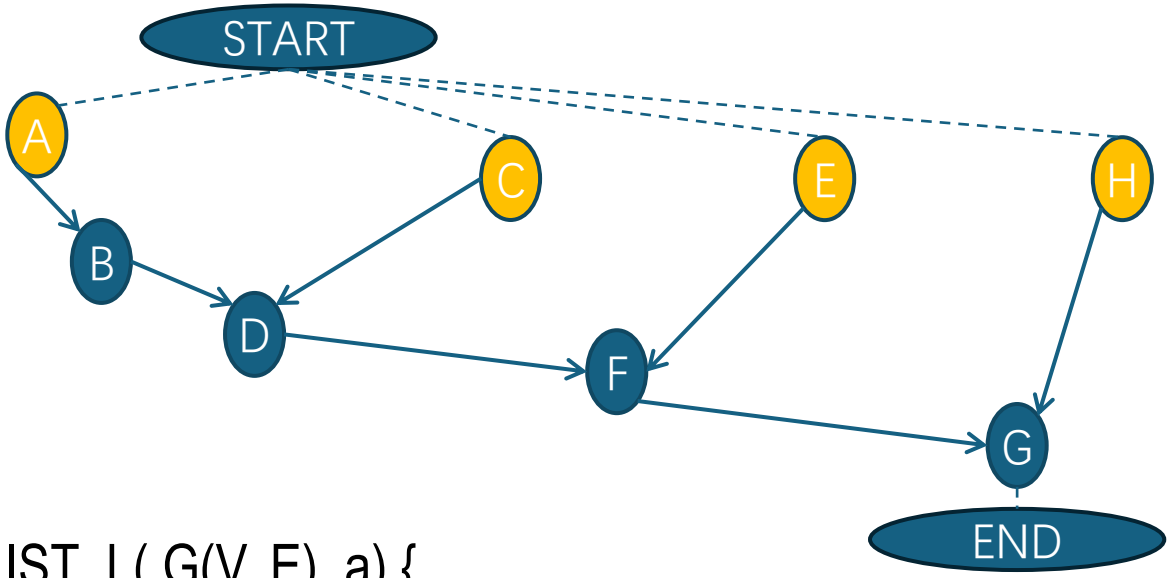
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	2
U	{E,H}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

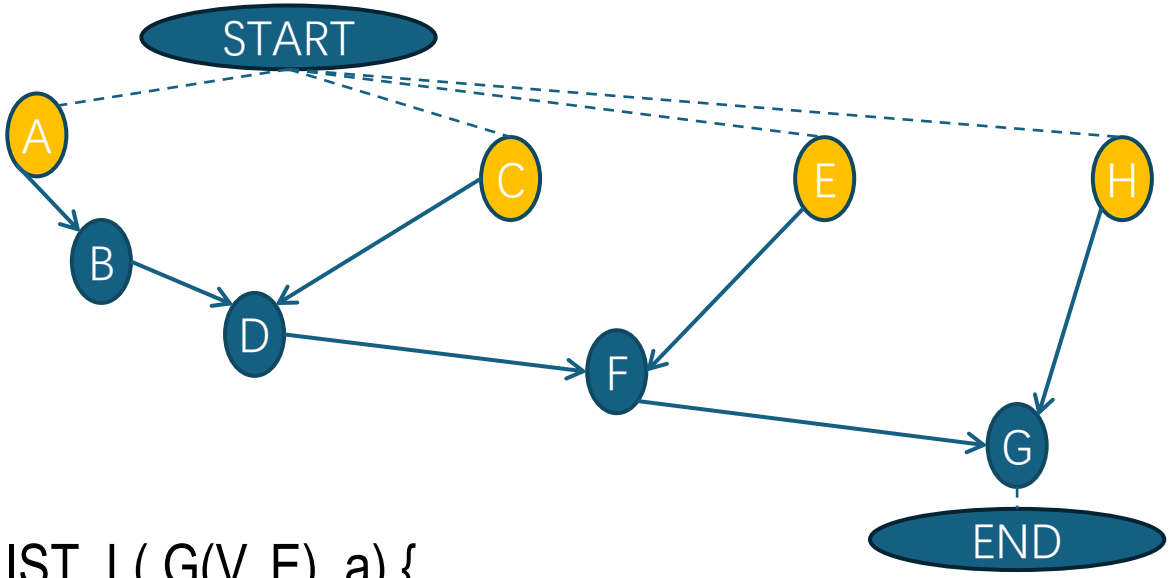
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	2
U	{E,H}
T	{A,C}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

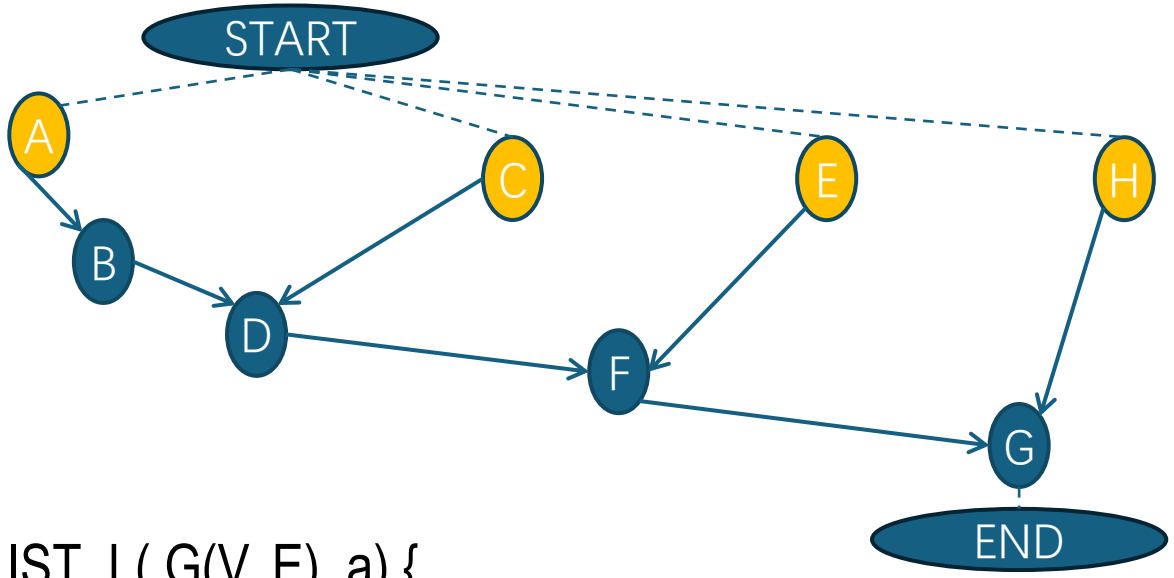
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	2
U	{E,H}
T	{A,C}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

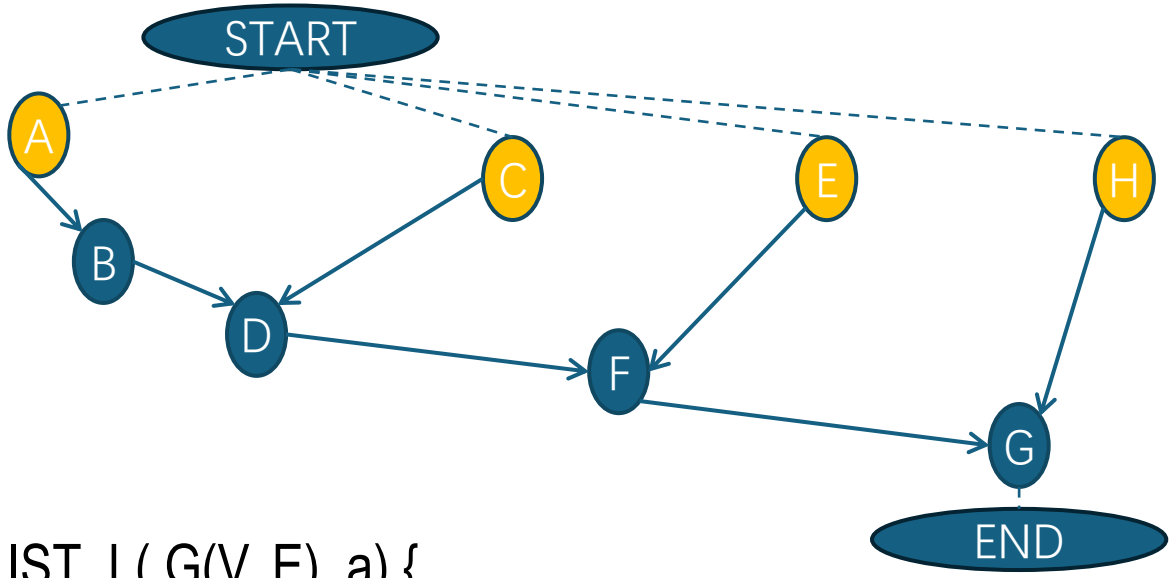
C 1

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	2
k	2
U	{E,H}
T	{A,C}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                   确定就绪的操作集 U_{I,k};
6                   确定当前正在进行的操作集 T_{I,k};
7                   选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                   在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

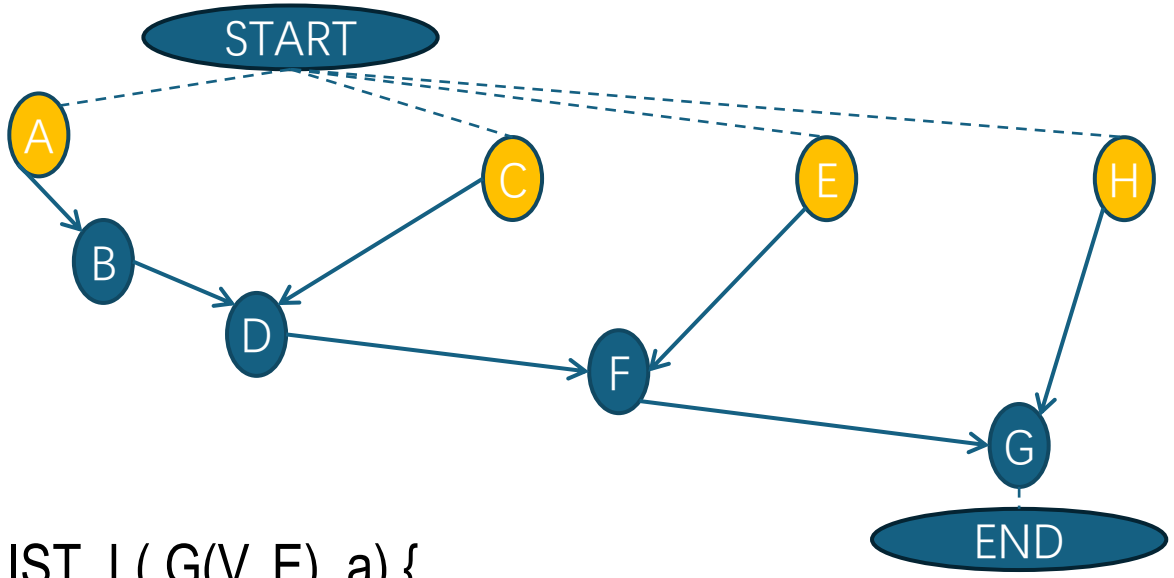


START 0
A 1
C 1

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	3
k	2
U	{E,H}
T	{A,C}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

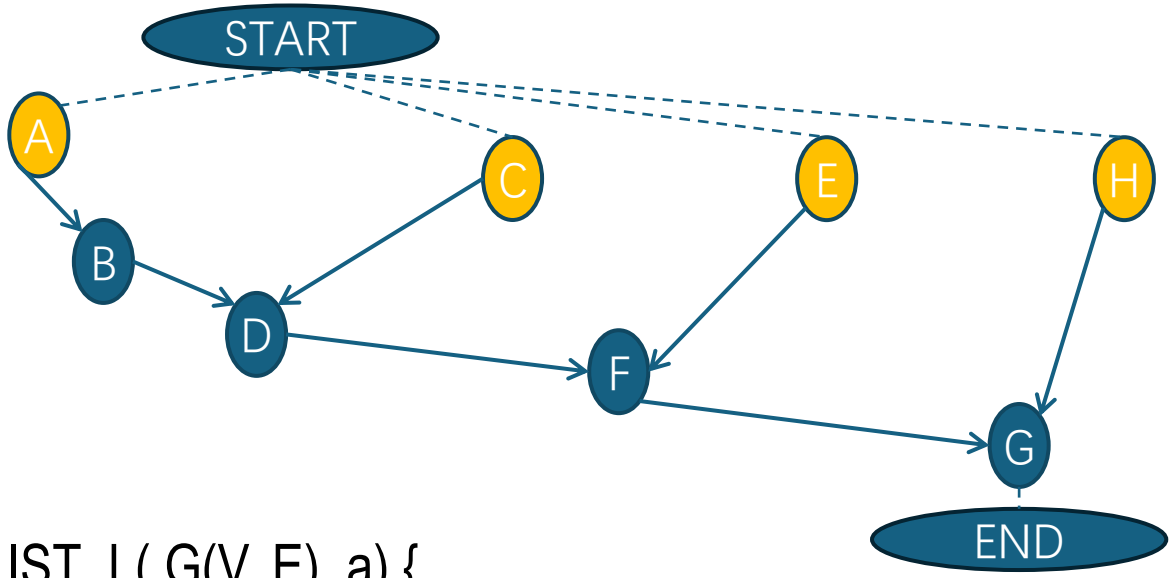


START 0
A 1
C 1

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	3
k	2
U	{E,H}
T	{A,C}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

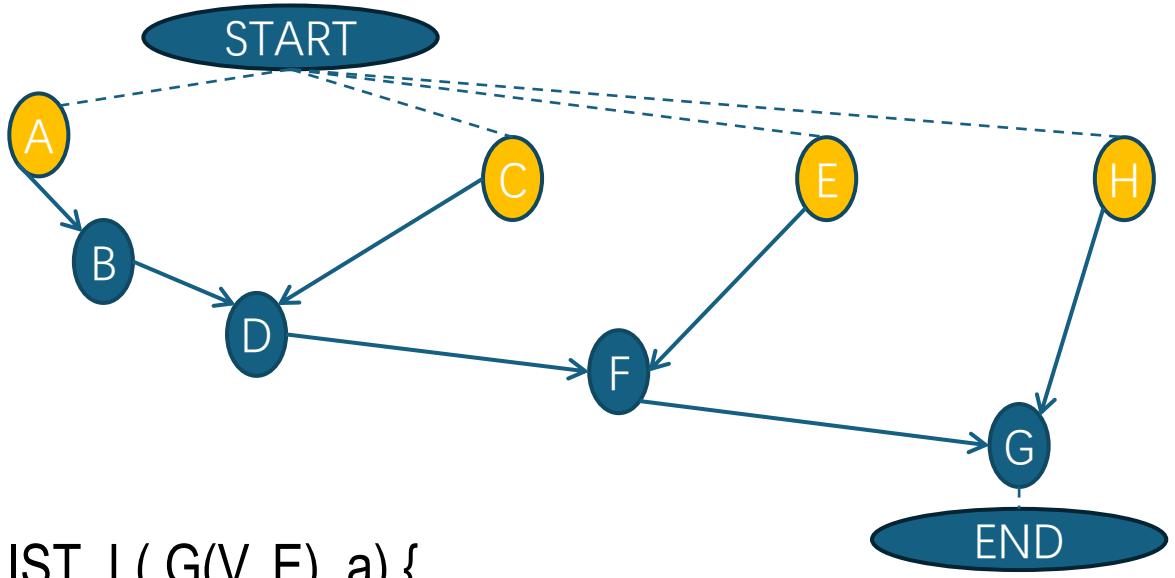


START 0
A 1
C 1

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	3
k	1
U	{E,H}
T	{A,C}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

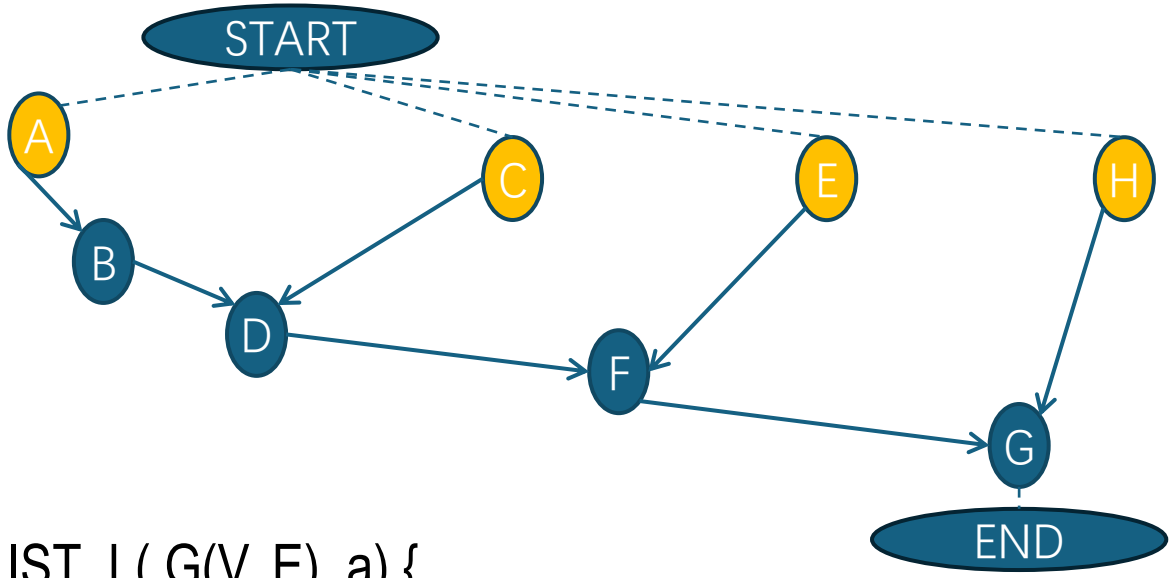


START 0
A 1
C 1

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	3
k	1
U	{B}
T	{A,C}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

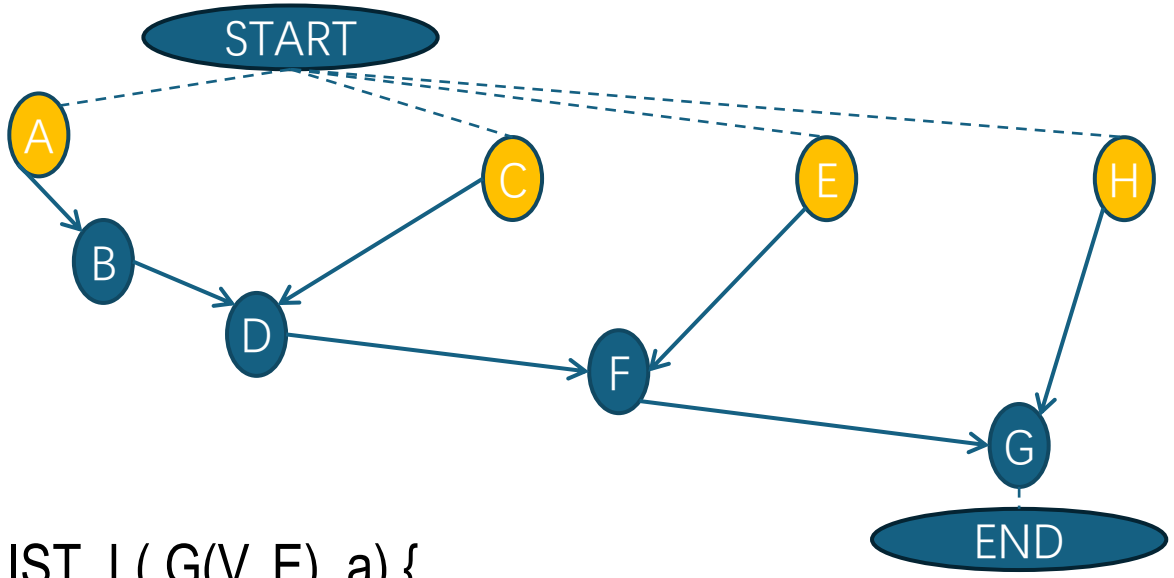


START 0
A 1
C 1

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	3
k	1
U	{B}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

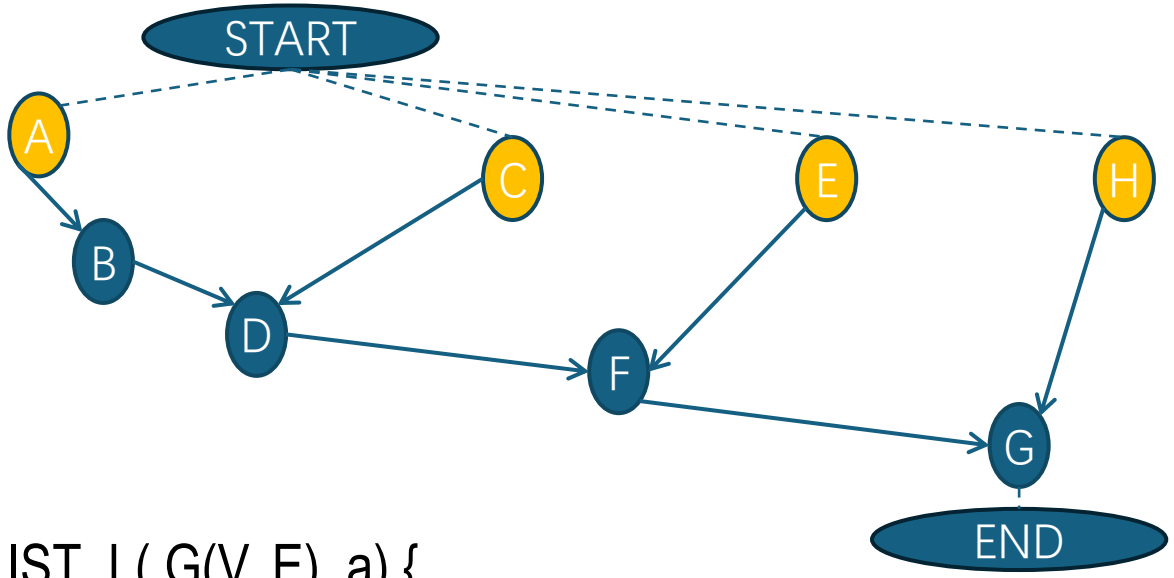


START 0
A 1
C 1

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	3
k	1
U	{B}
T	{}
S	{B}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

C 1

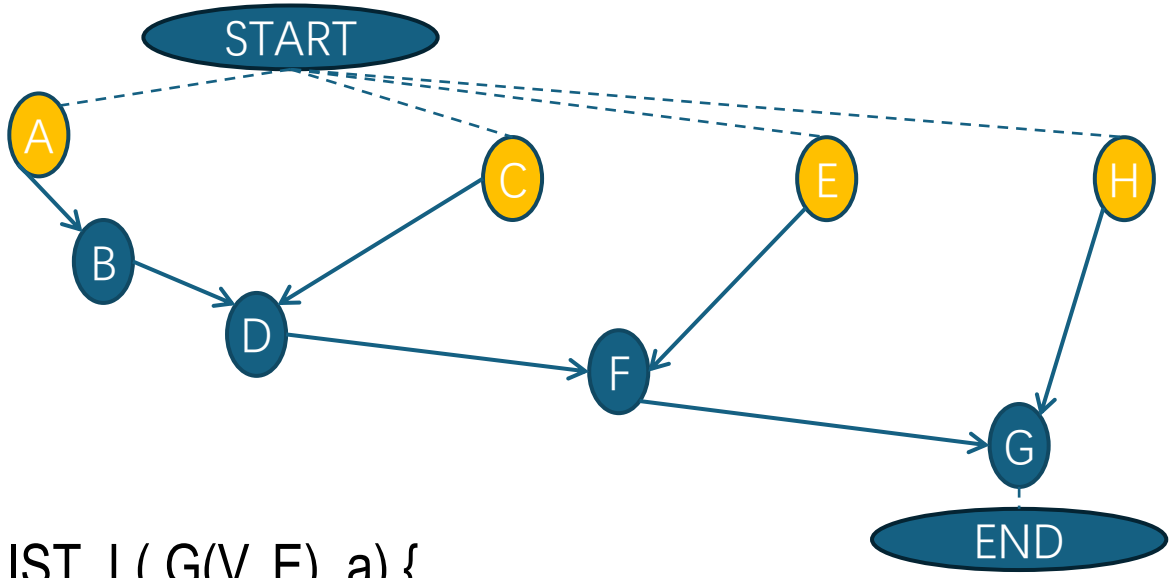
B 3

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	3
k	1
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

C 1

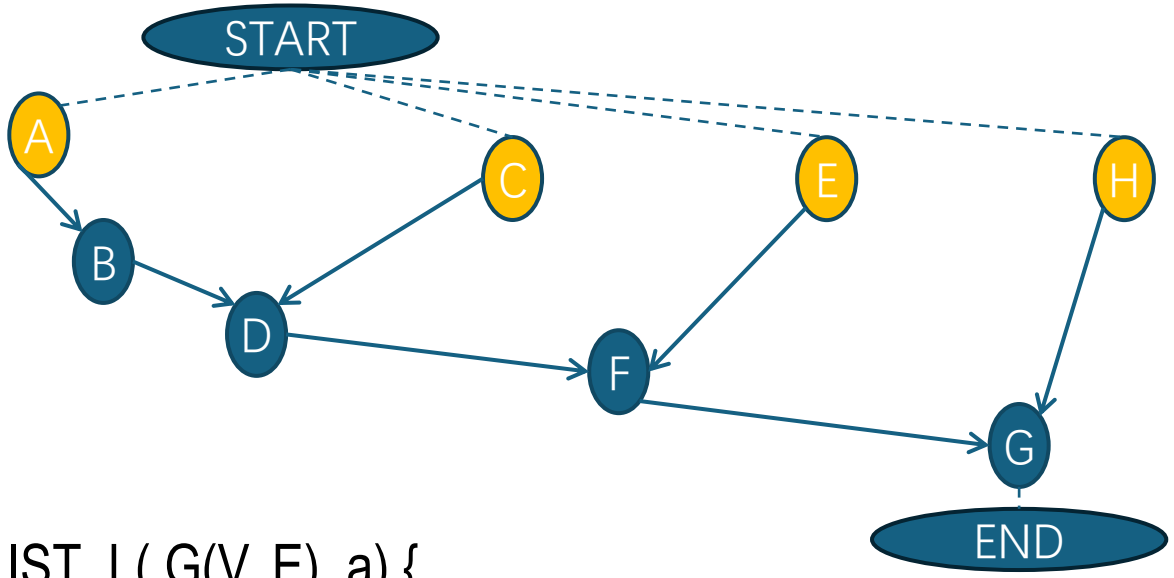
B 3

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	3
k	2
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4            对于每种资源类型 k = 1, 2, …, n_res {
5                  确定就绪的操作集 U_I,k;
6                  确定当前正在进行的操作集 T_I,k;
7                  选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                  在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9            }
10           I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

C 1

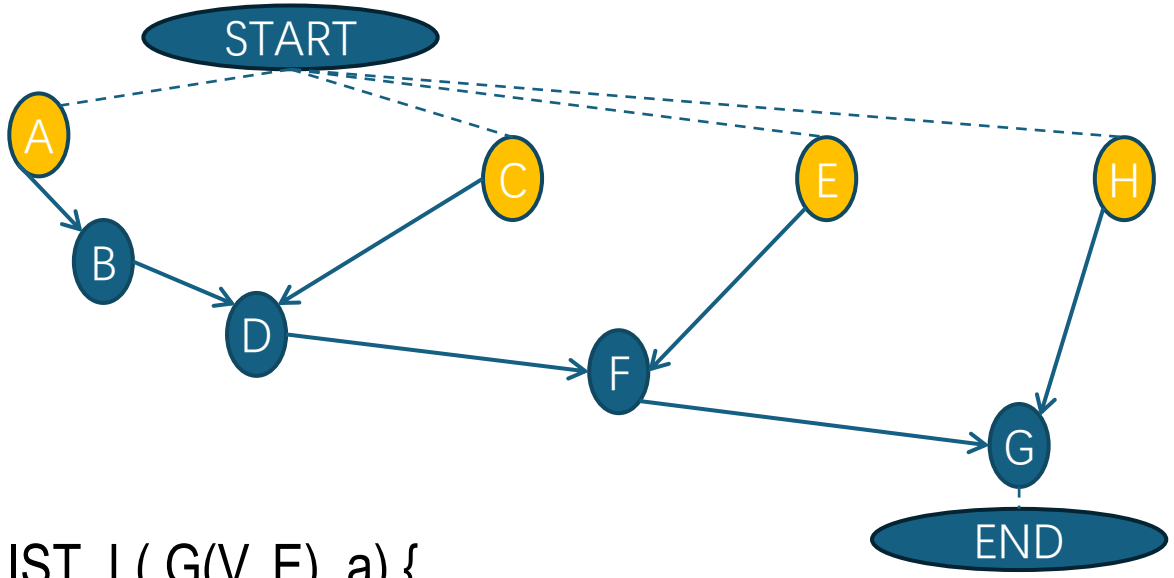
B 3

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	3
k	2
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4            对于每种资源类型 k = 1, 2, …, n_res {
5                  确定就绪的操作集 U_I,k;
6                  确定当前正在进行的操作集 T_I,k;
7                  选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                  在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9            }
10           I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

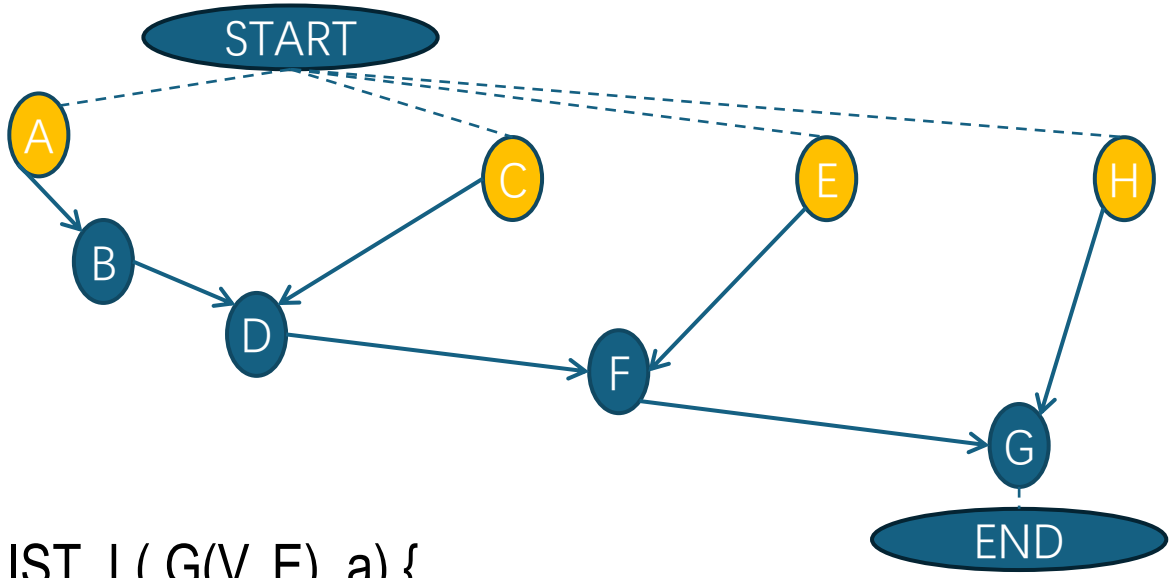


START 0
A 1
C 1
B 3

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	3
k	2
U	{E,H}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



START 0

A 1

C 1

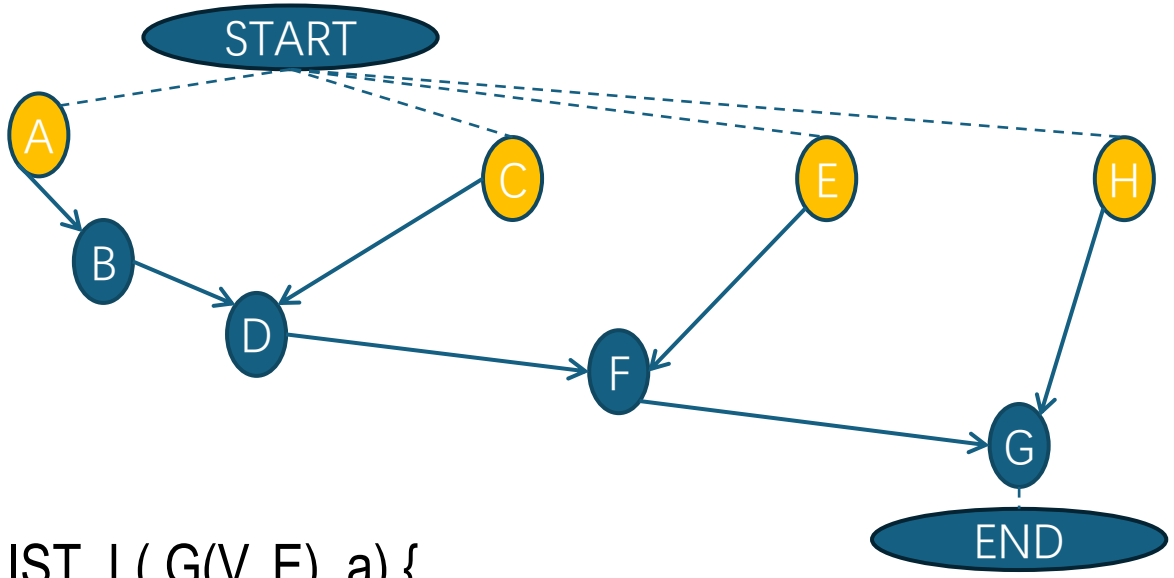
B 3

k=1为加法器，有一个

k=2为乘法器，有两个

变量	当前值
I	3
k	2
U	{E,H}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

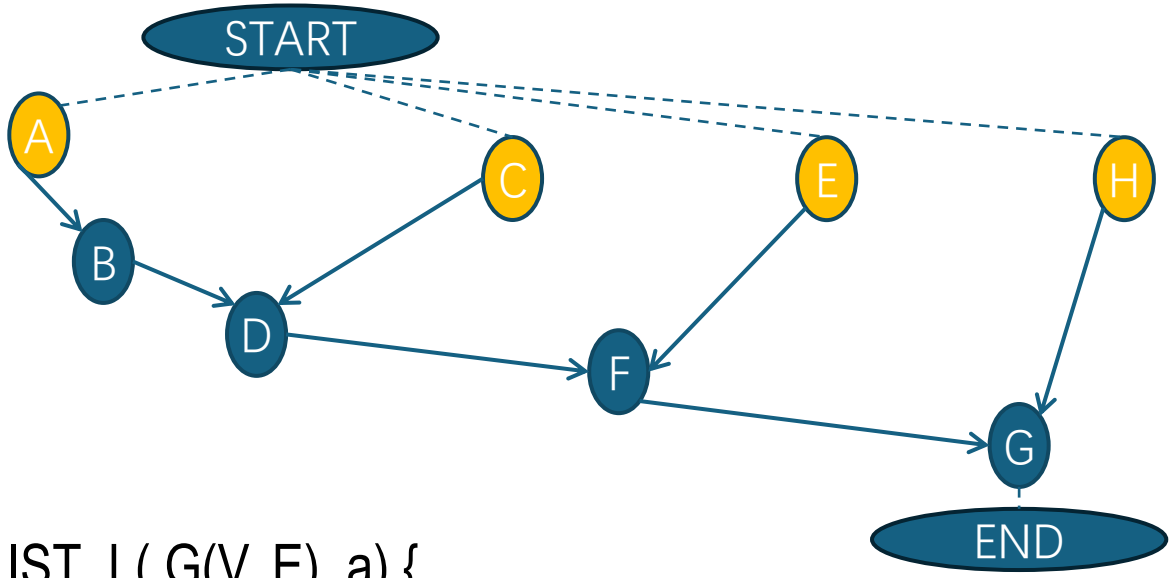


START 0
A 1
C 1
B 3

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	3
k	2
U	{E,H}
T	{}
S	{E,H}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

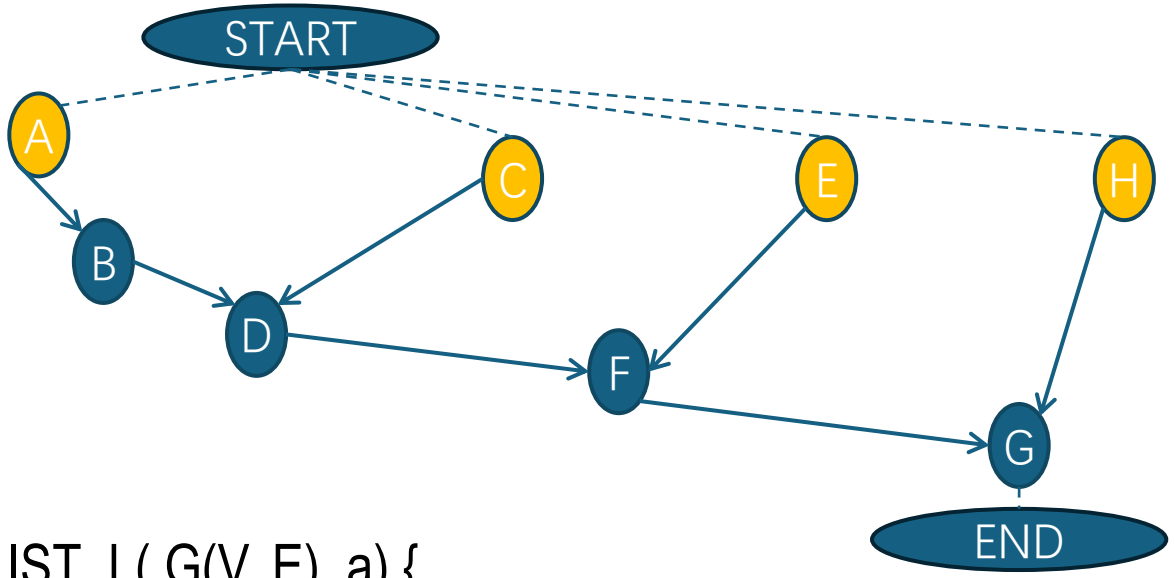


START 0
A 1
C 1
B 3
E 3
H 3

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	3
k	2
U	{E,H}
T	{}
S	{E,H}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

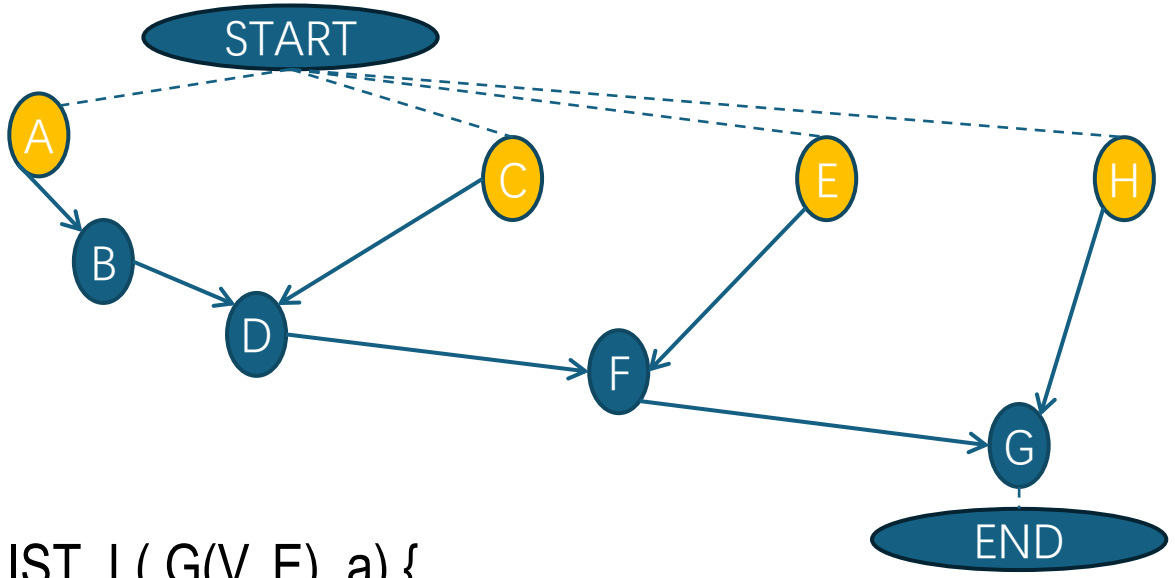


START 0
A 1
C 1
B 3
E 3
H 3

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	4
k	2
U	{E,H}
T	{}
S	{E,H}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

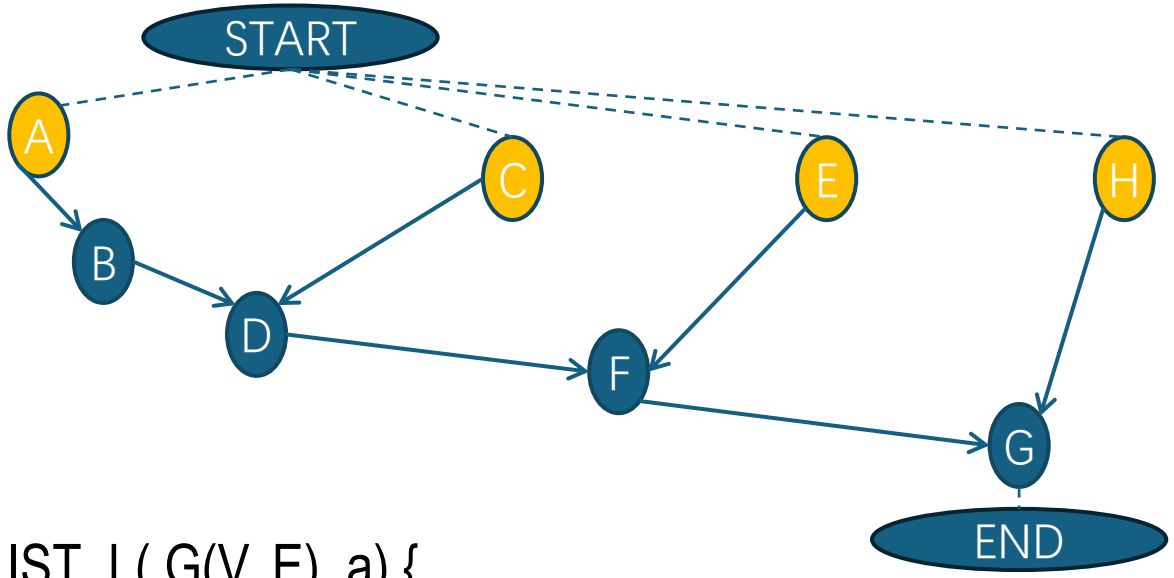


START 0
A 1
C 1
B 3
E 3
H 3

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	4
k	2
U	{E,H}
T	{}
S	{E,H}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

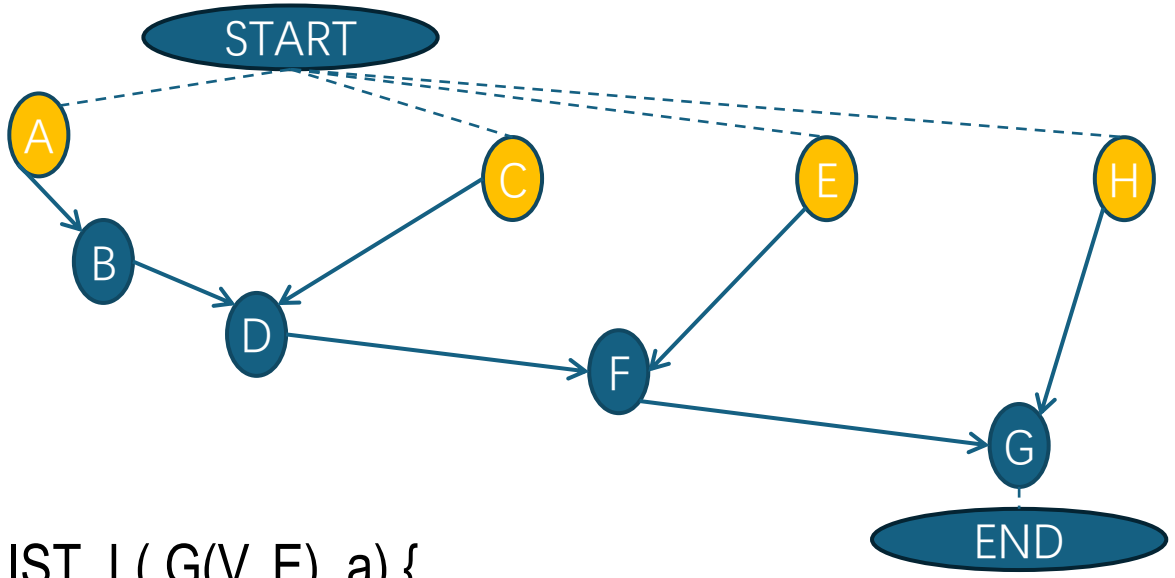


START 0
A 1
C 1
B 3
E 3
H 3

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	4
k	1
U	{E,H}
T	{}
S	{E,H}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4            对于每种资源类型 k = 1, 2, …, n_res {
5                  确定就绪的操作集 U_I,k;
6                  确定当前正在进行的操作集 T_I,k;
7                  选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                  在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9            }
10           I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

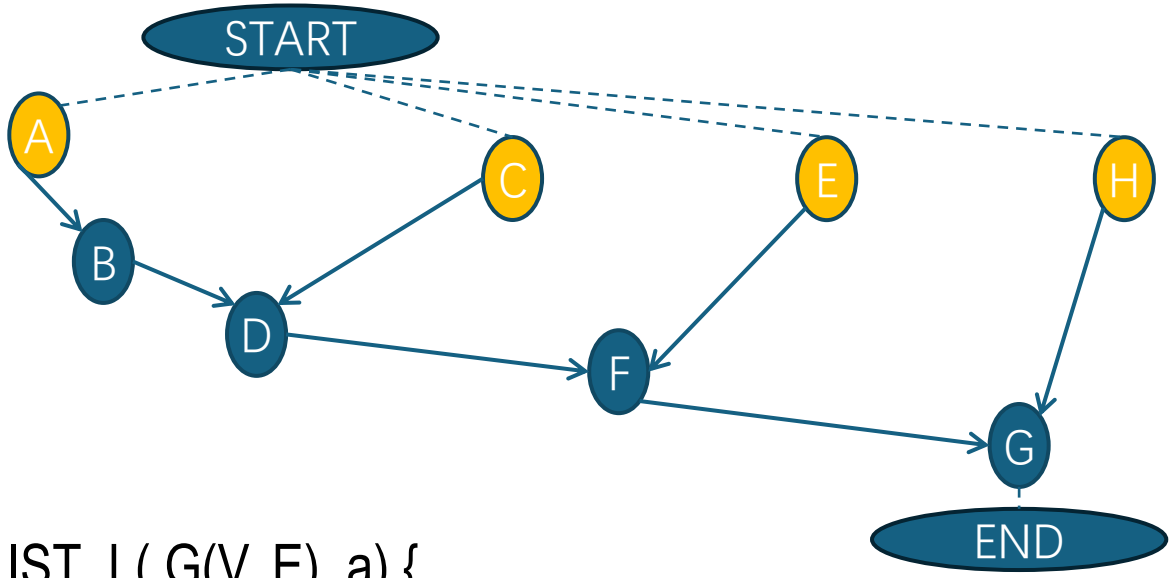


START 0
A 1
C 1
B 3
E 3
H 3

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	4
k	1
U	{D}
T	{}
S	{E,H}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

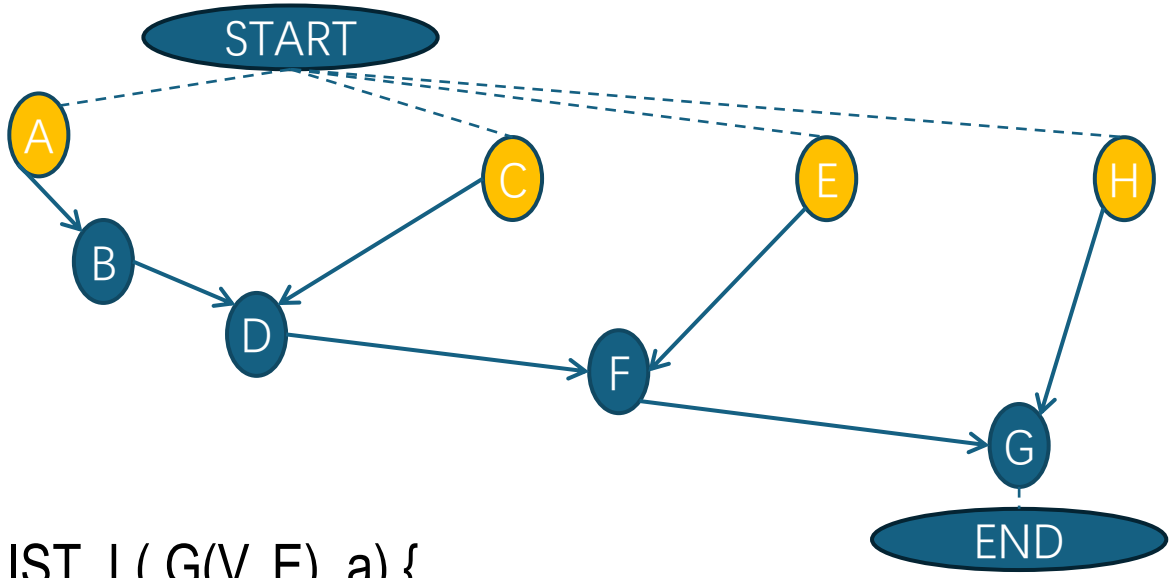


START 0
A 1
C 1
B 3
E 3
H 3

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
l	4
k	1
U	{D}
T	{}
S	{E,H}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  l = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_l,k;
6                 确定当前正在进行的操作集 T_l,k;
7                 选择一个子集S_k ⊆ U_l,k ,使得 |S_k| + |T_l,k| ≤ a_k;
8                 在步骤 l 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = l;
9             }
10            l = l + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

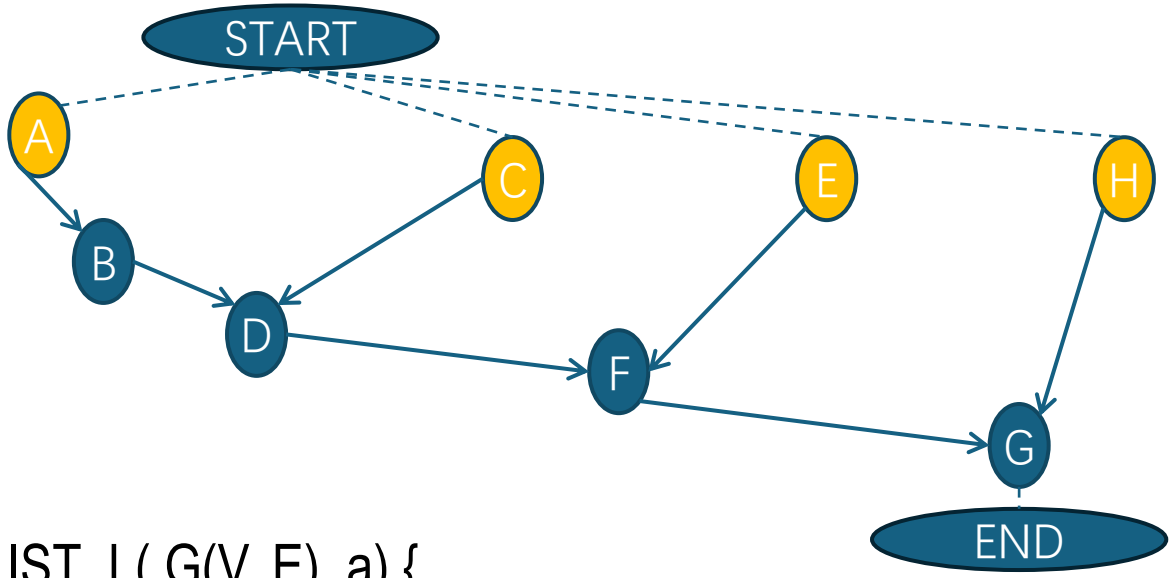


START 0
A 1
C 1
B 3
E 3
H 3

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
l	4
k	1
U	{D}
T	{}
S	{D}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  l = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_l,k;
6                 确定当前正在进行的操作集 T_l,k;
7                 选择一个子集S_k ⊆ U_l,k ,使得 |S_k| + |T_l,k| ≤ a_k;
8                 在步骤 l 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = l;
9             }
10            l = l + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

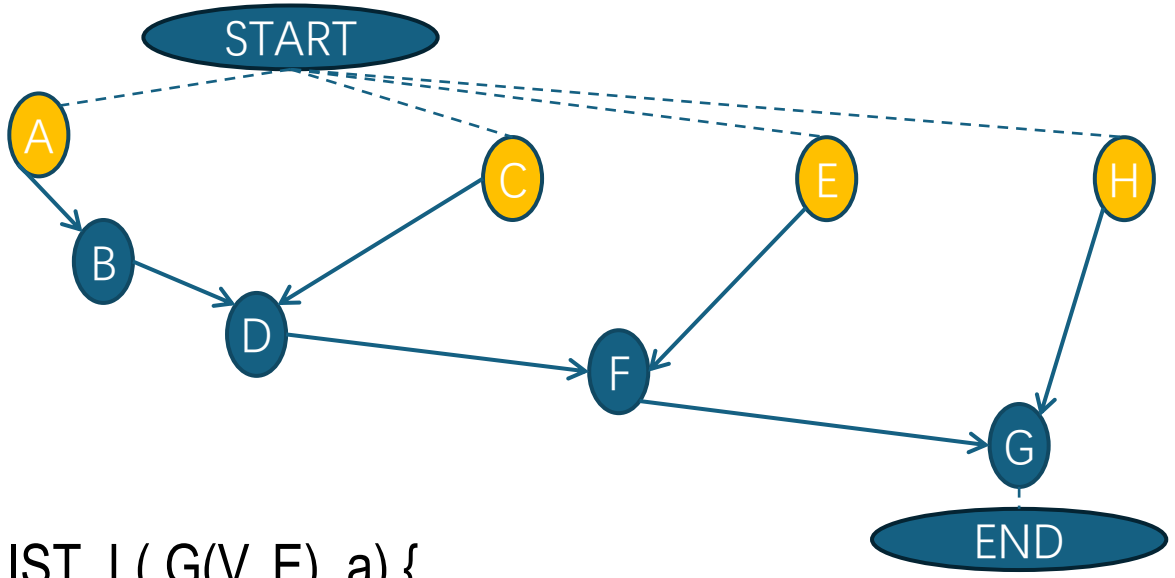


START 0
A 1
C 1
B 3
E 3
H 3
D 4

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	4
k	1
U	{D}
T	{}
S	{D}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

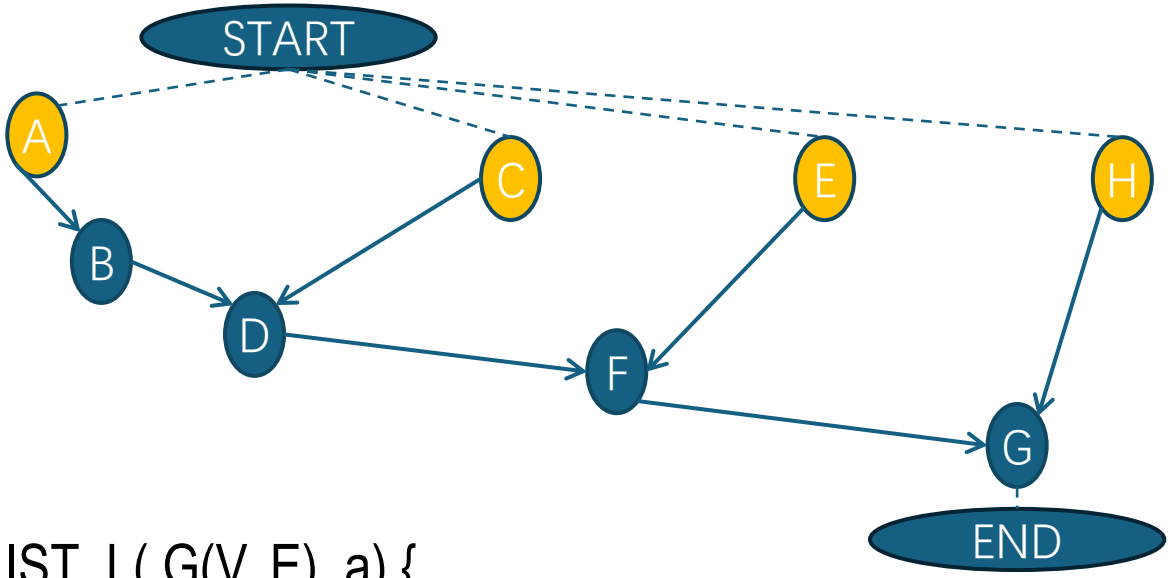


START 0
A 1
C 1
B 3
E 3
H 3
D 4

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	4
k	2
U	{D}
T	{}
S	{D}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4            对于每种资源类型 k = 1, 2, …, n_res {
5                  确定就绪的操作集 U_{I,k};
6                  确定当前正在进行的操作集 T_{I,k};
7                  选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                  在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9            }
10           I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

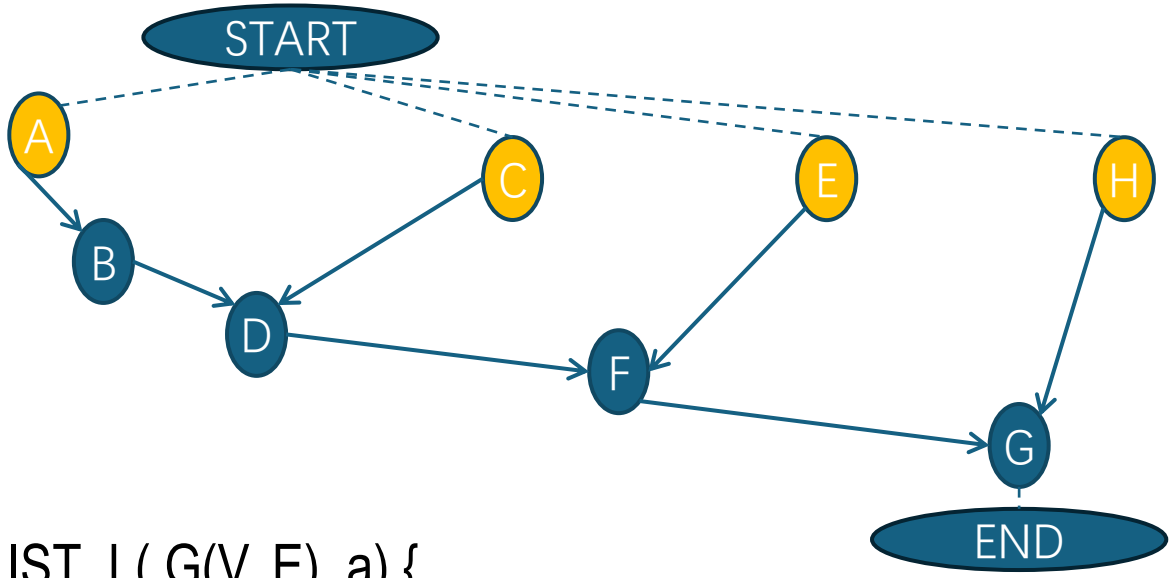


START 0
A 1
C 1
B 3
E 3
H 3
D 4

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	4
k	2
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

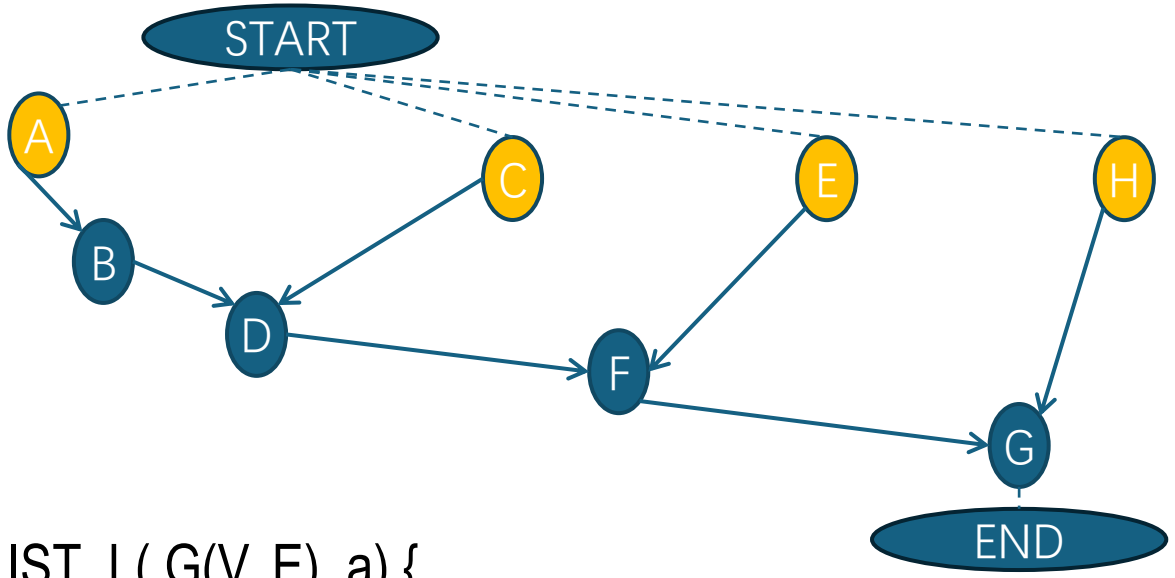


START 0
A 1
C 1
B 3
E 3
H 3
D 4

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	5
k	2
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

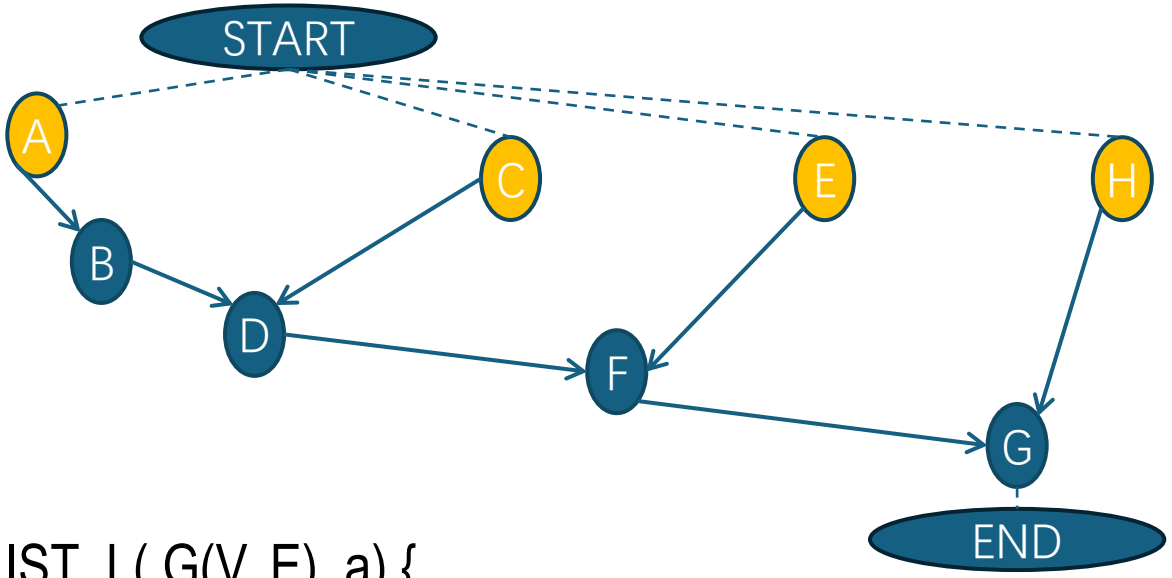


START 0
A 1
C 1
B 3
E 3
H 3
D 4

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	5
k	1
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

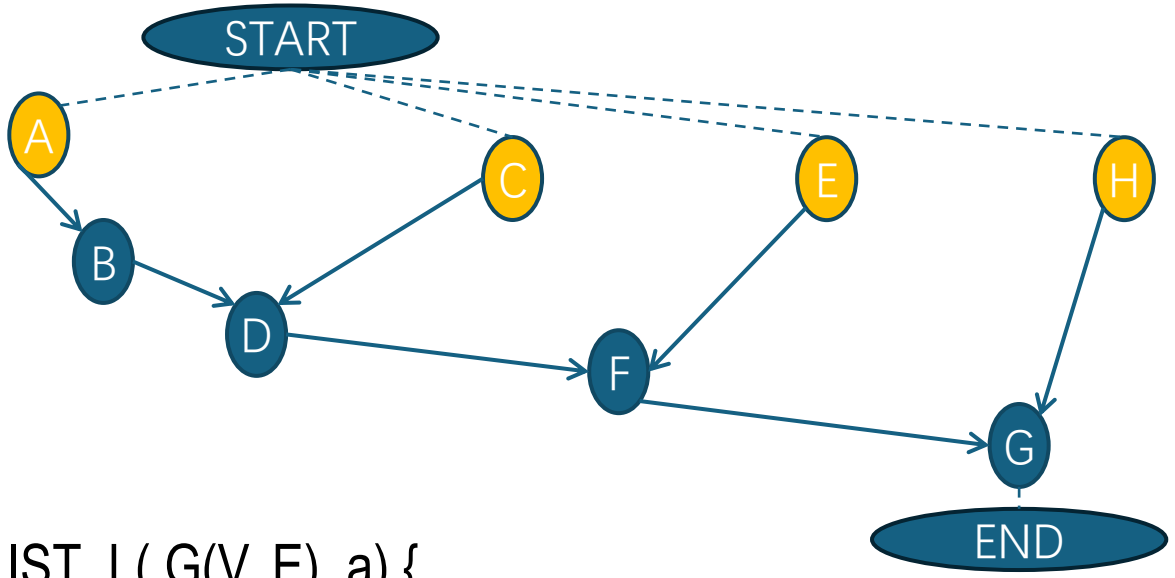


START 0
A 1
C 1
B 3
E 3
H 3
D 4
F 5

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	5
k	1
U	{F}
T	{}
S	{F}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

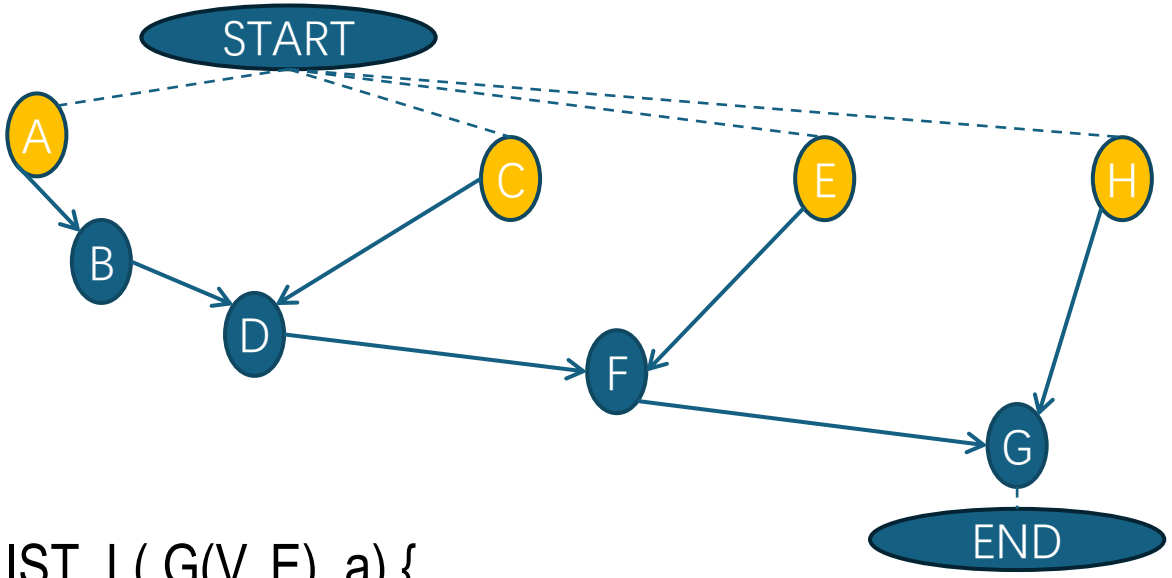


START 0
A 1
C 1
B 3
E 3
H 3
D 4
F 5

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	5
k	2
U	{F}
T	{}
S	{F}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4            对于每种资源类型 k = 1, 2, …, n_res {
5                  确定就绪的操作集 U_{I,k};
6                  确定当前正在进行的操作集 T_{I,k};
7                  选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                  在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9            }
10           I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

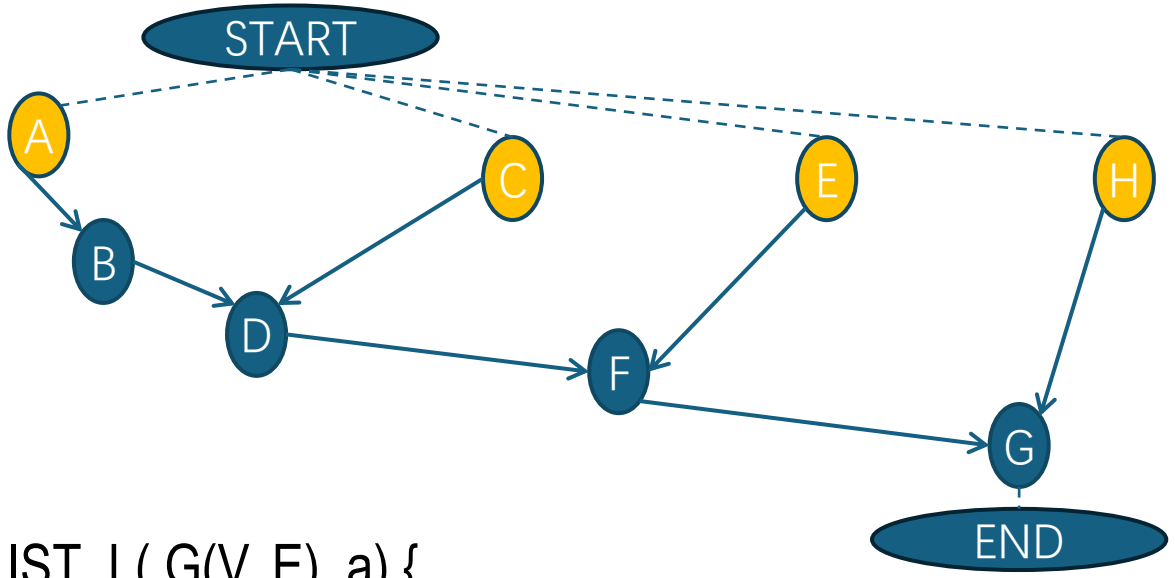


START 0
A 1
C 1
B 3
E 3
H 3
D 4
F 5

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	5
k	2
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

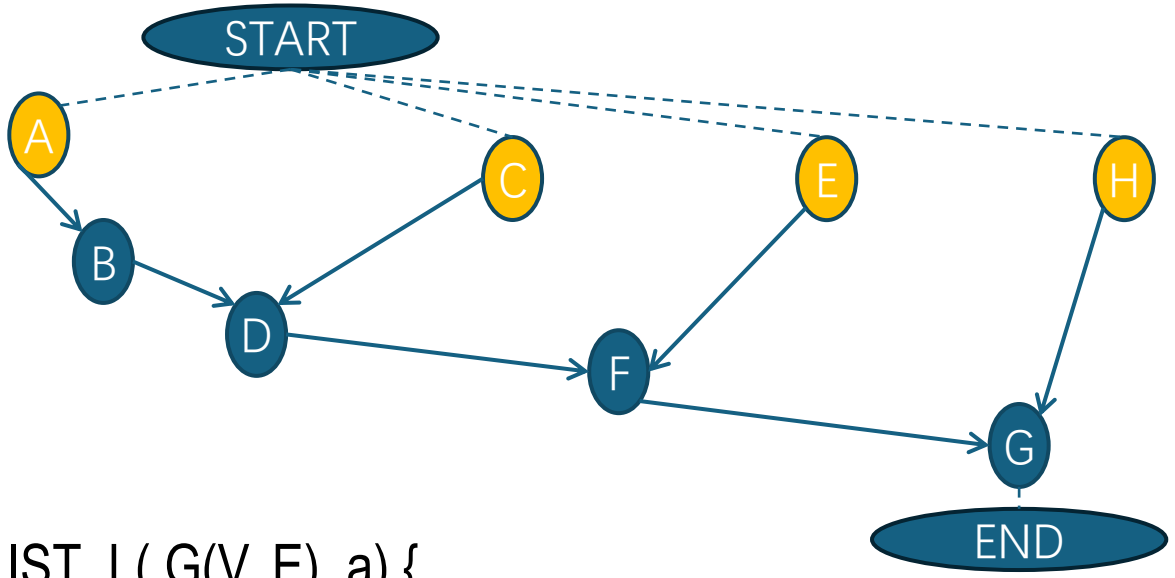


START 0
A 1
C 1
B 3
E 3
H 3
D 4
F 5

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	6
k	2
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

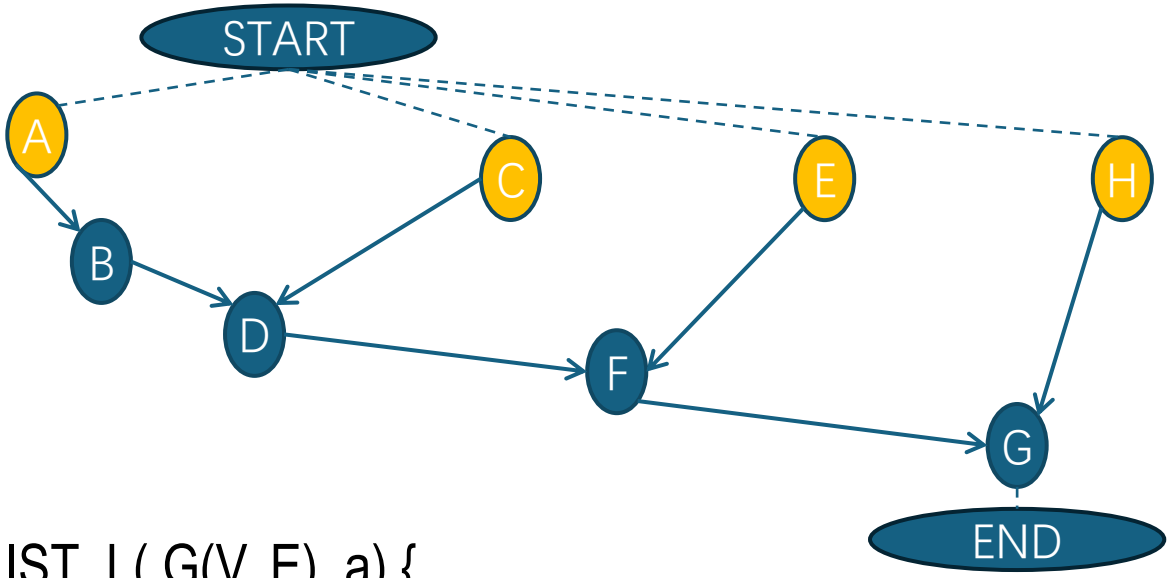


START 0
A 1
C 1
B 3
E 3
H 3
D 4
F 5

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	6
k	1
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

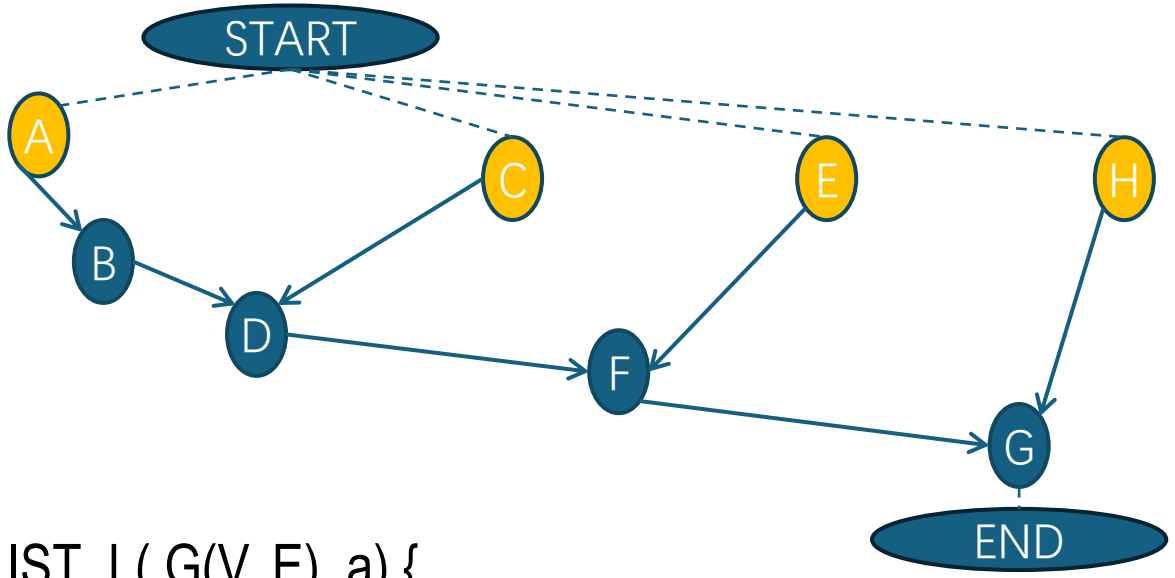


START 0
A 1
C 1
B 3
E 3
H 3
D 4
F 5
G 6

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	6
k	1
U	{G}
T	{}
S	{G}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

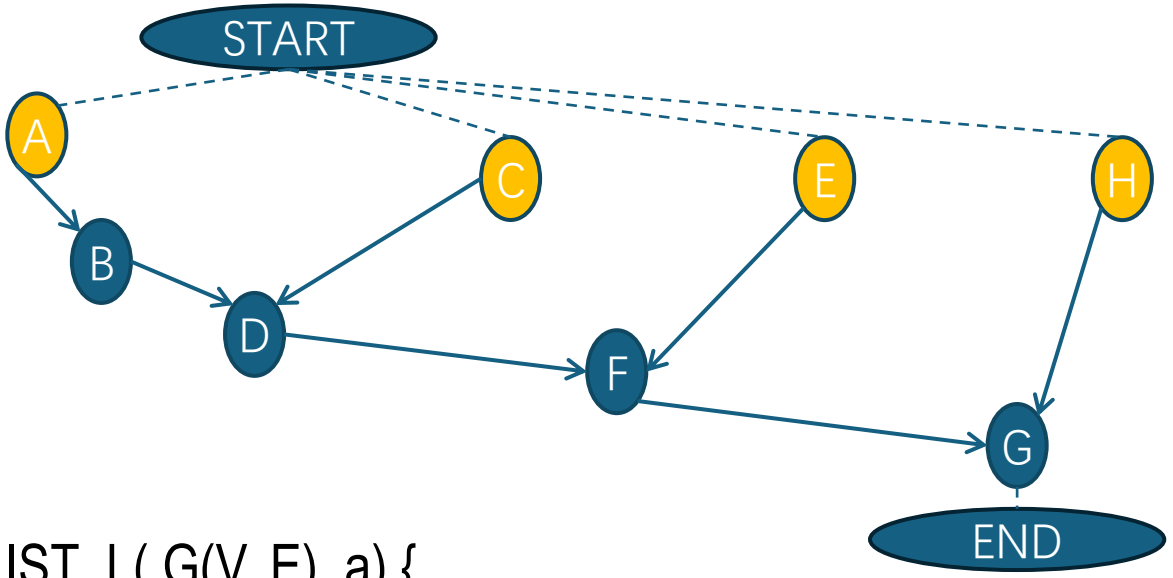


START 0
A 1
C 1
B 3
E 3
H 3
D 4
F 5
G 6

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	6
k	2
U	{G}
T	{}
S	{G}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4            对于每种资源类型 k = 1, 2, …, n_res {
5                  确定就绪的操作集 U_{I,k};
6                  确定当前正在进行的操作集 T_{I,k};
7                  选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                  在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9            }
10           I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

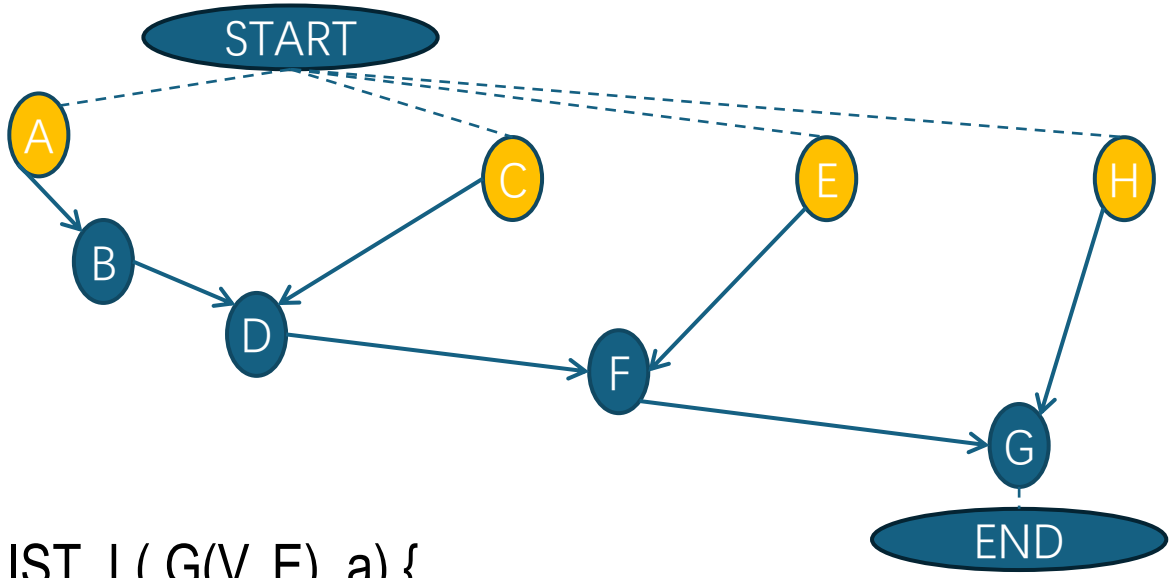


START 0
A 1
C 1
B 3
E 3
H 3
D 4
F 5
G 6

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	6
k	2
U	{}
T	{}
S	{}

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_I,k;
6                 确定当前正在进行的操作集 T_I,k;
7                 选择一个子集S_k ⊆ U_I,k ,使得 |S_k| + |T_I,k| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```



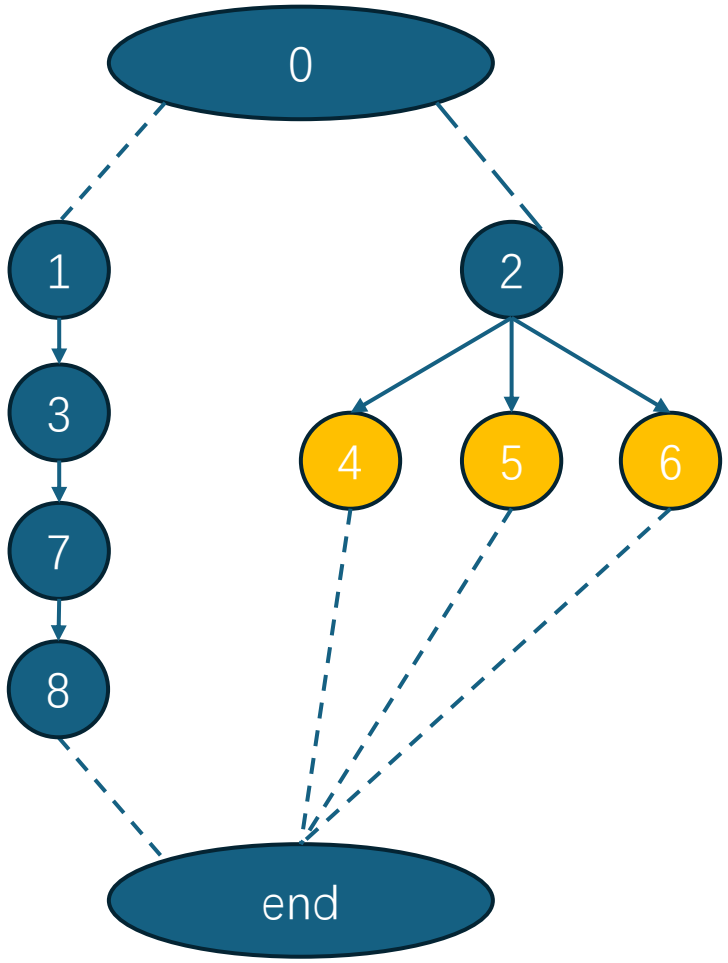
START 0
A 1
C 1
B 3
E 3
H 3
D 4
F 5
G 6
END 7

k=1为加法器，有一个
k=2为乘法器，有两个

变量	当前值
I	6
k	2
U	{}
T	{}
S	{}

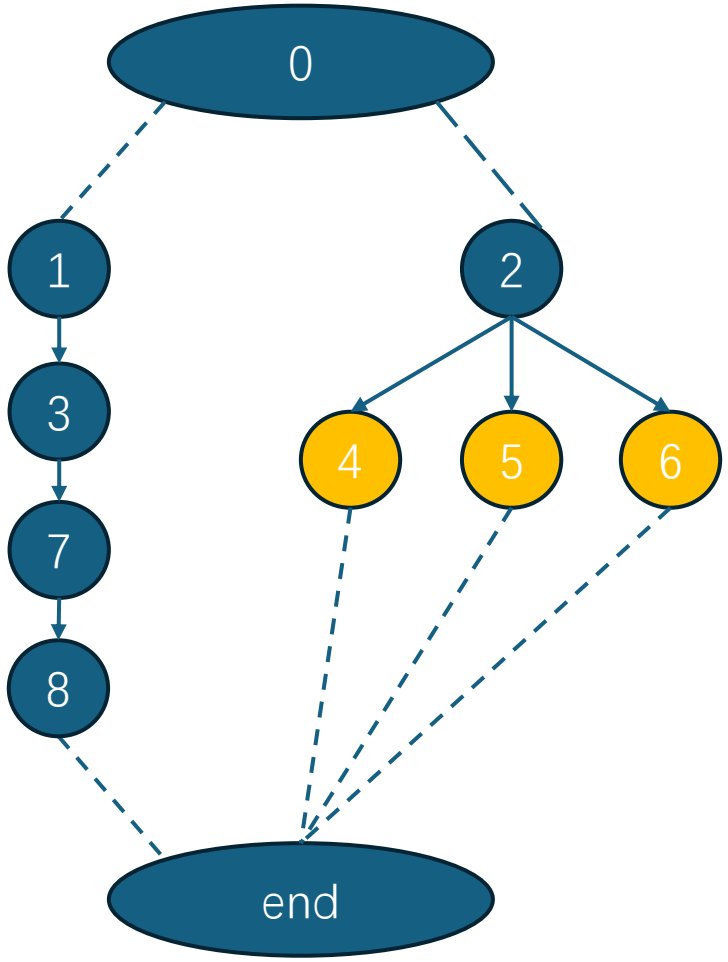
```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  I = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, ..., n_res {
5                 确定就绪的操作集 U_{I,k};
6                 确定当前正在进行的操作集 T_{I,k};
7                 选择一个子集S_k ⊆ U_{I,k} ,使得 |S_k| + |T_{I,k}| ≤ a_k;
8                 在步骤 I 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = I;
9             }
10            I = I + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

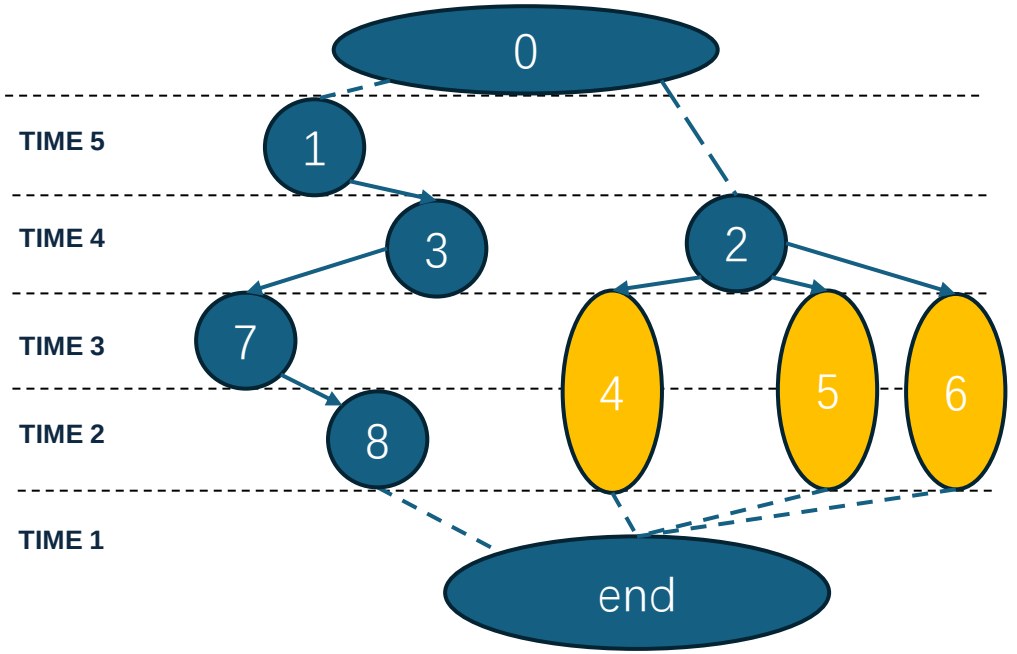
能不能按照离终点的距离选择结点执行顺序?



能不能按照离终点的距离选择结点执行顺序?

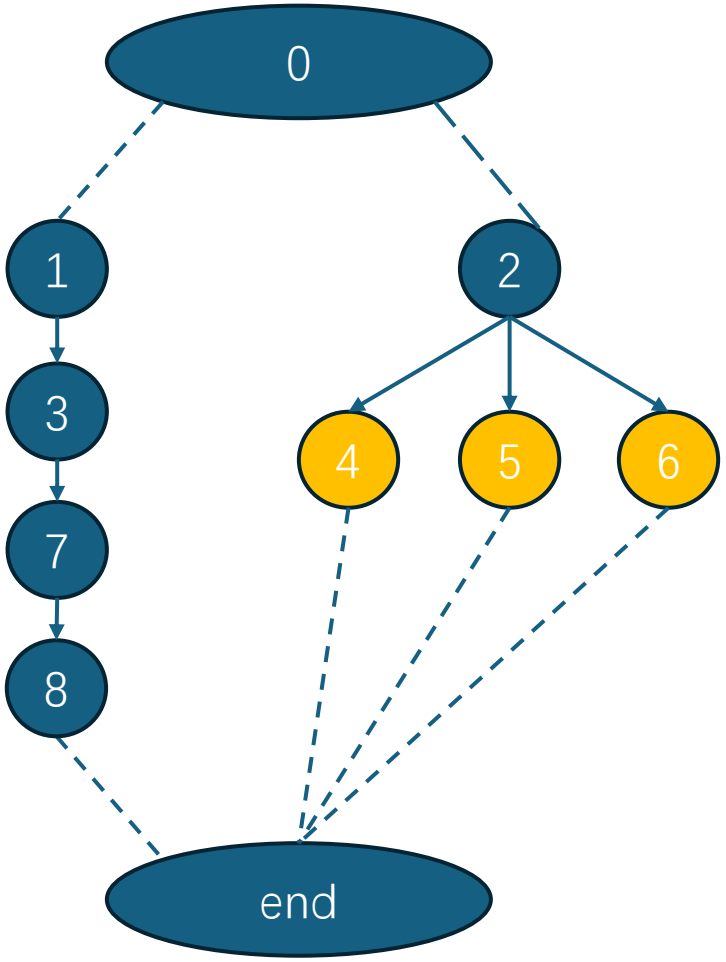
有一个加法器和一个乘法器，
乘法器延迟为2，加法器延迟为1

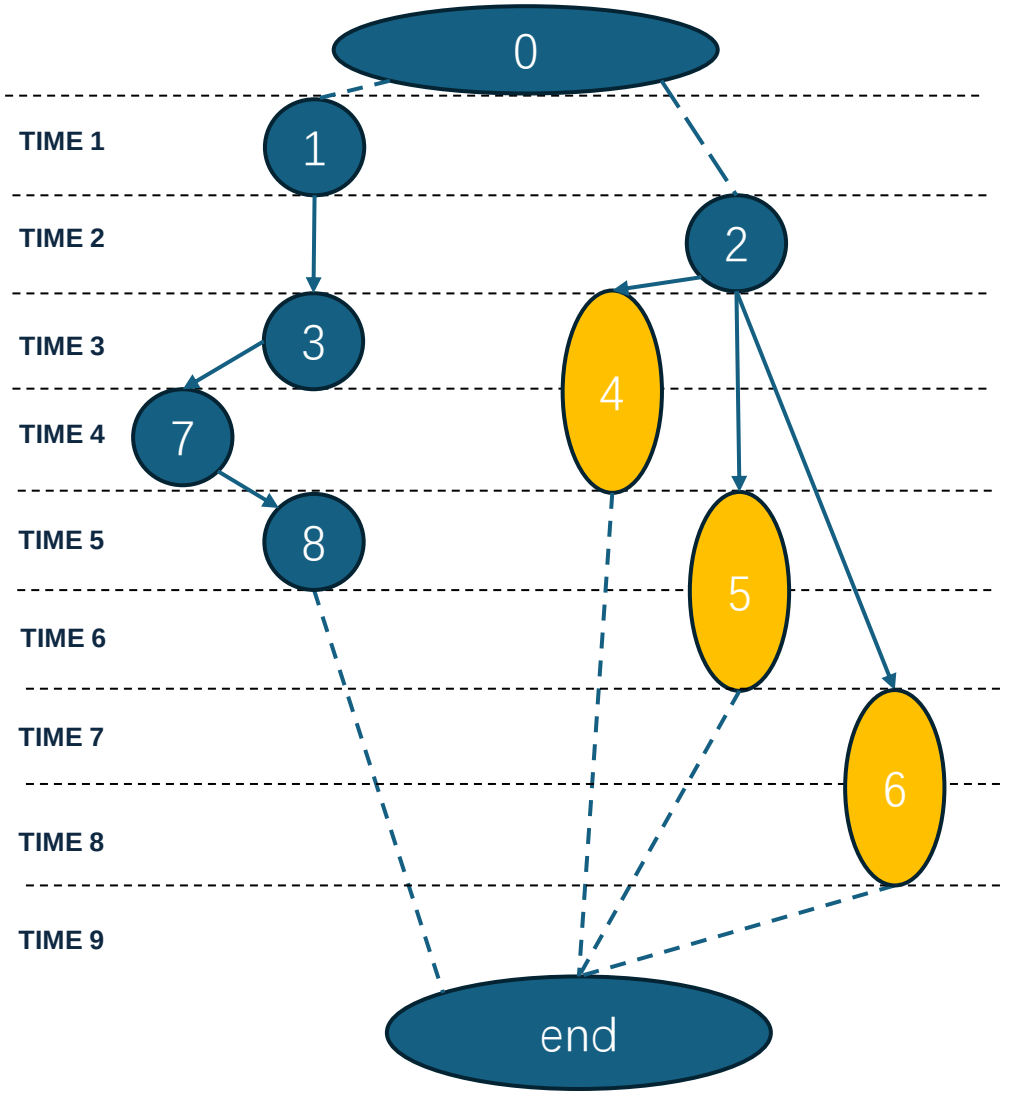




1 5
3 4
7 3
8 2
2 4
4 3
5 3
6 3

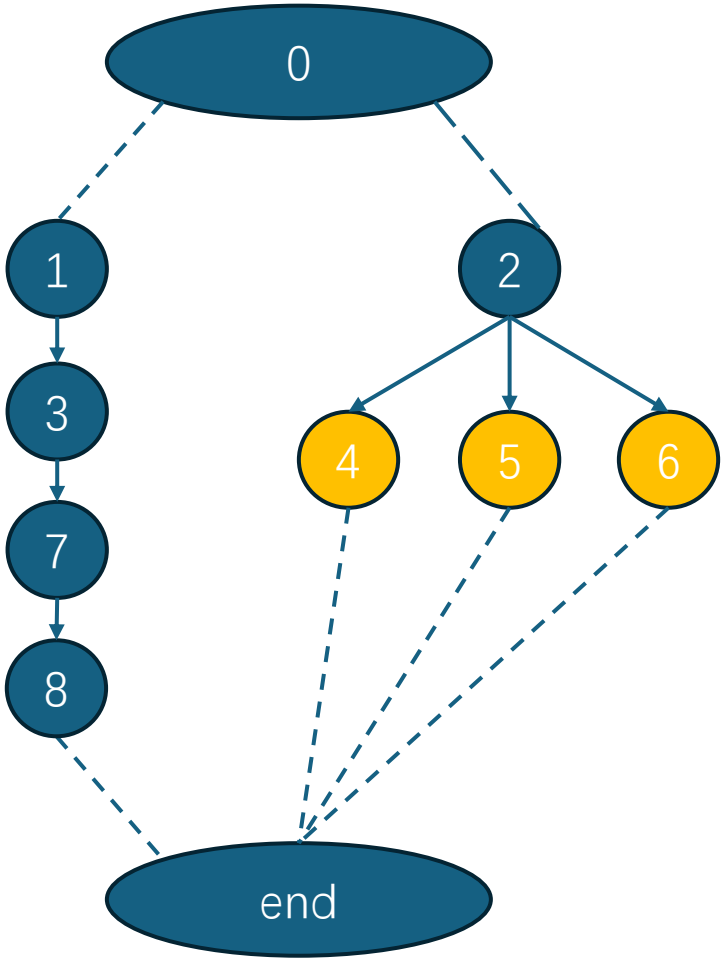
有一个加法器和一个乘法器，
乘法器延迟为2，加法器延迟为1

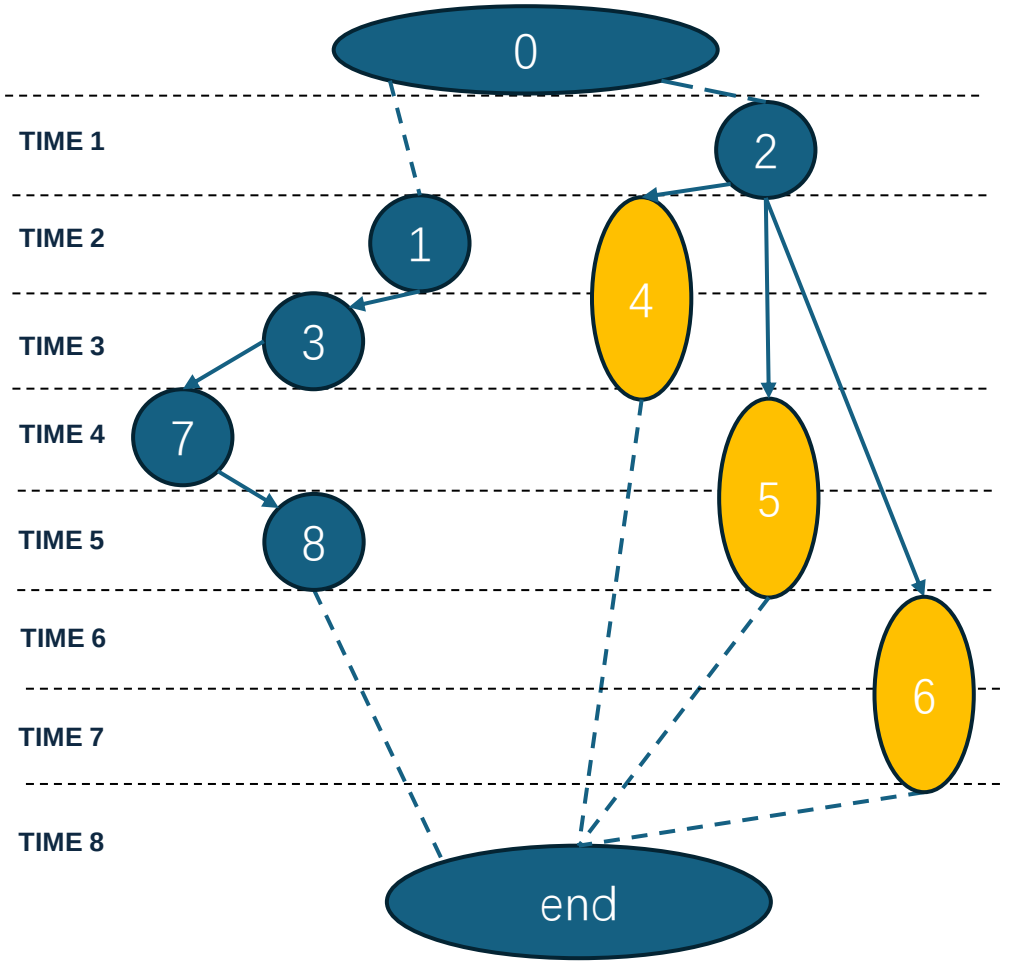




1 5
3 4
7 3
8 2
2 4
4 3
5 3
6 3

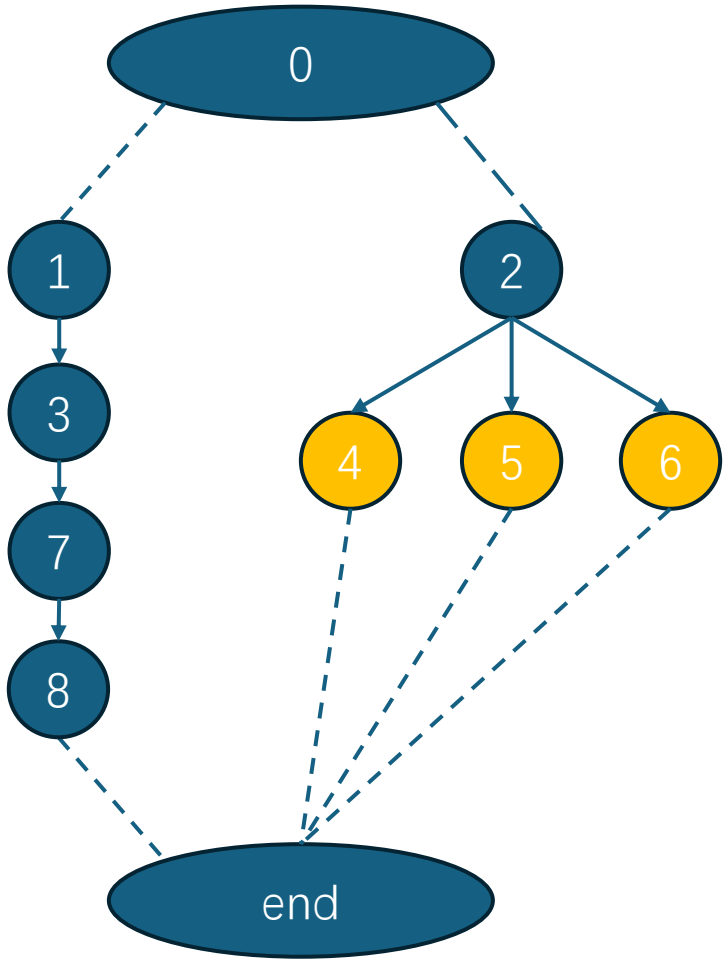
有一个加法器和一个乘法器，
乘法器延迟为2，加法器延迟为1





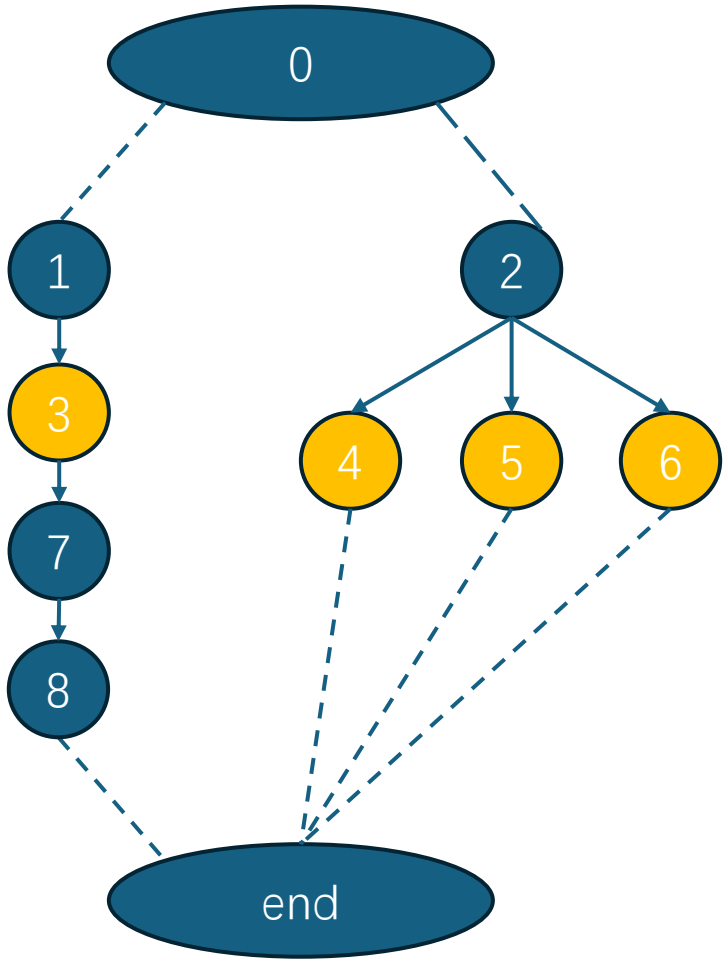
1 5
3 4
7 3
8 2
2 4
4 3
5 3
6 3

有一个加法器和一个乘法器，
乘法器延迟为2，加法器延迟为1



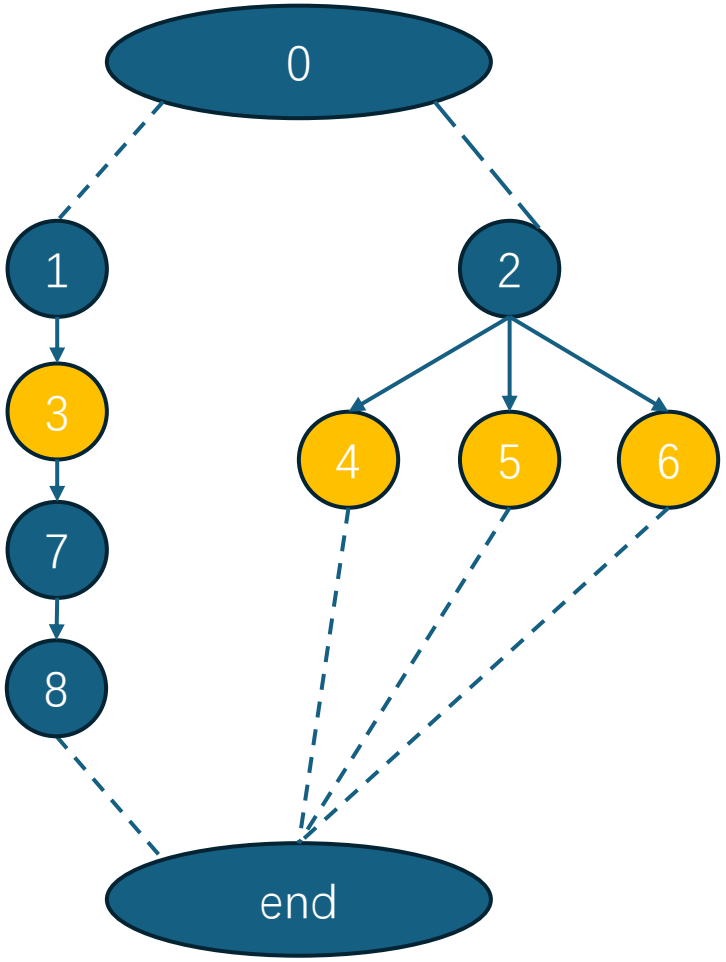
能不能按照子节点延迟之和选择结点顺序?

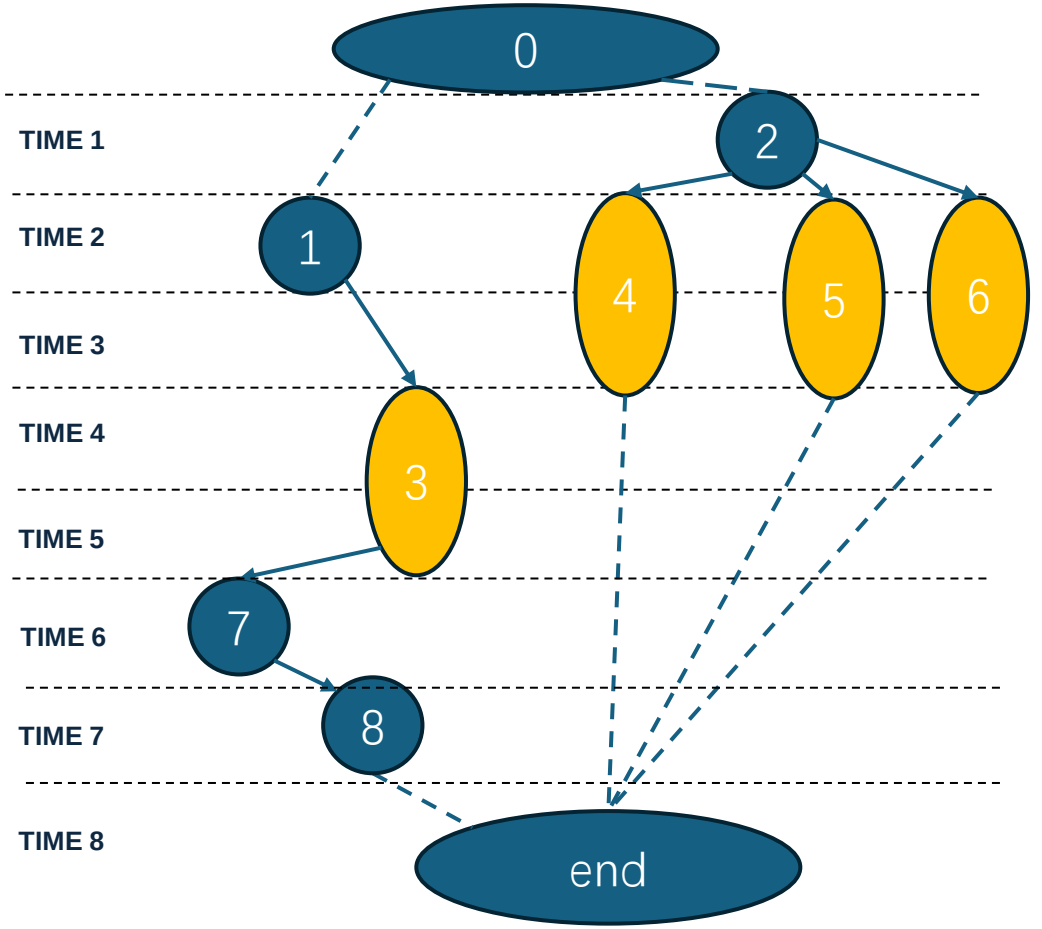
有一个加法器和一个乘法器，
乘法器延迟为2，加法器延迟为1



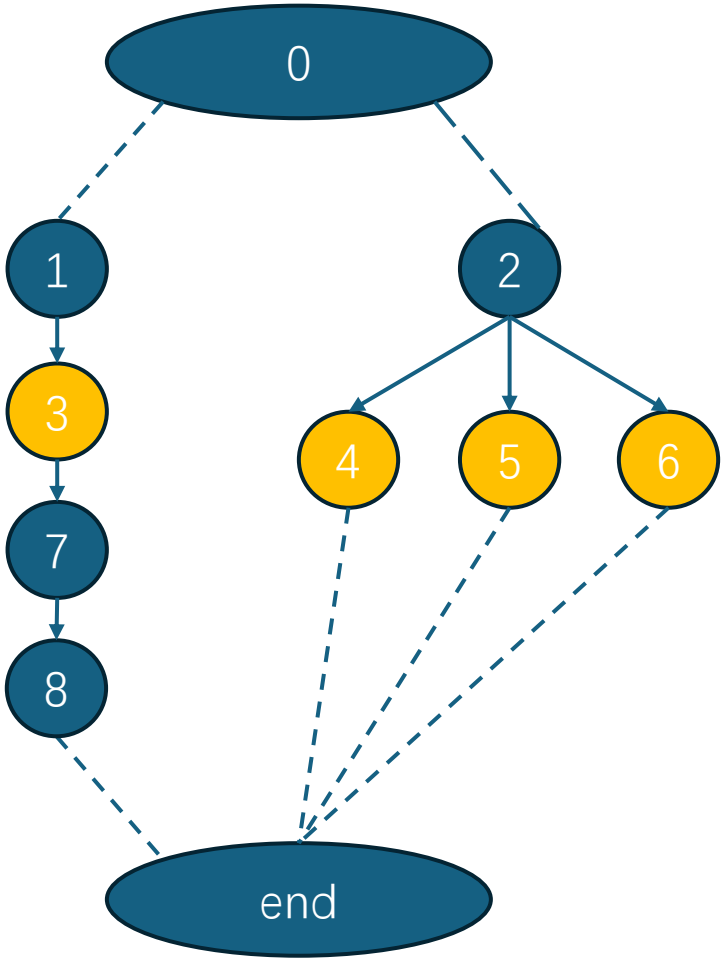
能不能按照子节点延迟之和选择结点顺序?

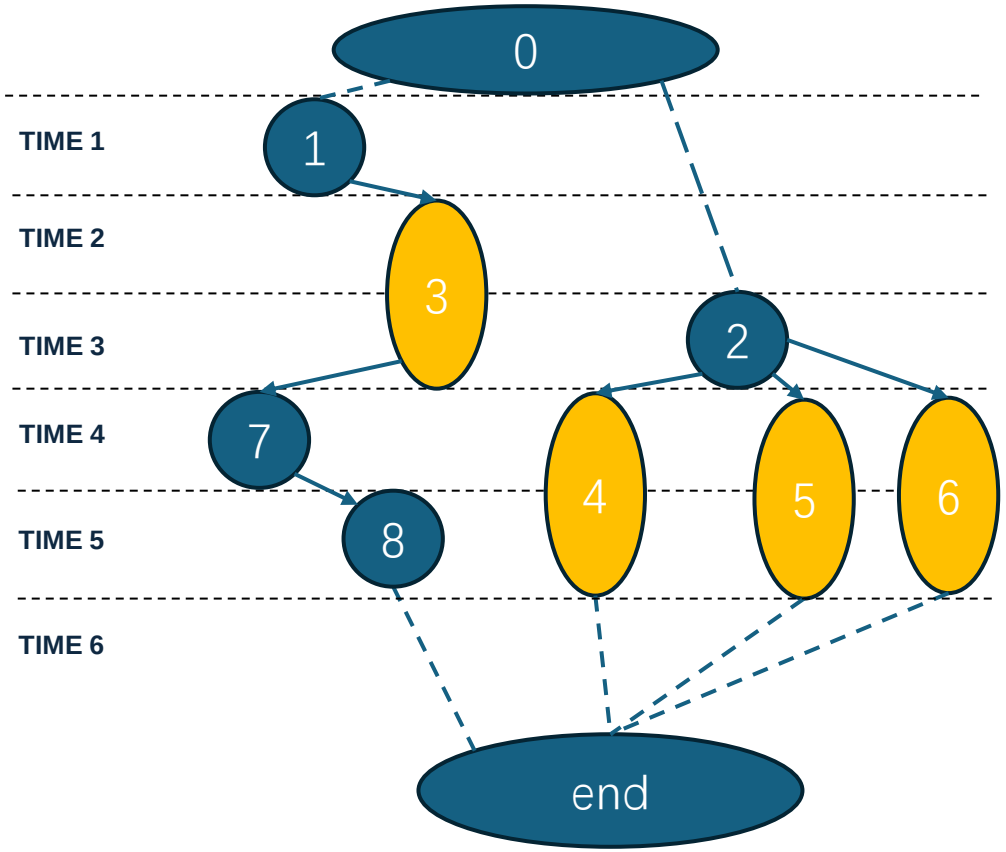
有一个加法器和三个乘法器，
乘法器延迟为2，加法器延迟为1





有一个加法器和三个乘法器，
乘法器延迟为2，加法器延迟为1





有一个加法器和三个乘法器，
乘法器延迟为2，加法器延迟为1

```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  l = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_l,k;
6                 确定当前正在进行的操作集 T_l,k;
7                 选择一个子集S_k ⊆ U_l,k ,使得 |S_k| + |T_l,k| ≤ a_k;
8                 在步骤 l 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = l;
9             }
10            l = l + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

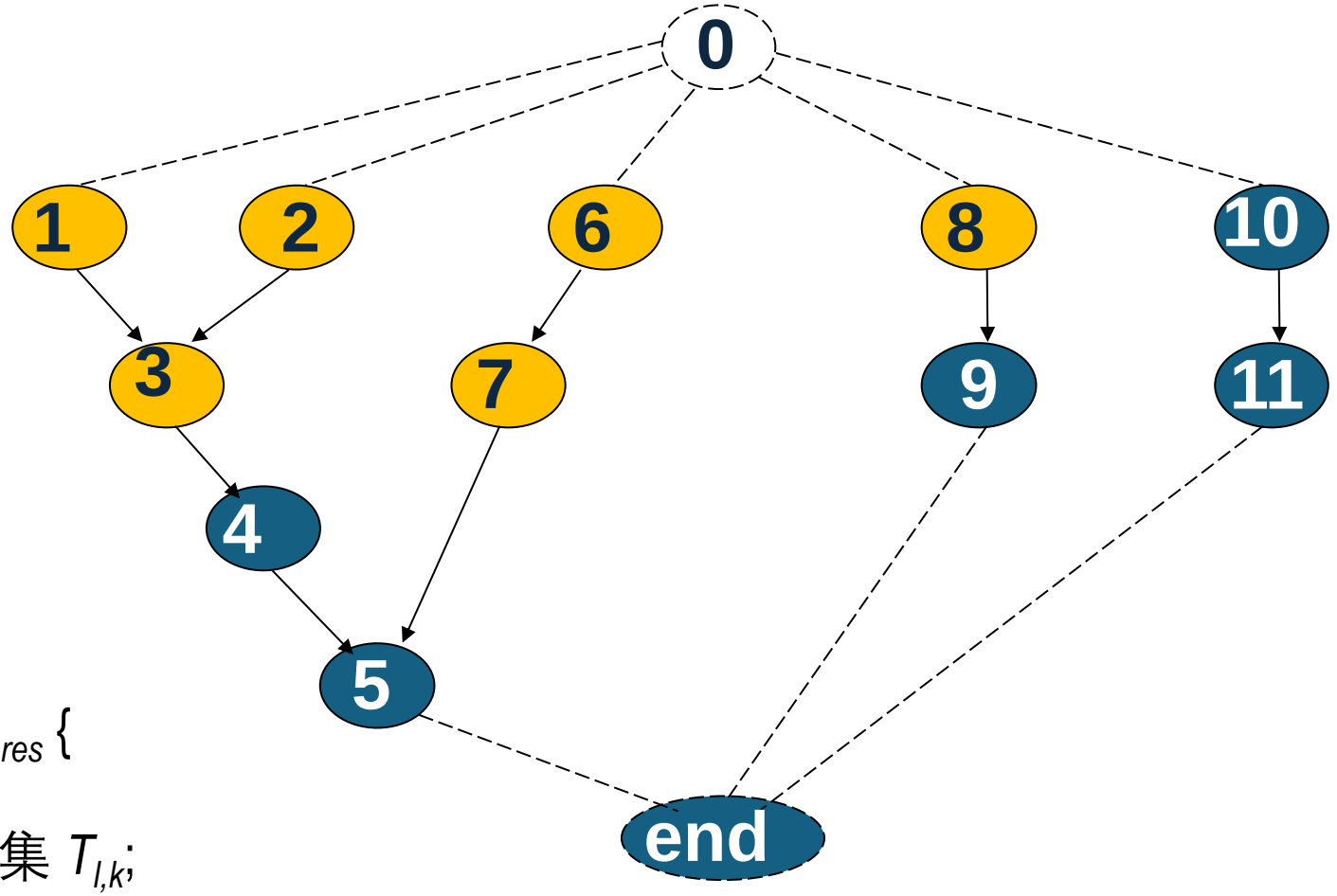
随堂作业

in-class assignment

3个乘法器，延迟为2

2个加法器，延迟为1

请写出最差情况下的调度结果




```
1 LIST_L( G(V, E), a) {
2         t0 = 0;  l = 1;
3         重复执行以下步骤 {
4             对于每种资源类型 k = 1, 2, …, n_res {
5                 确定就绪的操作集 U_l,k;
6                 确定当前正在进行的操作集 T_l,k;
7                 选择一个子集S_k ⊆ U_l,k ,使得 |S_k| + |T_l,k| ≤ a_k;
8                 在步骤 l 处调度 S 中的所有操作，即对所有 vi ∈ S 设定调度时间 ti = l;
9             }
10            l = l + 1;
11        }
12        直到 vn 被调度 ;
13 }
```

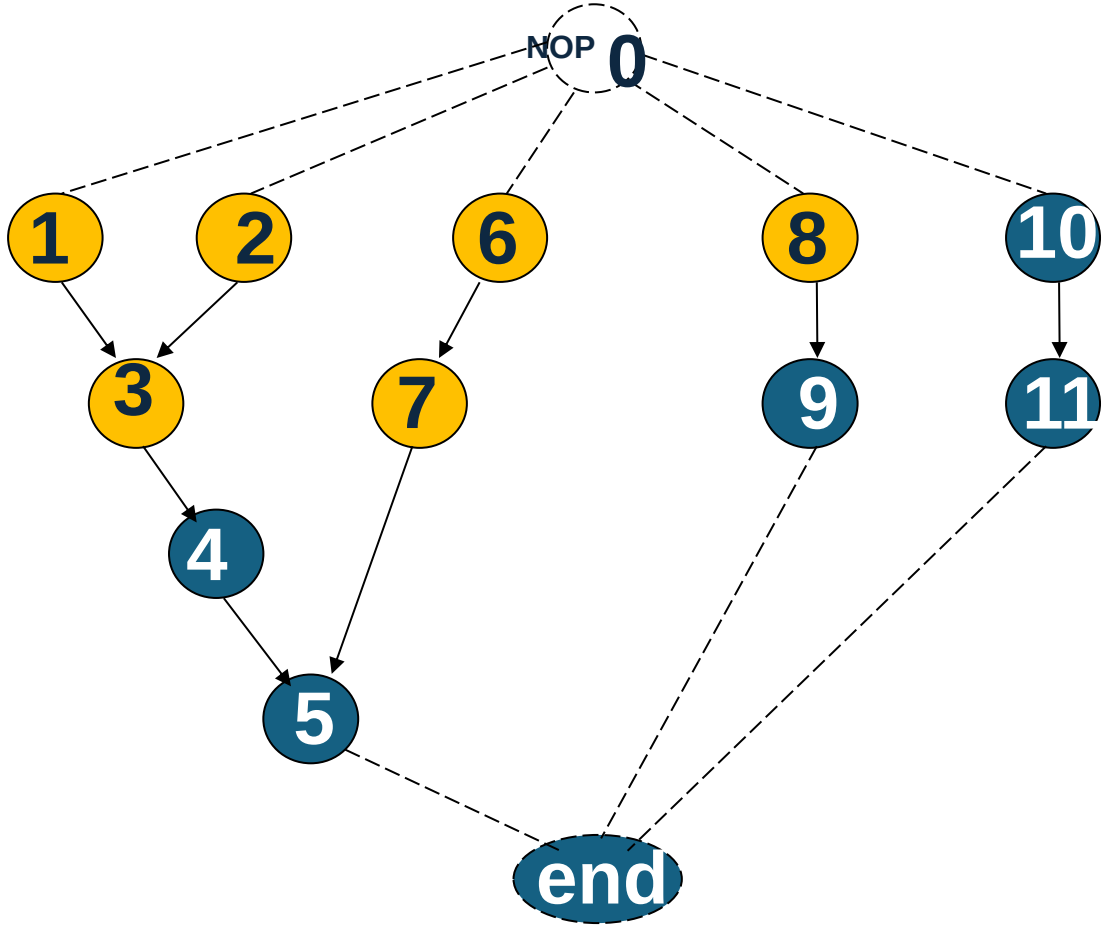
3个乘法器，延迟为2

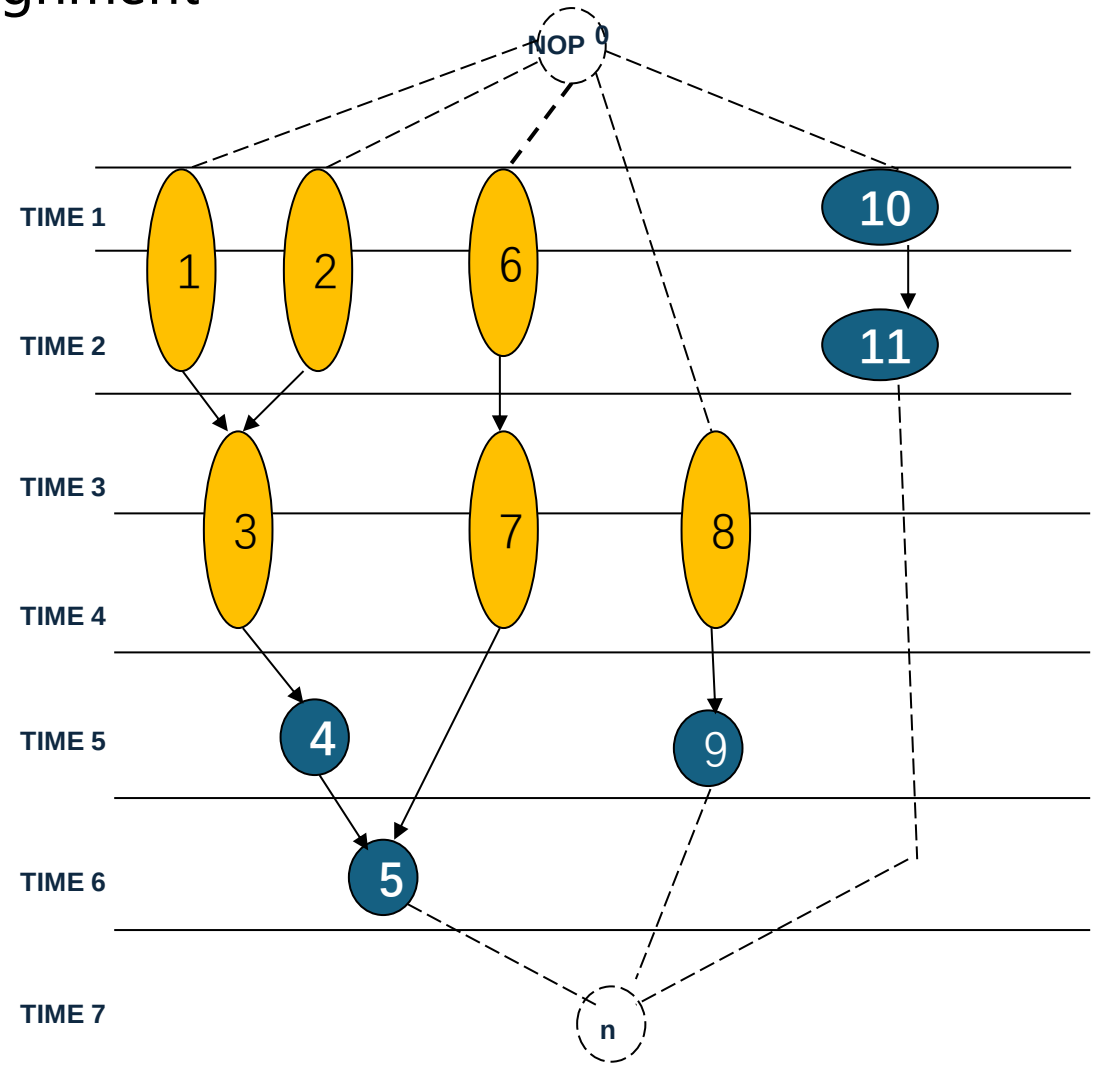
2个加法器，延迟为1

请写出最差情况下的调度结果

随堂作业

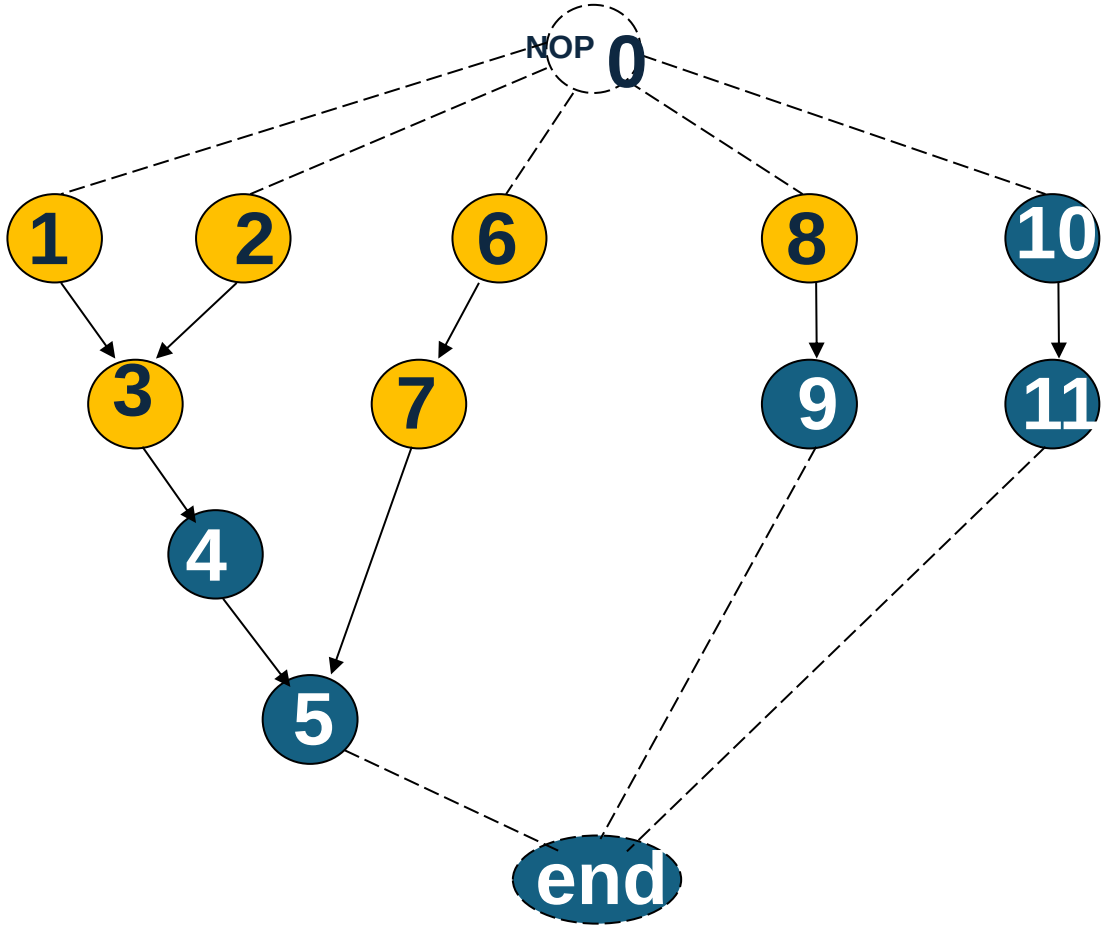
in-class assignment

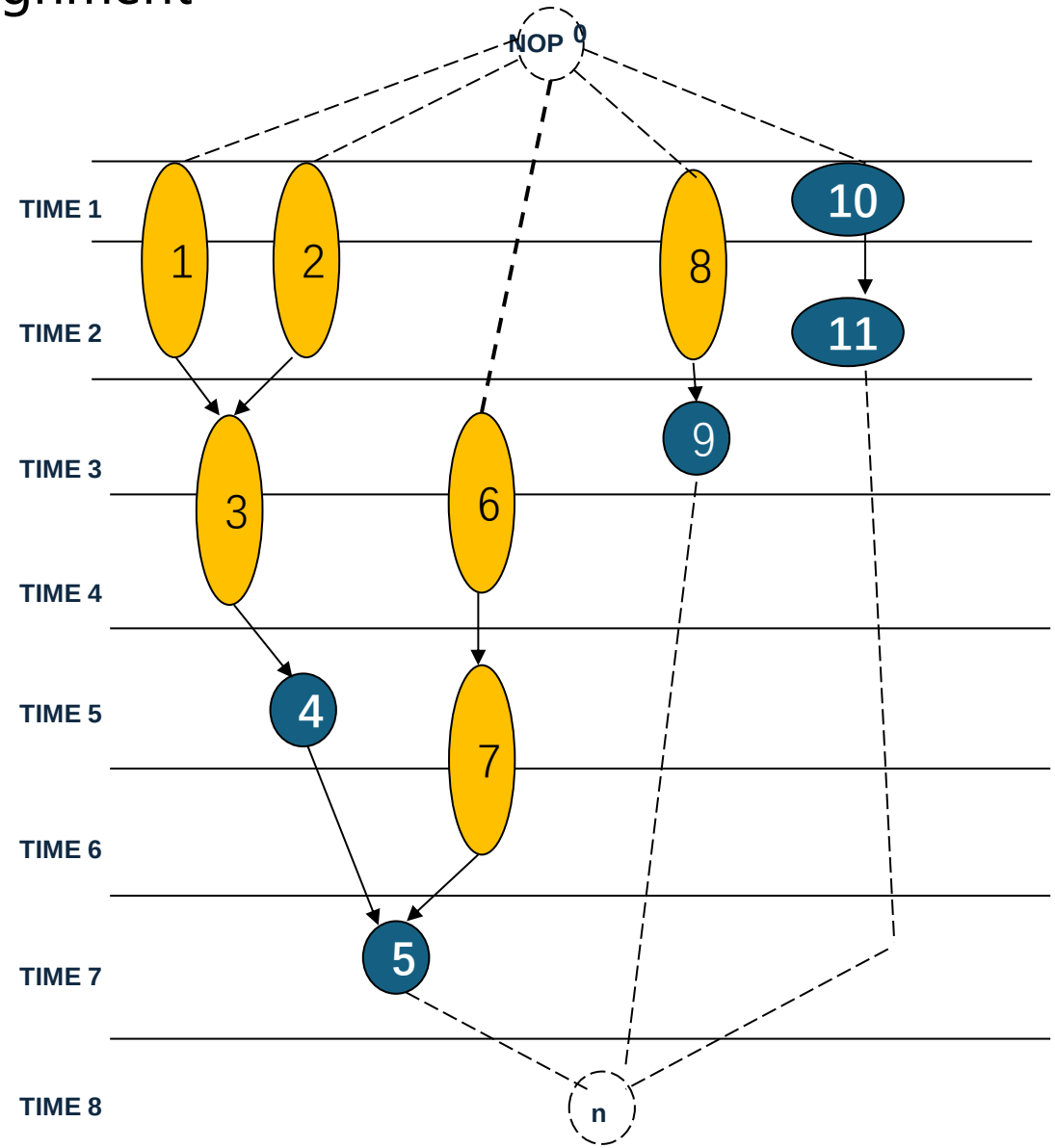




随堂作业

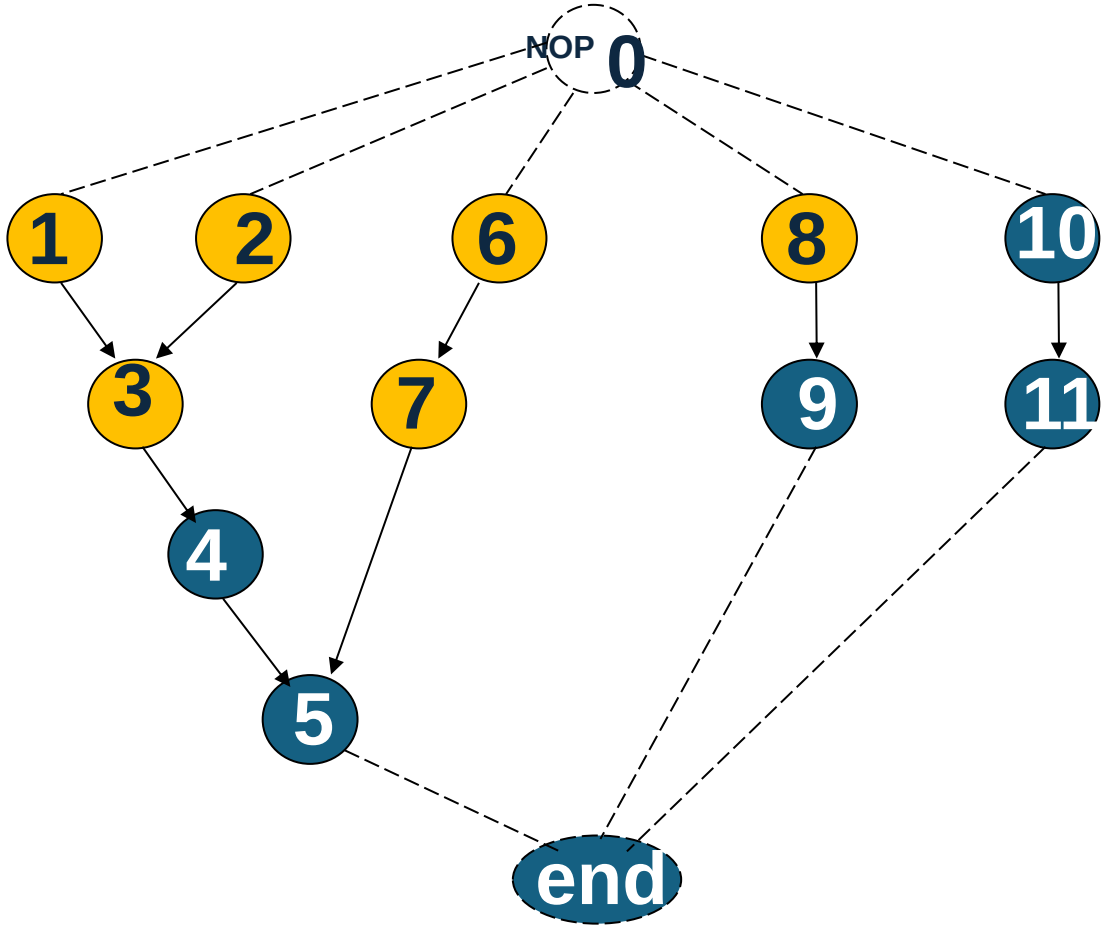
in-class assignment

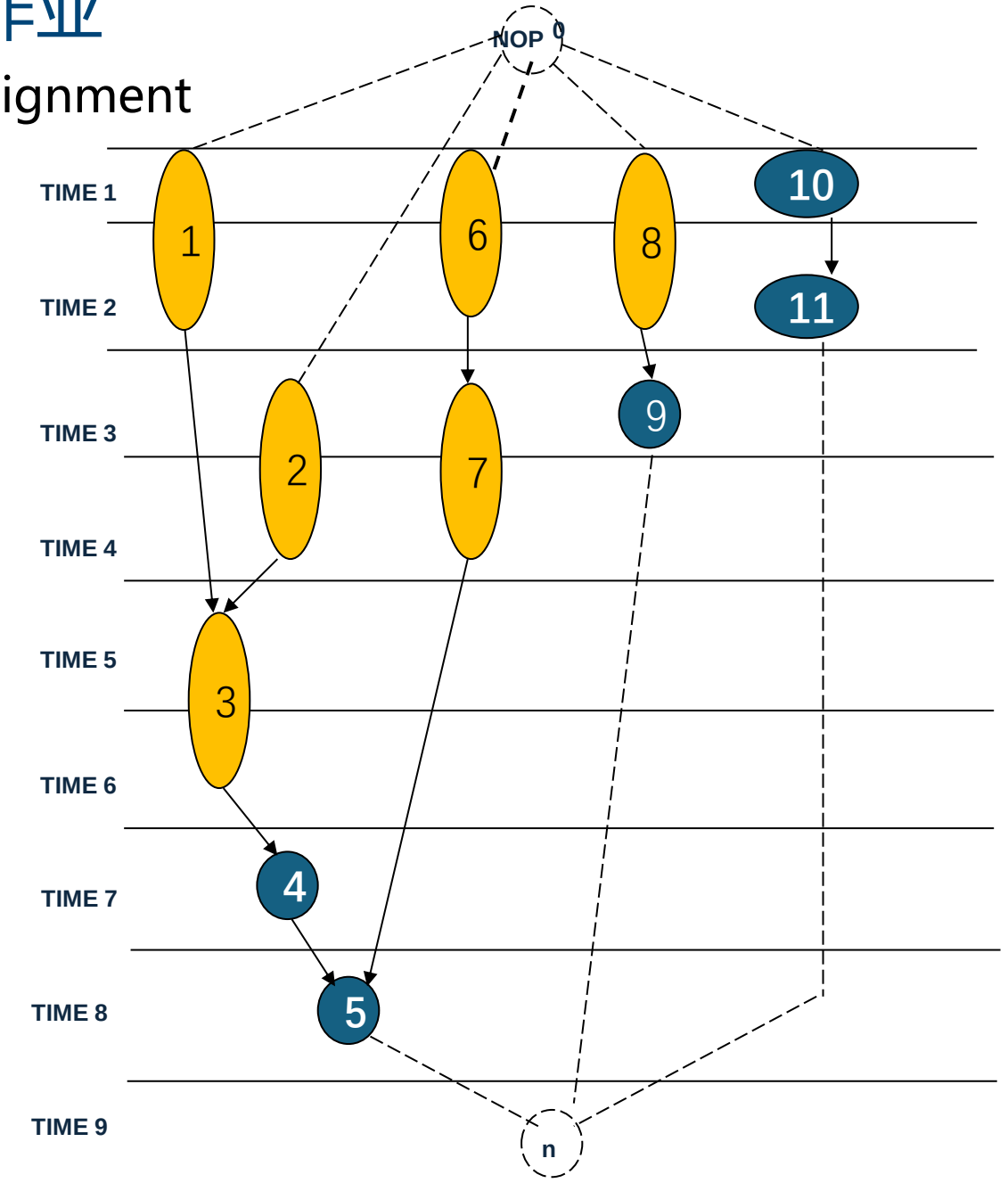




随堂作业

in-class assignment





有约束的调度

Constrained Scheduling

- 约束调度
  - 一般情况下是NP完全问题
  - 在面积或资源的约束下最小化延迟（ML-RCS）
  - 使受到延迟约束的资源最小化（MR-LCS）
- 确切解决方法
  - ILP：整数线性规划（Integer linear program）
  - Hu算法：适用于只有一种资源类型的问题
- 启发式算法
  - 列表调度（List scheduling）
  - 力导向调度（Force-directed scheduling）

列表调度算法：MR-LCS

List scheduling algorithm for minimum resource usage

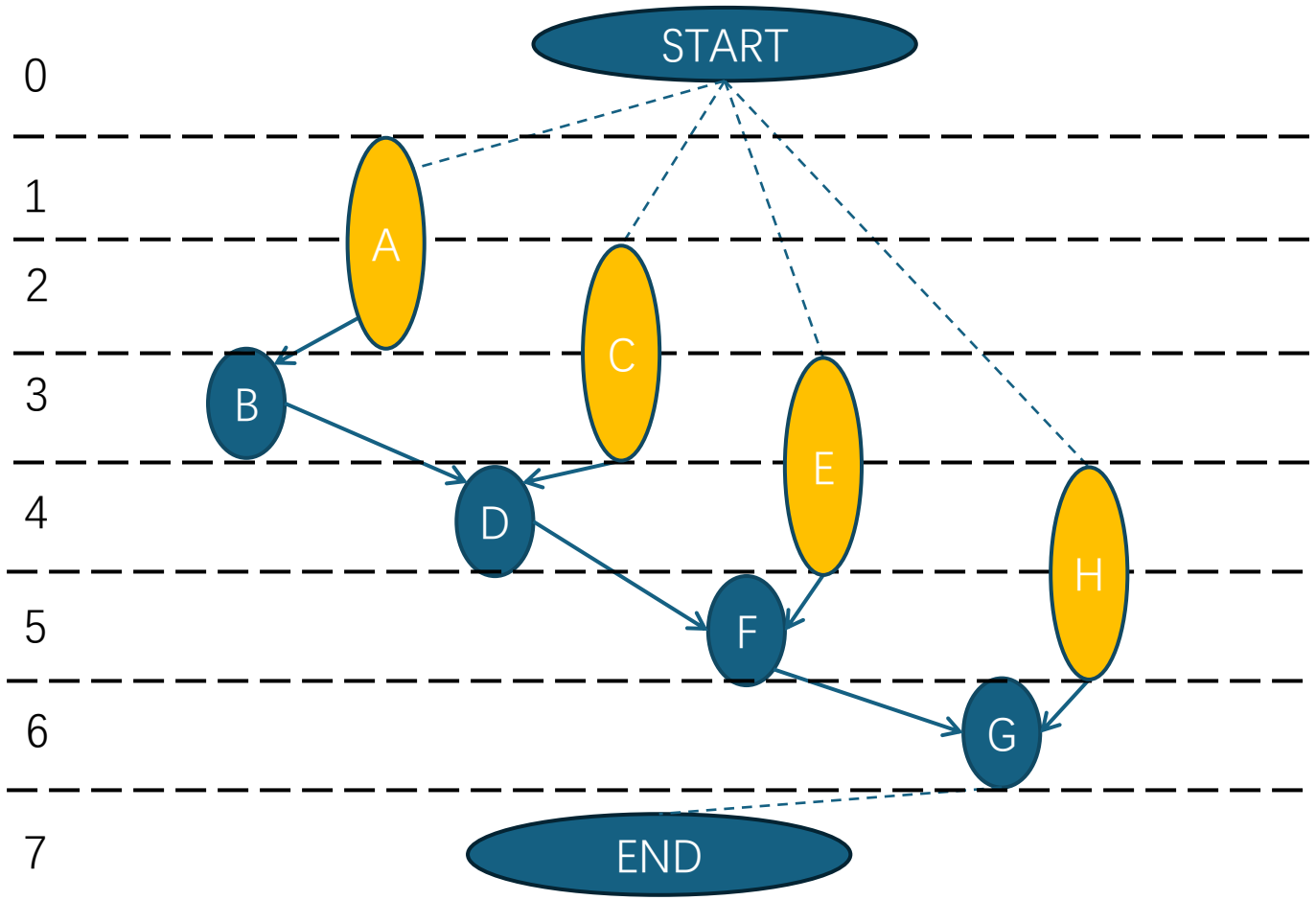
```
LIST_R( G(V, E), λ) {
        a = 1;
        Compute the latest possible start times t^L by ALAP ( G(V, E), λ);
        if (t_0 < 0)
            return (Ø);
         t_0 = 0;    l = 1;
        repeat {
                for each resource type k = 1, 2, …, n_res {
                    Determine ready operations U_l,k;
                    Compute the slacks  { s_i = t_i – l      for all v_i ϵ U_lk};
                    Schedule the candidate operations with zero slack and update a;
                    Schedule the candidate operations not needing additional resources;
                    }
                l = l + 1;
        }
        until (v_n is scheduled) ;
        return (t, a);
}
```

列表调度算法：MR-LCS

List scheduling algorithm for minimum resource usage

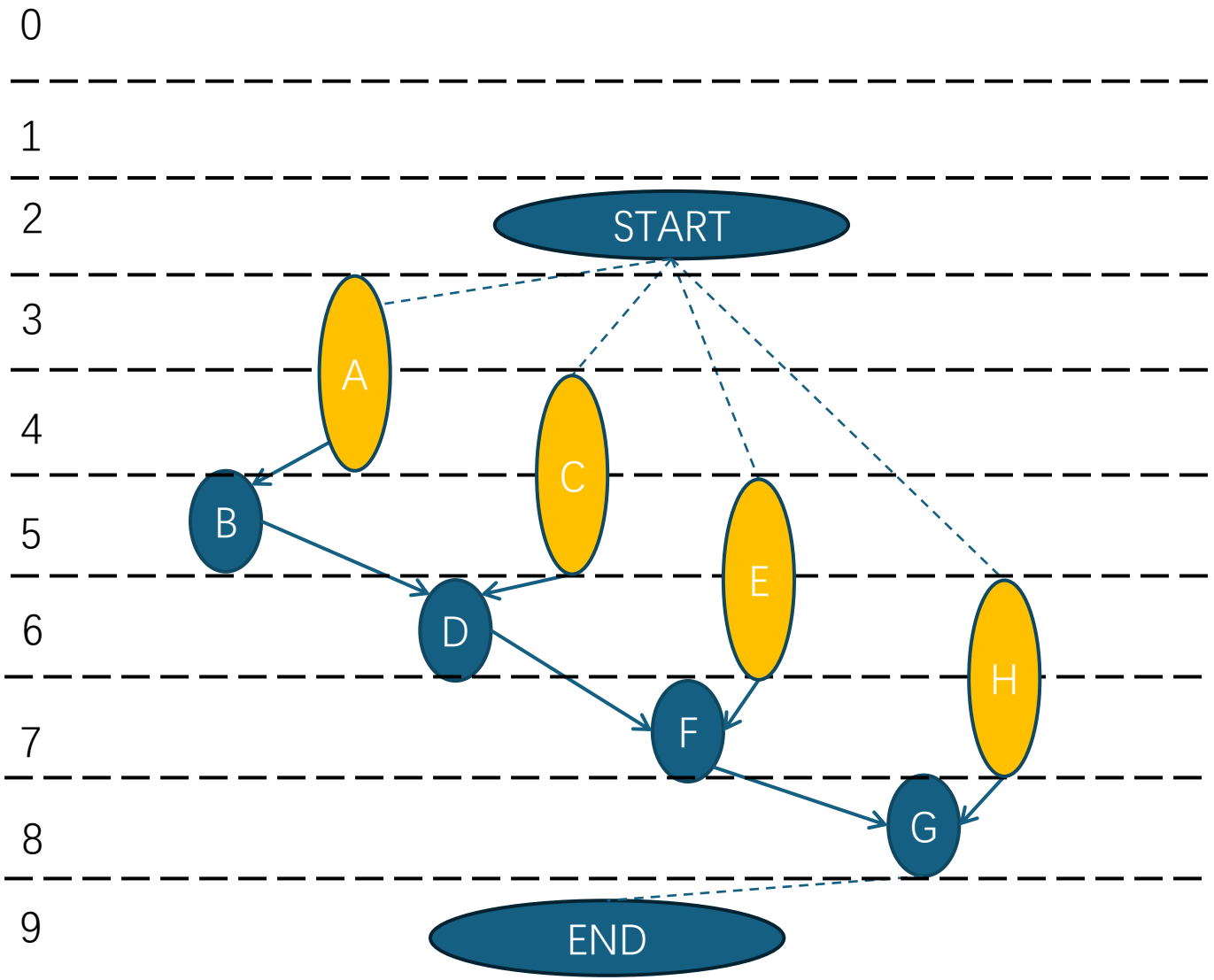
```
1 LIST_R( G(V, E), λ) {
2         a = 1;（a代表资源数）
3         通过 ALAP ( G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
4         if (t_0 < 0)
5            return (Ø);
6          t_0 = 0;    l = 1;
7         重复执行以下步骤{
8                   对于每种资源类型 k = 1, 2, …, n_res {
9                      确定就绪的操作集 U_l,k;
10                     对U_l,k中的每个顶点，计算其slacks s_i = t^L_i – l ;
11                     调度所有松弛度为零的候选操作，并更新 a;
12                     调度不需要额外资源的候选操作;
13                     }
14                  l = l + 1;
15        }
16        直到 vn 被调度完成;
17        return (t, a);
18 }
```

λ为6时ALAP调度的结果：



START 0
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6
END 7

λ为8时ALAP调度的结果：



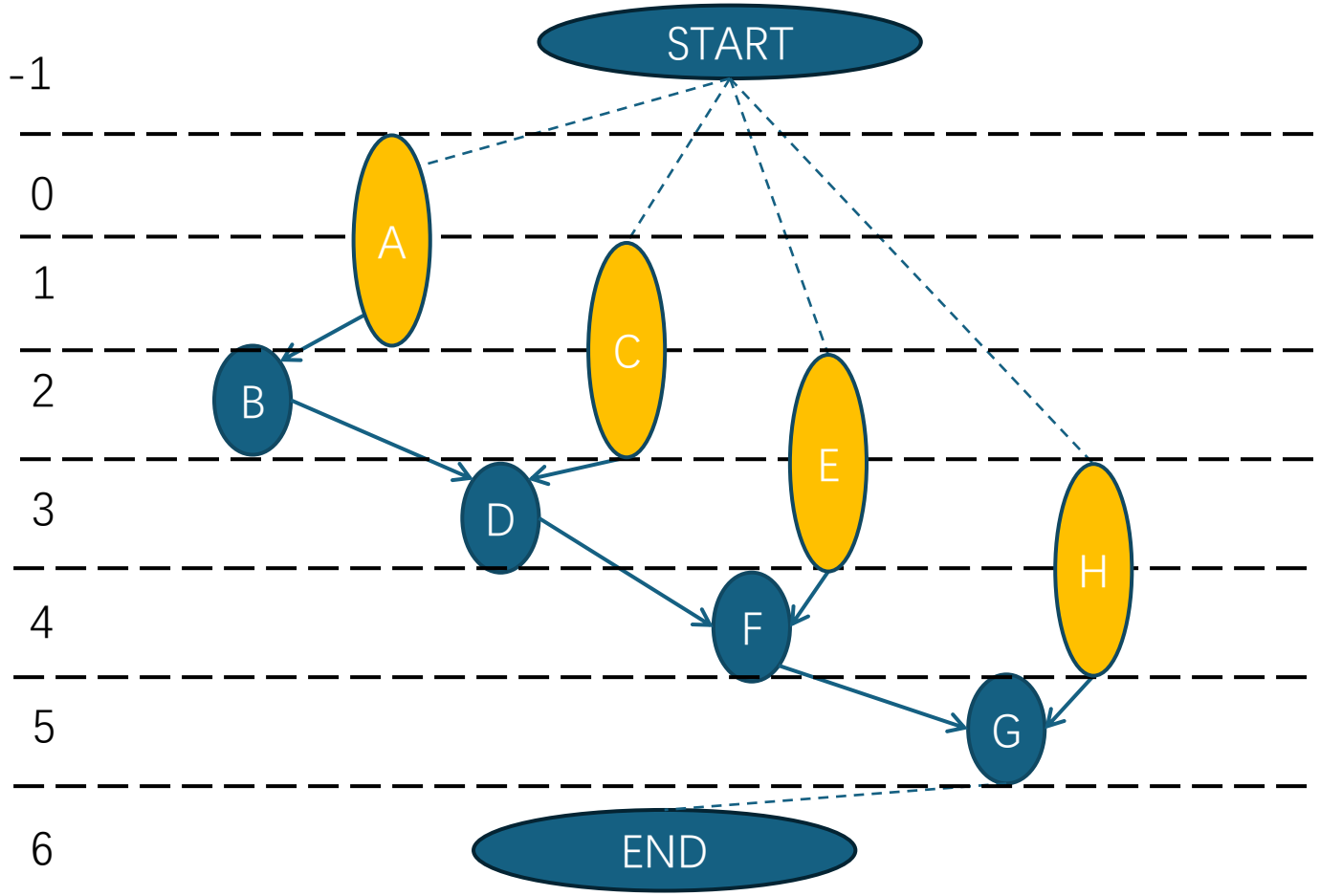
START 2
A 3
B 5
C 4
D 6
E 5
F 7
H 6
G 8
END 9

列表调度算法：MR-LCS

List scheduling algorithm for minimum resource usage

```
1 LIST_R( G(V, E), λ) {
2         a = 1;（a代表资源数）
3         通过 ALAP ( G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
4         if (t_0 < 0)
5            return (Ø);
6          t_0 = 0;    l = 1;
7         重复执行以下步骤{
8                   对于每种资源类型 k = 1, 2, …, n_res {
9                      确定就绪的操作集 U_l,k;
10                     对U_l,k中的每个顶点，计算其slacks s_i = t^L_i – l ;
11                     调度所有松弛度为零的候选操作，并更新 a;
12                     调度不需要额外资源的候选操作;
13                     }
14                  l = l + 1;
15        }
16        直到 vn 被调度完成;
17        return (t, a);
18 }
```

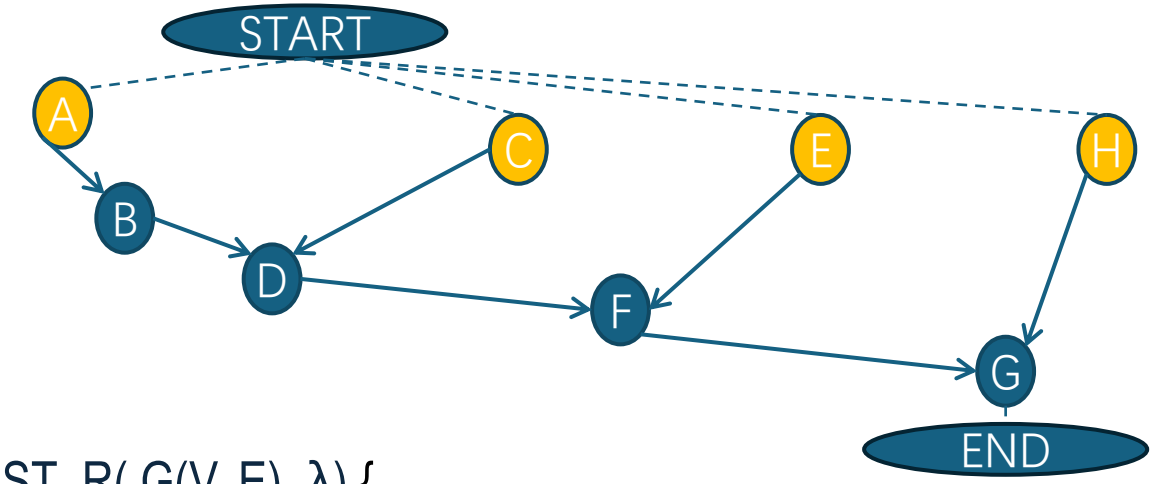
ALAP调度的结果：



列表调度算法：MR-LCS

List scheduling algorithm for minimum resource usage

```
1 LIST_R( G(V, E), λ) {
2         a = 1;（a代表资源数）
3         通过 ALAP ( G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
4         if (t_0 < 0)
5            return (Ø);
6          t_0 = 0;    l = 1;
7         重复执行以下步骤{
8                   对于每种资源类型 k = 1, 2, …, n_res {
9                      确定就绪的操作集 U_l,k;
10                     对U_l,k中的每个顶点，计算其slacks s_i = t^L_i – l ;
11                     调度所有松弛度为零的候选操作，并更新 a;
12                     调度不需要额外资源的候选操作;
13                     }
14                  l = l + 1;
15        }
16        直到 vn 被调度完成;
17        return (t, a);
18 }
```



假设λ=6，乘法器延迟为2，加法器延迟为1

LIST_R(G(V, E), λ) {

a = 1; （a代表资源数）

通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;

if (t^L_0 < 0)

return (Ø);

t_0 = 0; l = 1;

重复执行以下步骤{

对于每种资源类型 k = 1, 2, …, n_{res} {

确定就绪的操作集 $U_{l,k}$;

对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;

调度所有松弛度为零的候选操作，并更新 a;

调度不需要额外资源的候选操作;

}

$l = l + 1$;}

直到 vn 被调度完成;

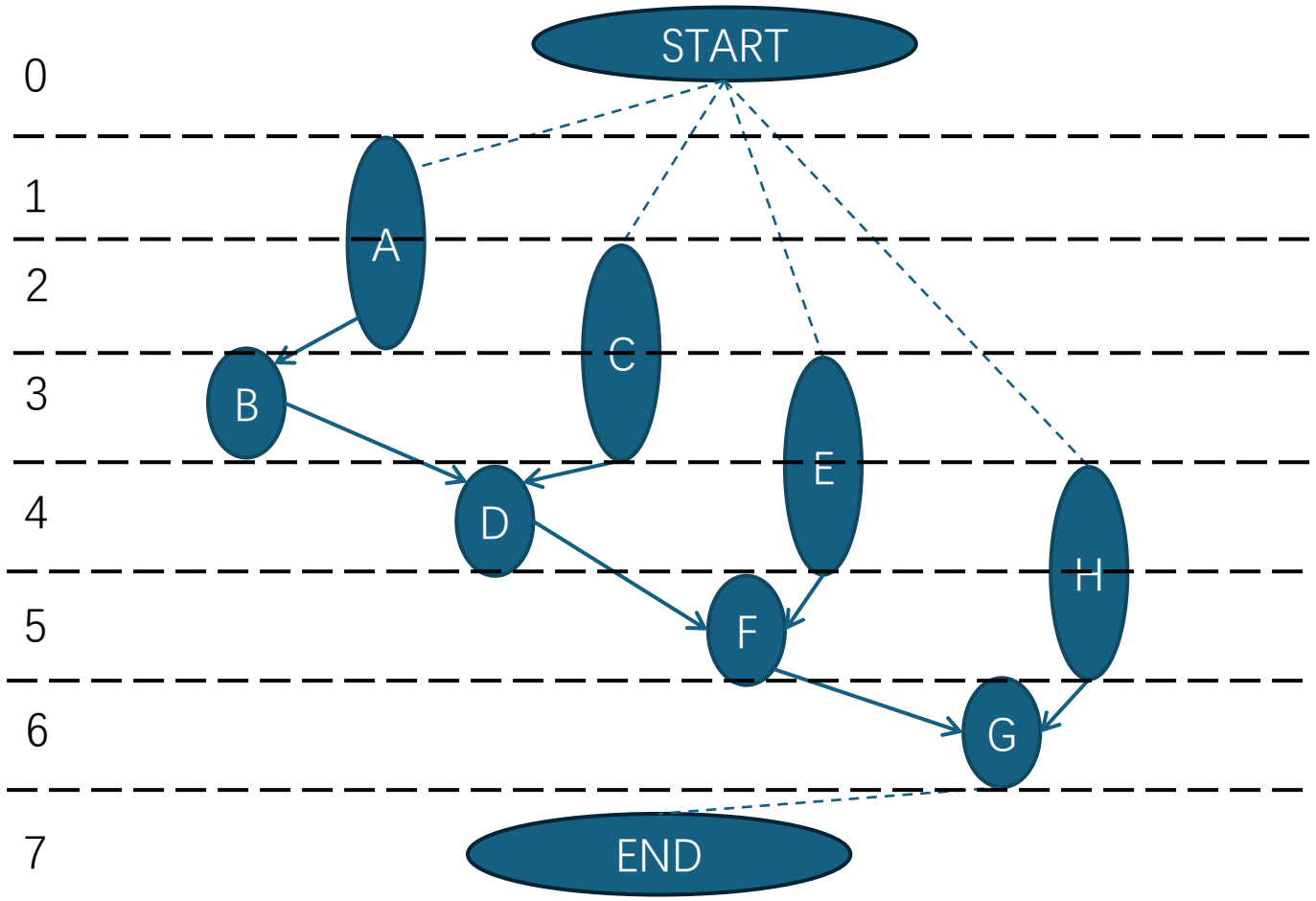
return (t, a);}

k=1为加法操作

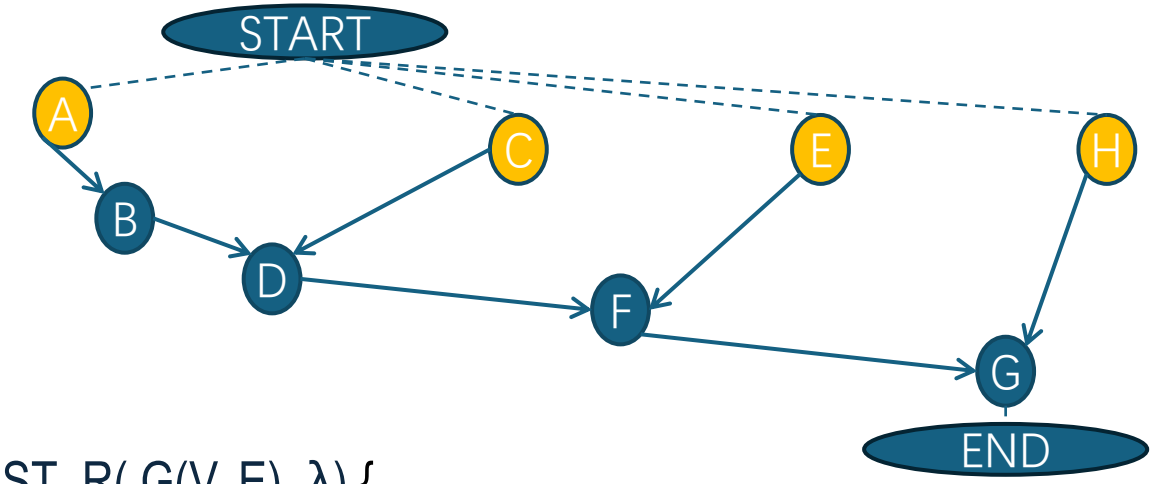
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l		
k		
U		
S		

ALAP调度的结果：



START 0
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6
END 7

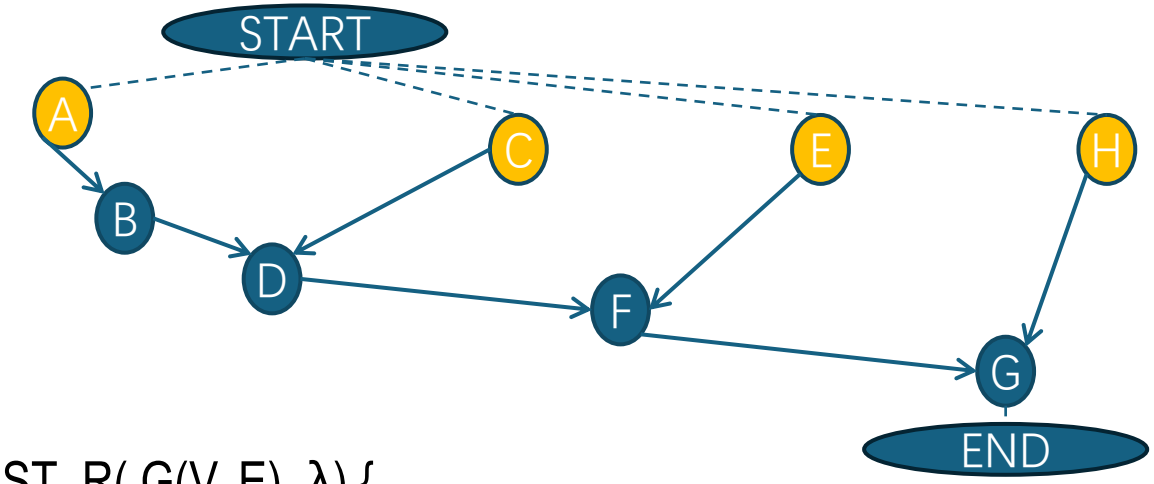


A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l		
k		
U		
S		



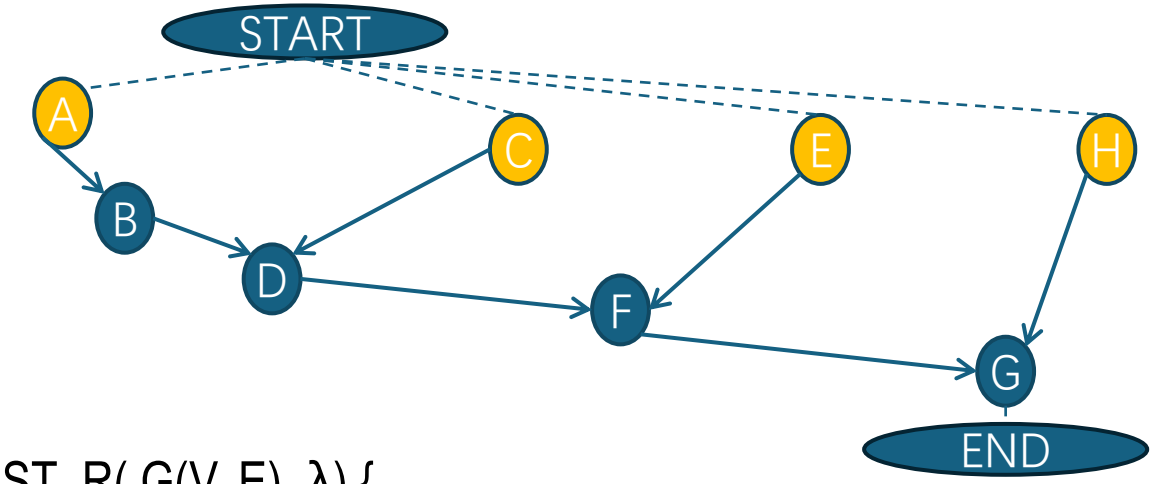
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点， 计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作， 并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	1	
k		
U		
S		



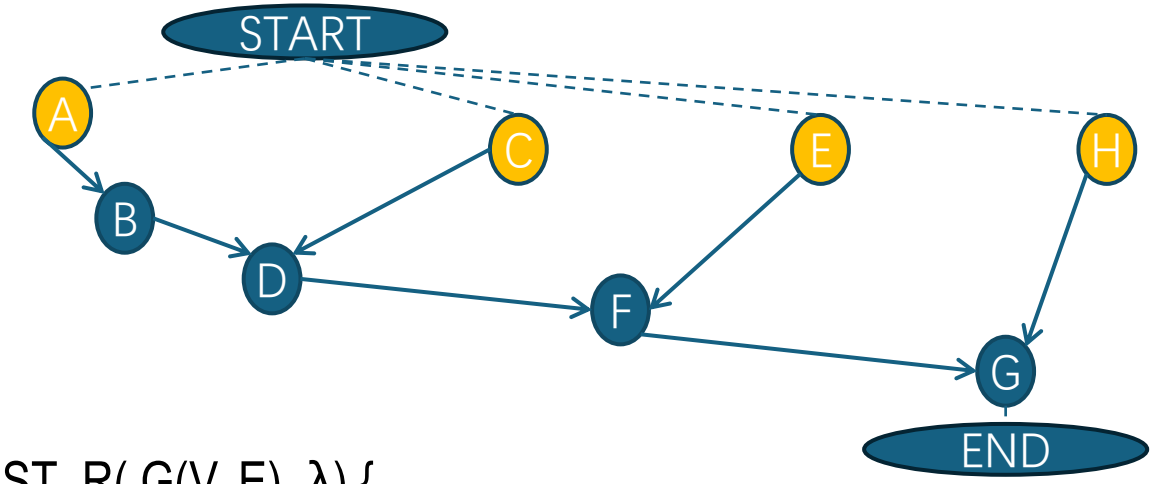
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点， 计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作， 并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	1	
k		
U		
S		



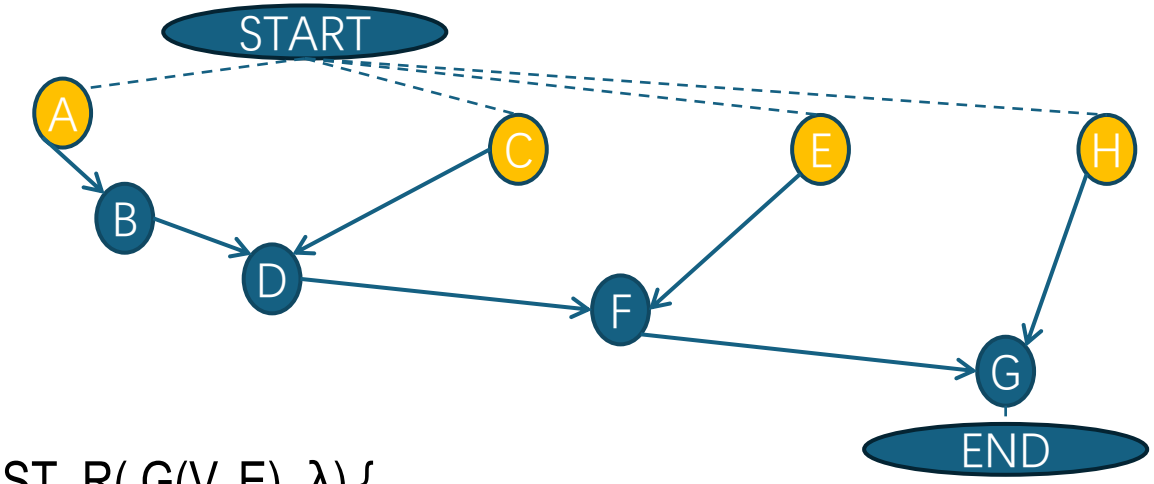
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	1	
k	1	
U		
S		



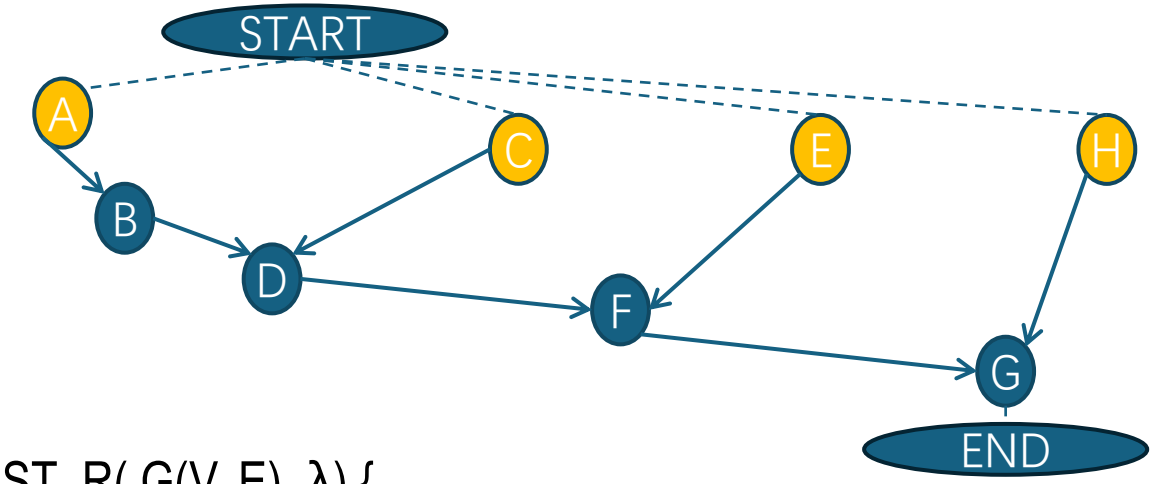
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	1	
k	1	
U	{}	
S	{}	



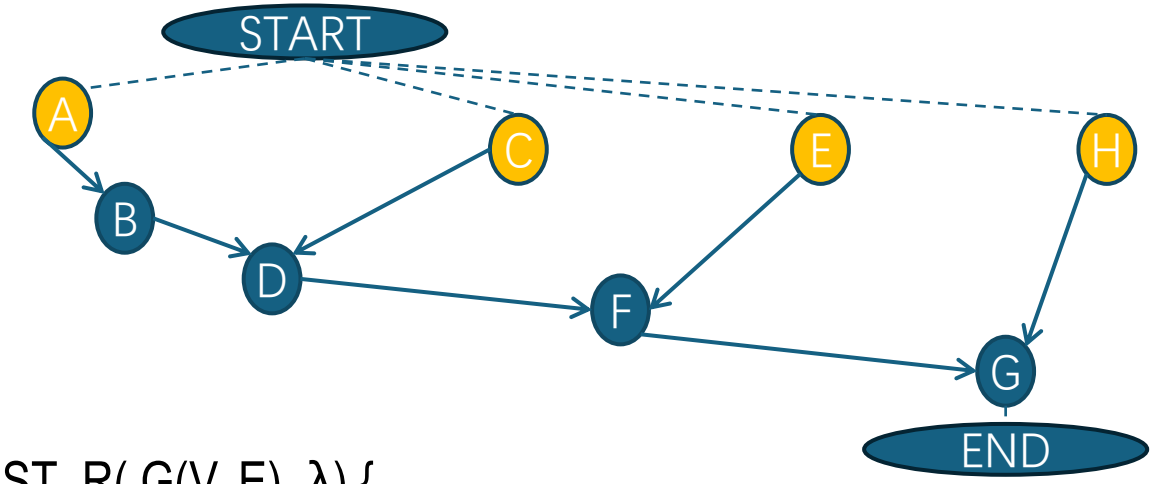
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	1	
k	2	
U	{}	
S	{}	



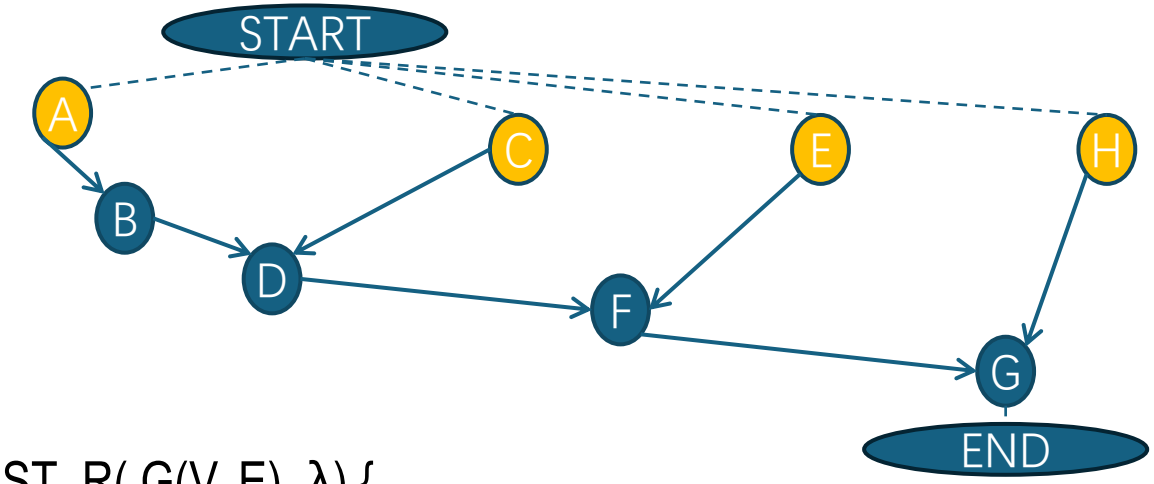
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	1	
k	2	
U	{A,C,E,H}	
S	{}	



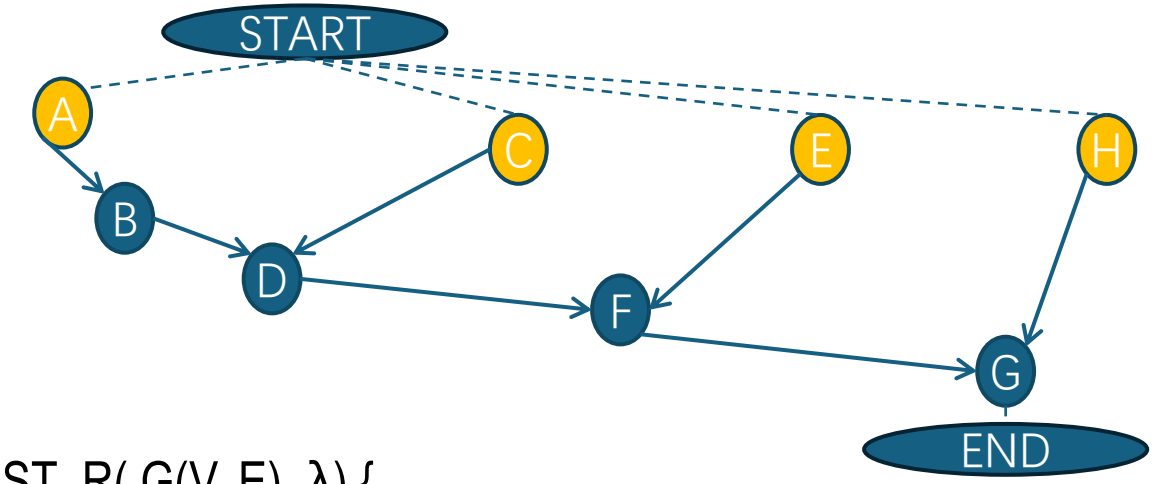
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	1	
k	2	
U	{A,C,E,H}	
S	{s_A=0, s_C=1, s_E=2, s_H=3}	



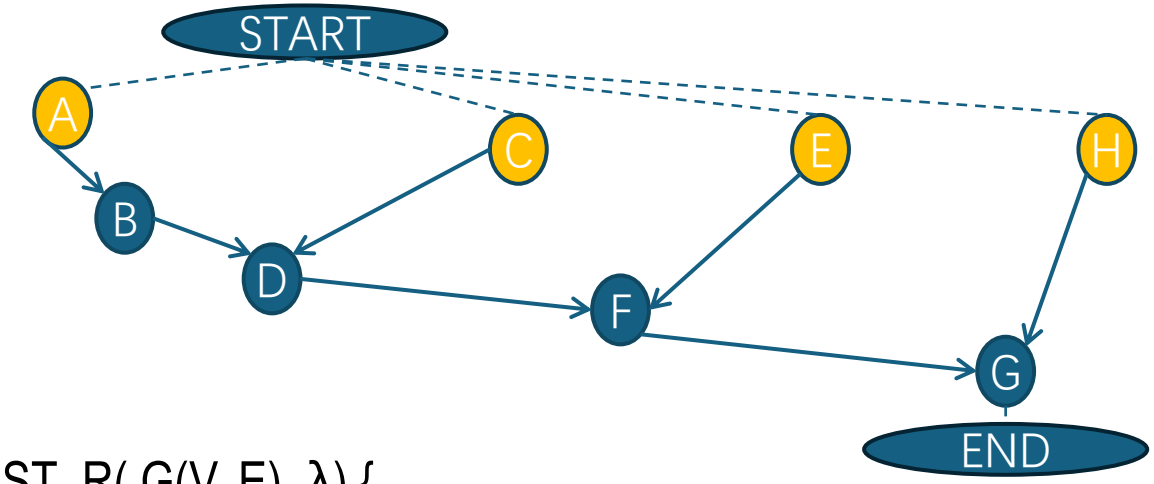
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	1	
k	2	
U	{A,C,E,H}	
S	{s_A=0, s_C=1, s_E=2, s_H=3}	



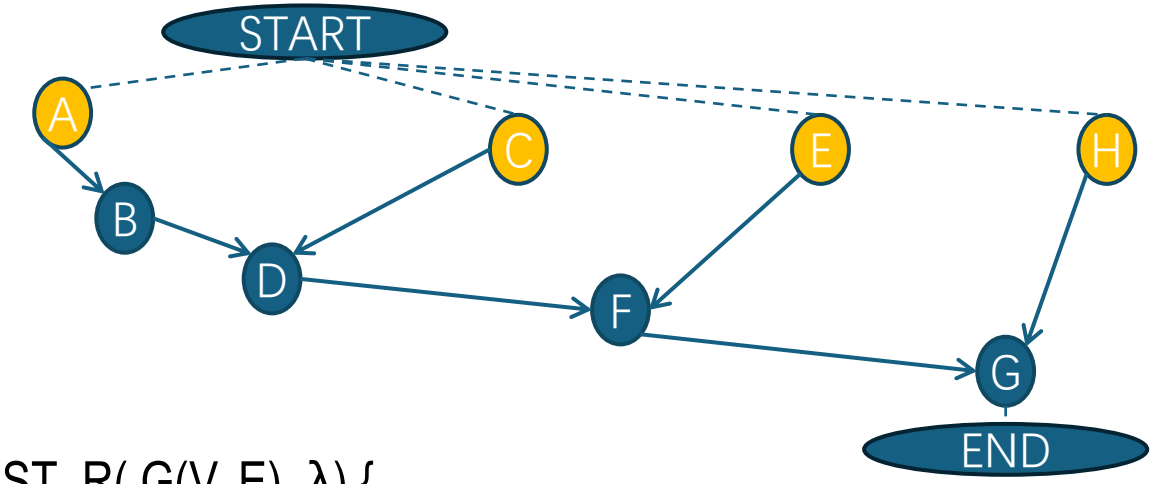
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	1	
k	2	
U	{A,C,E,H}	
S	{s_A=0, s_C=1, s_E=2, s_H=3}	



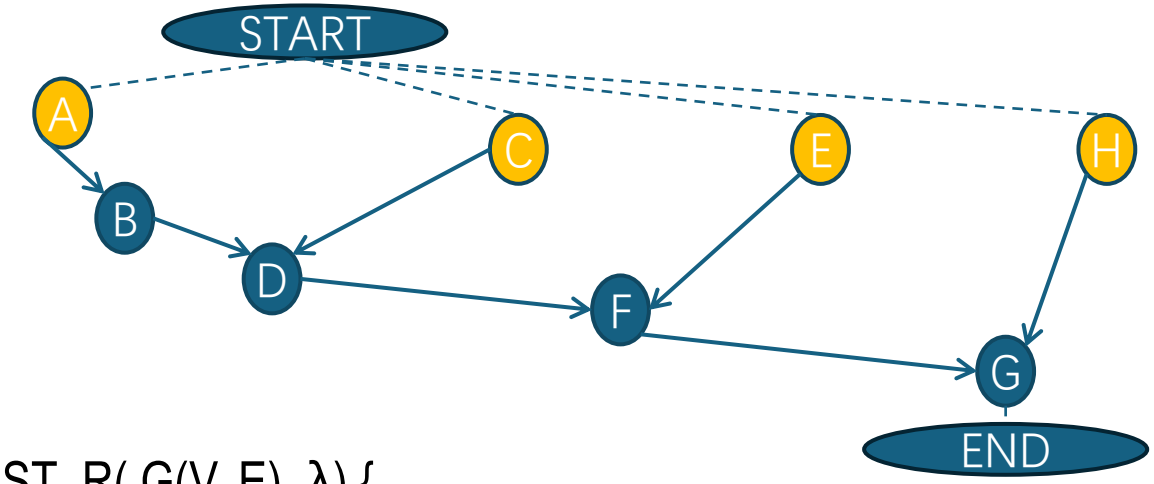
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	2	
k	2	
U	{A,C,E,H}	
S	{s_A=0, s_C=1, s_E=2, s_H=3}	



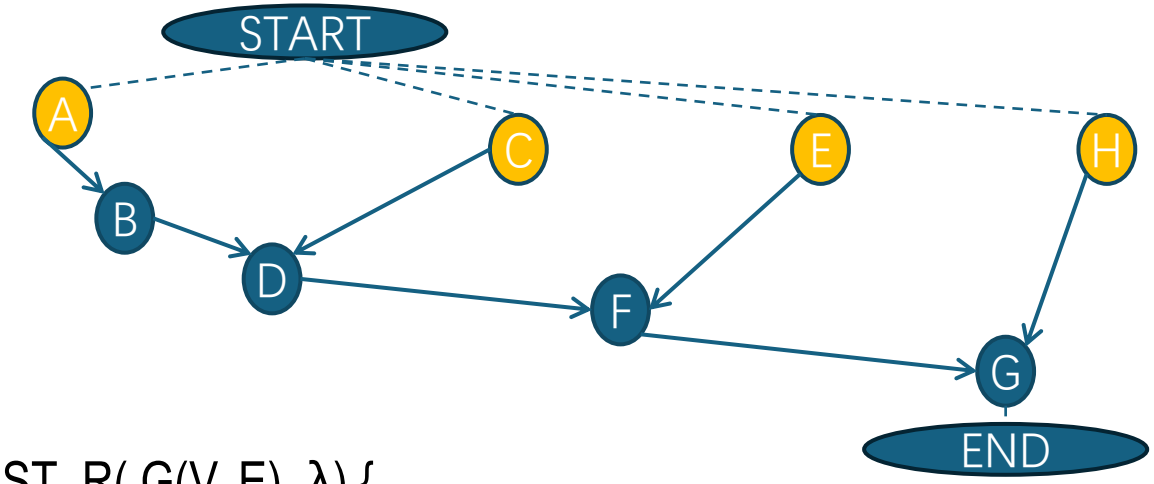
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	2	
k	1	
U	{A,C,E,H}	
S	{s_A=0, s_C=1, s_E=2, s_H=3}	



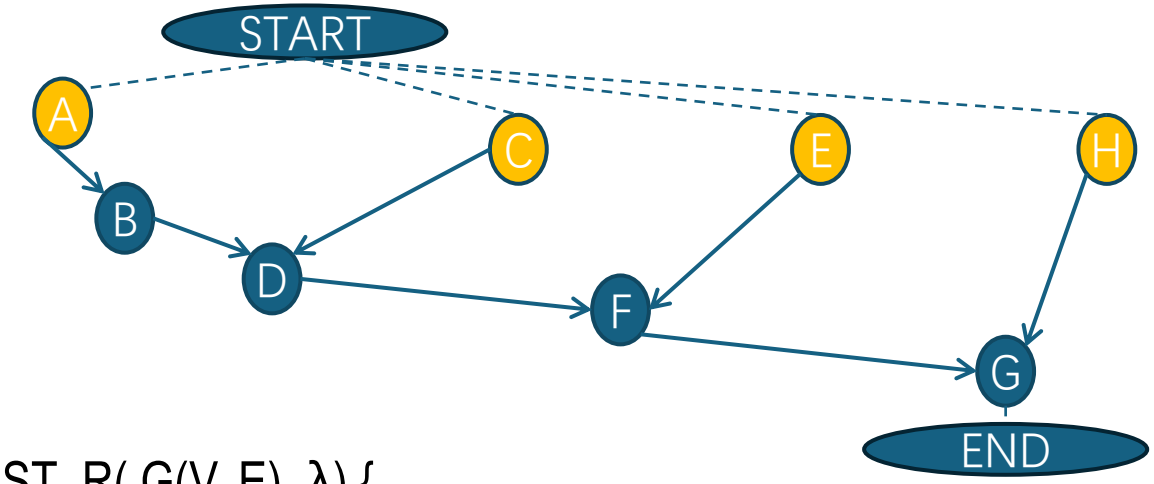
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	2	
k	1	
U	{}	
S	{}	



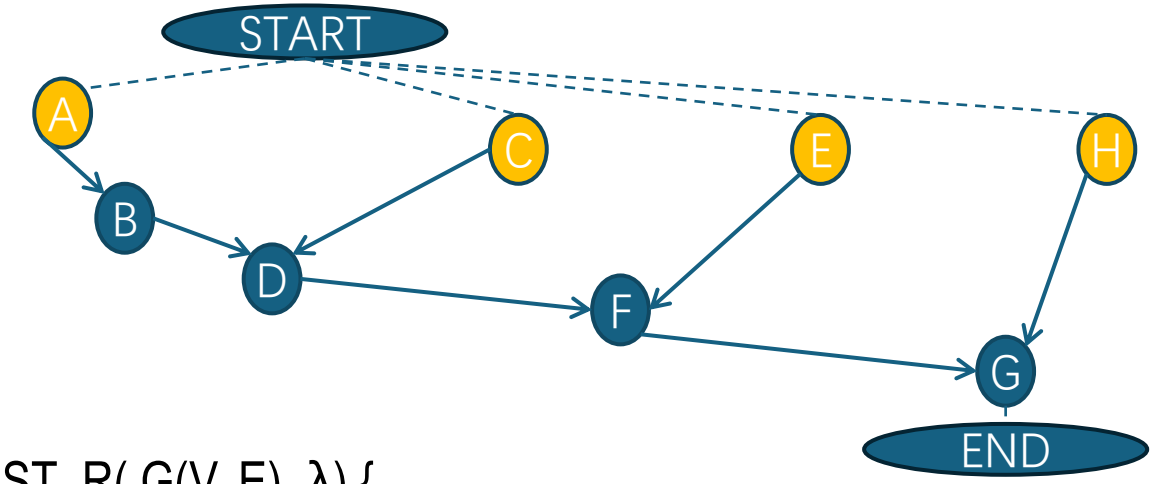
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	2	
k	2	
U	{}	
S	{}	



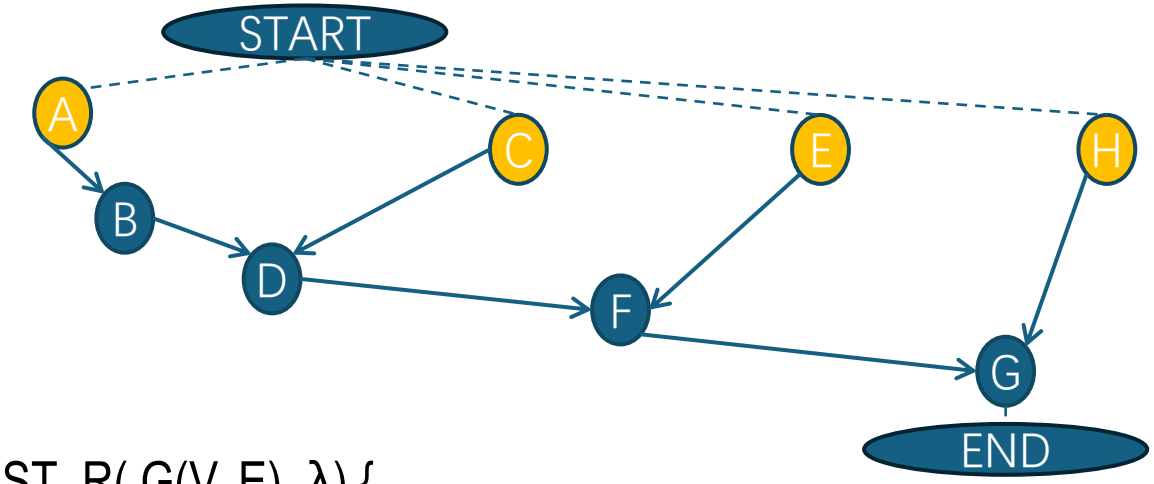
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	2	
k	2	
U	{C,E,H}	
S	{}	



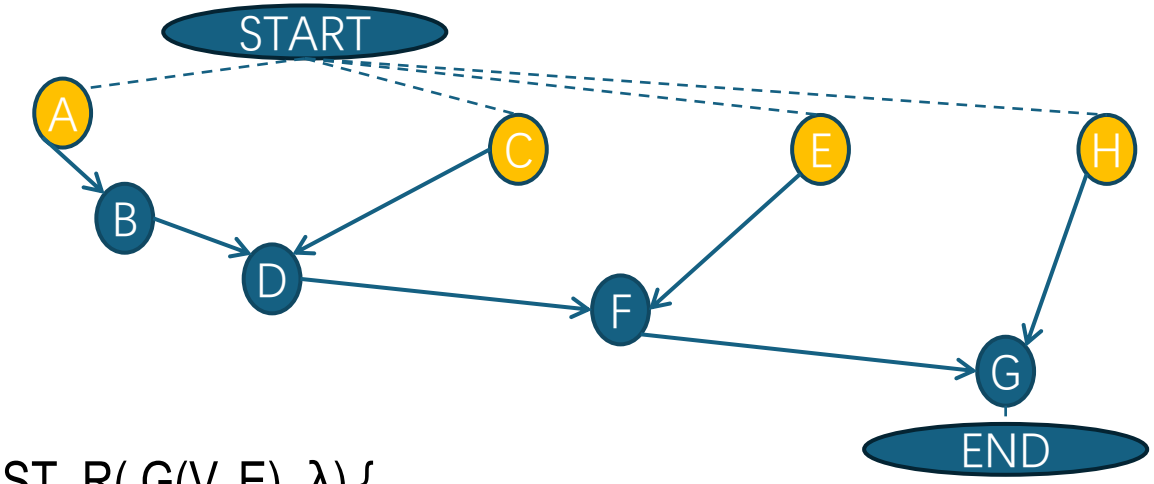
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	1
l	2	
k	2	
U	{C,E,H}	
S	{s_C=0, s_E=1, s_H=2}	



A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0

A 1

C 2

LIST_R(G(V, E), λ) {

a = 1; （a代表资源数）

通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;

if (t^L_0 < 0)

return (Ø);

t_0 = 0; l = 1;

重复执行以下步骤{

对于每种资源类型 k = 1, 2, …, n_{res} {

确定就绪的操作集 $U_{l,k}$;

对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;

调度所有松弛度为零的候选操作，并更新 a;

调度不需要额外资源的候选操作;

}

$l = l + 1$;}

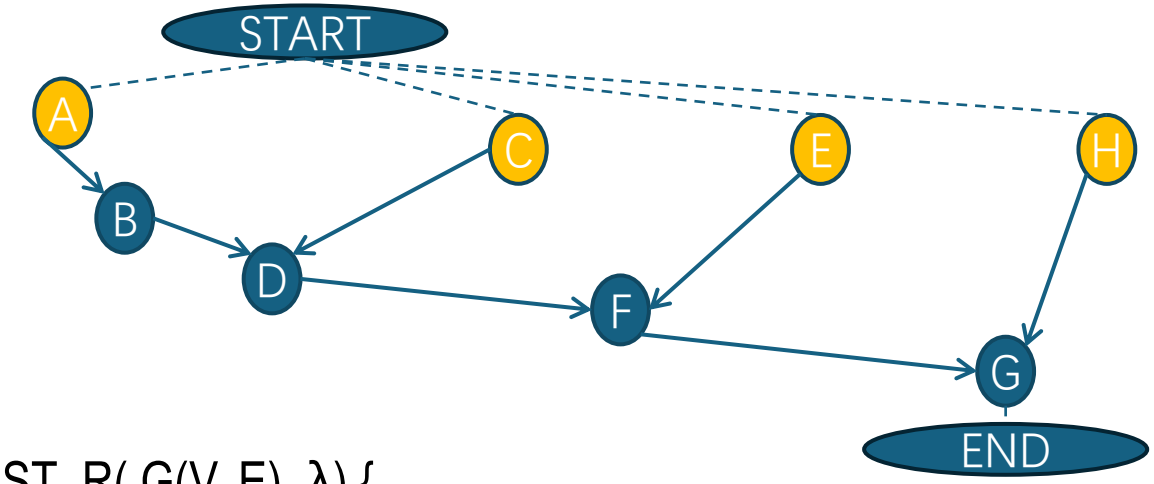
直到 vn 被调度完成;

return (t, a);}

k=1为加法操作

k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	2	
k	2	
U	{C,E,H}	
S	{s_C=0, s_E=1, s_H=2}	



A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0

A 1

C 2

LIST_R(G(V, E), λ) {

a = 1; （a代表资源数）

通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;

if (t^L_0 < 0)

return (Ø);

t_0 = 0; l = 1;

重复执行以下步骤{

对于每种资源类型 k = 1, 2, …, n_{res} {

确定就绪的操作集 $U_{l,k}$;

对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;

调度所有松弛度为零的候选操作，并更新 a;

调度不需要额外资源的候选操作;

}

$l = l + 1$;}

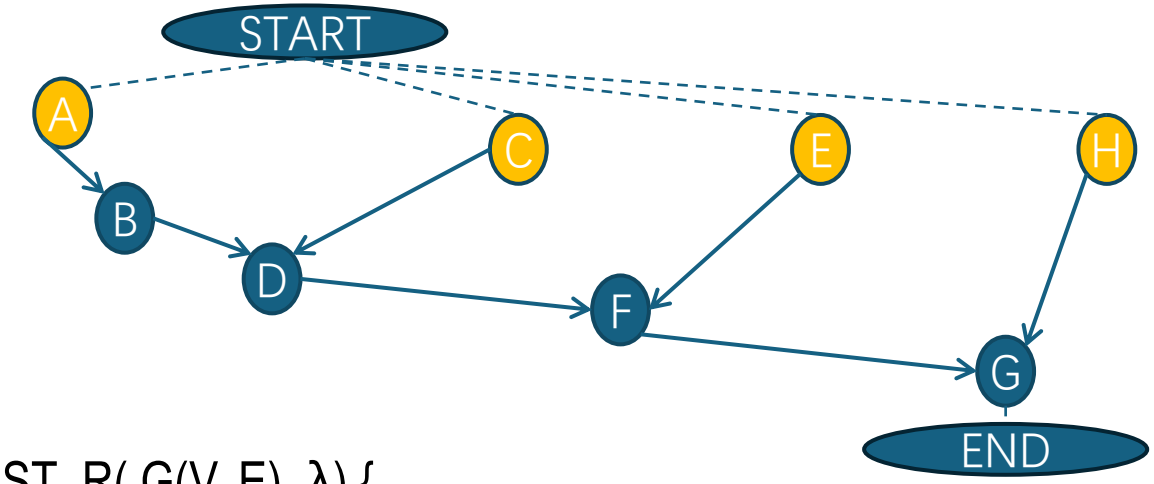
直到 vn 被调度完成;

return (t, a);}

k=1为加法操作

k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	2	
k	2	
U	{C,E,H}	
S	{s_C=0, s_E=1, s_H=2}	



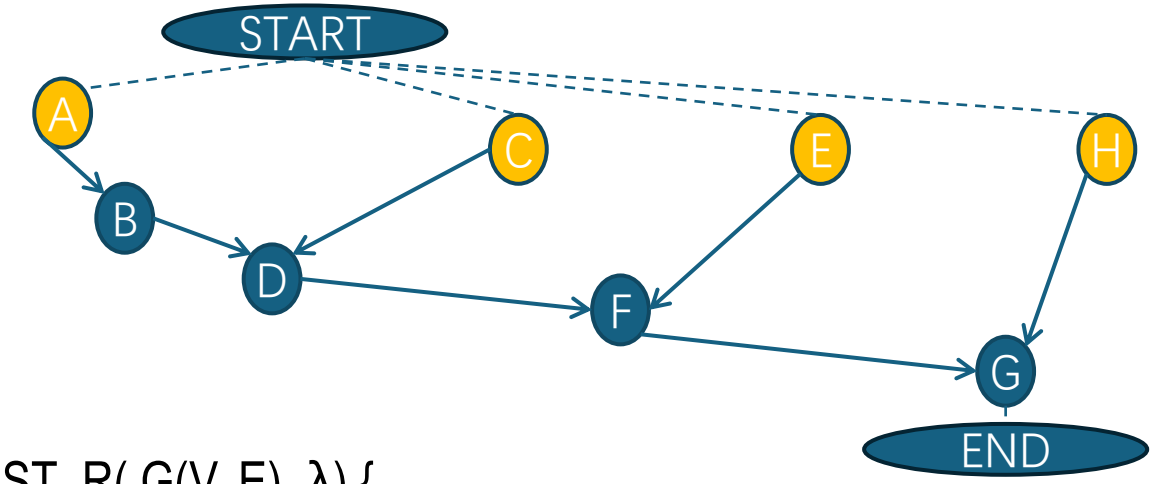
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点， 计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作， 并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	3	
k	2	
U	{C,E,H}	
S	{s_C=0, s_E=1, s_H=2}	



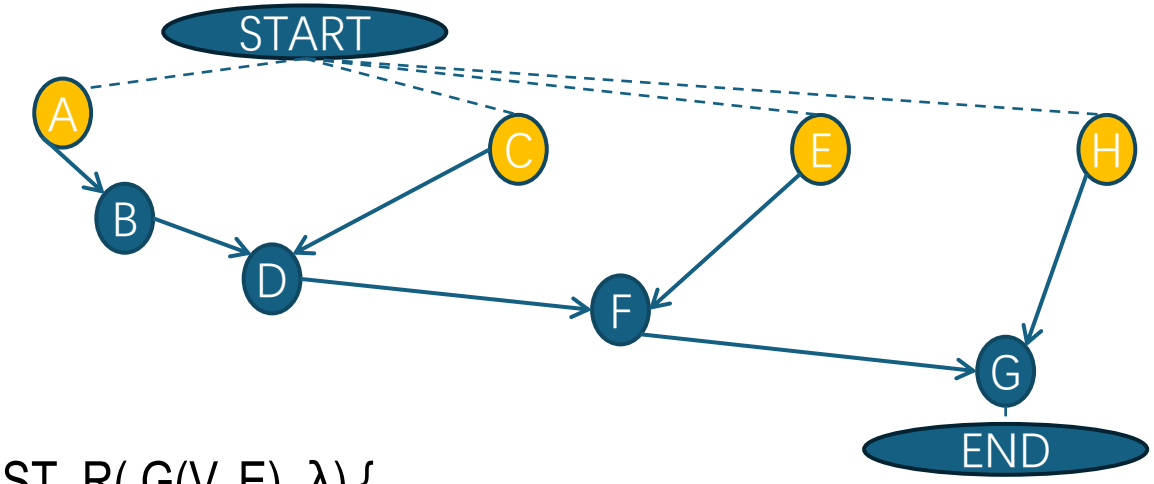
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	3	
k	1	
U	{C,E,H}	
S	{s_C=0, s_E=1, s_H=2}	



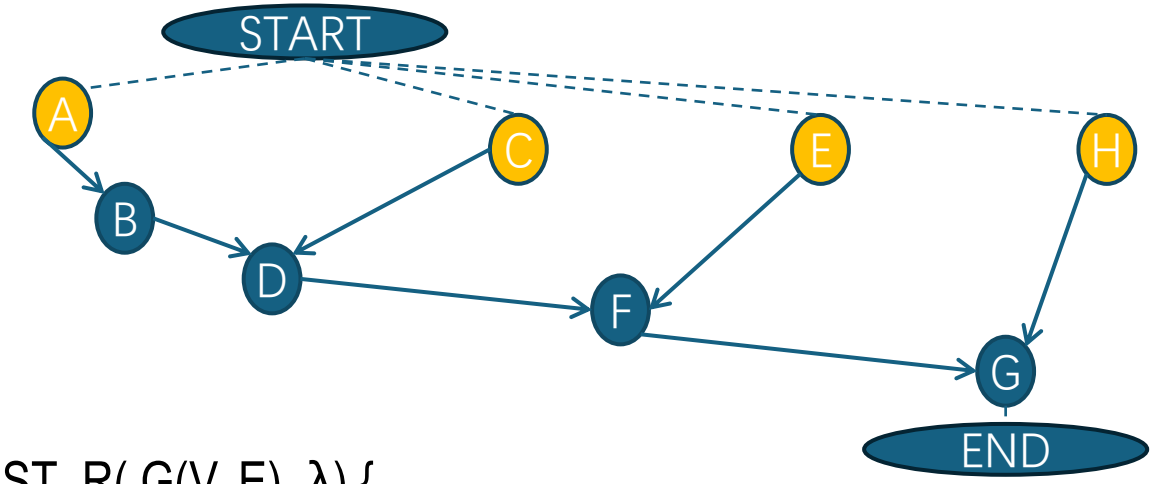
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	3	
k	1	
U	{B}	
S	$\{s_C=0, s_E=1, s_H=2\}$	



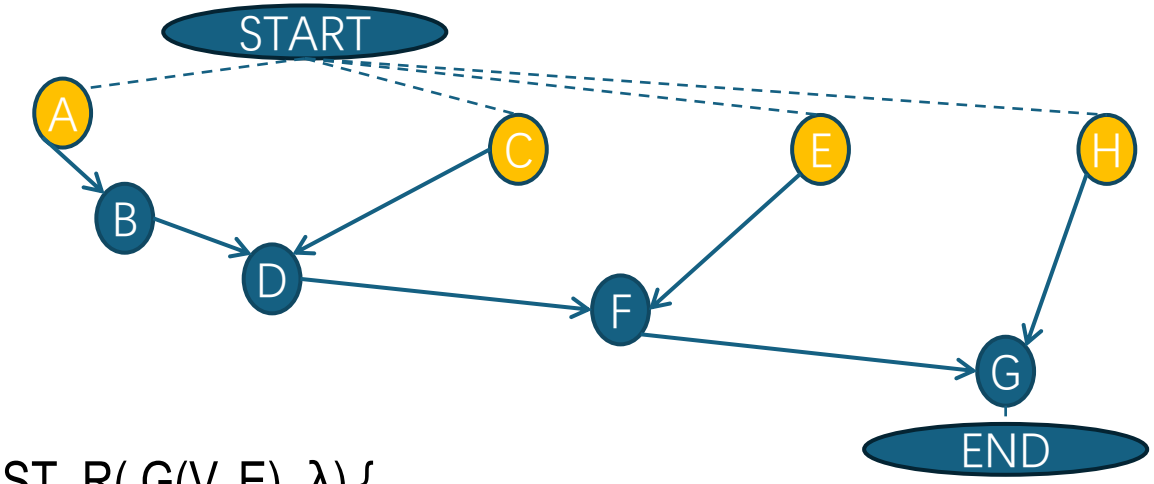
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	3	
k	1	
U	{B}	
S	{s_B=0}	



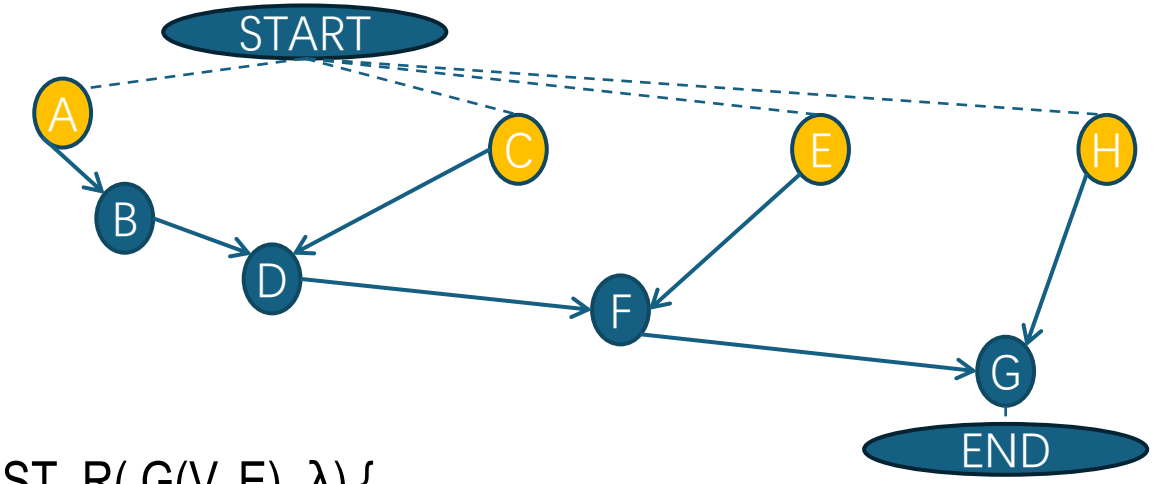
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	3	
k	1	
U	{B}	
S	{s_B=0}	



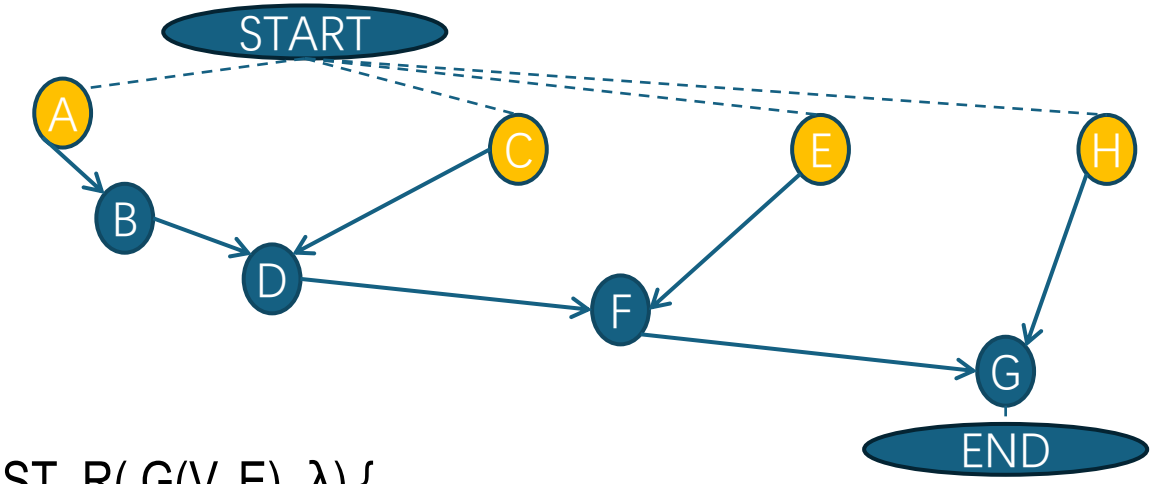
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	3	
k	1	
U	{B}	
S	{s_B=0}	



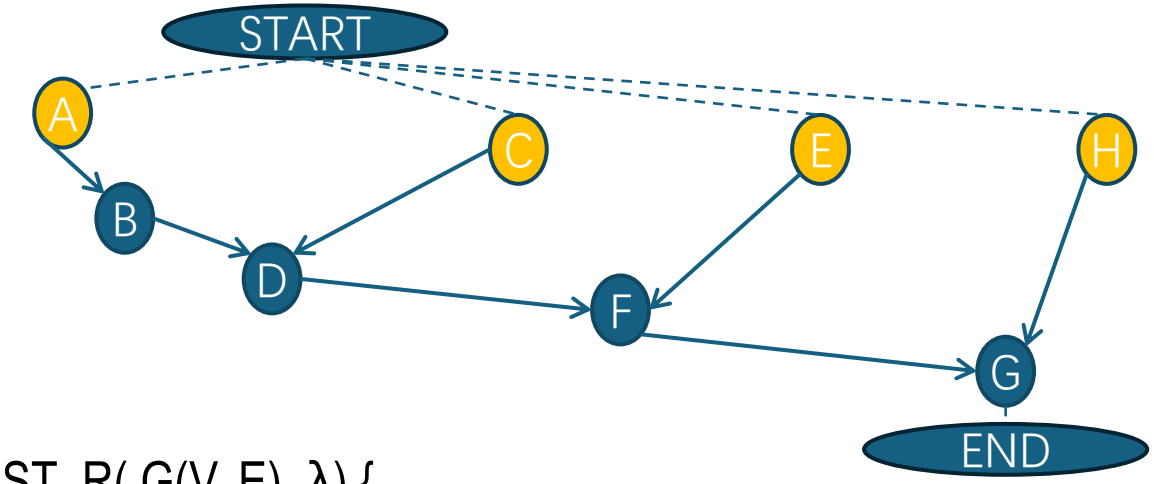
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	3	
k	2	
U	{B}	
S	{s_B=0}	



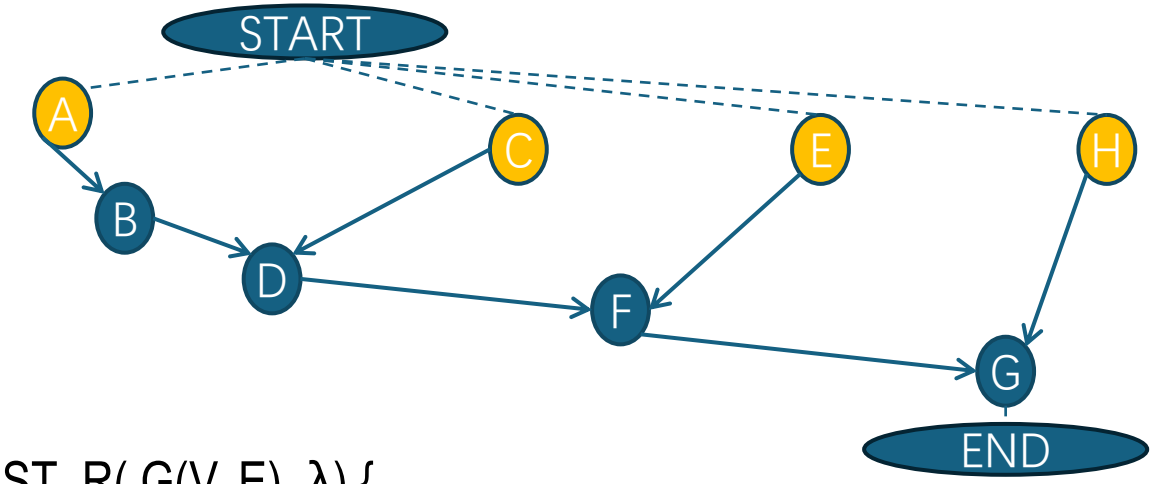
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	3	
k	2	
U	{E,H}	
S	{s_E=0, s_H=1}	



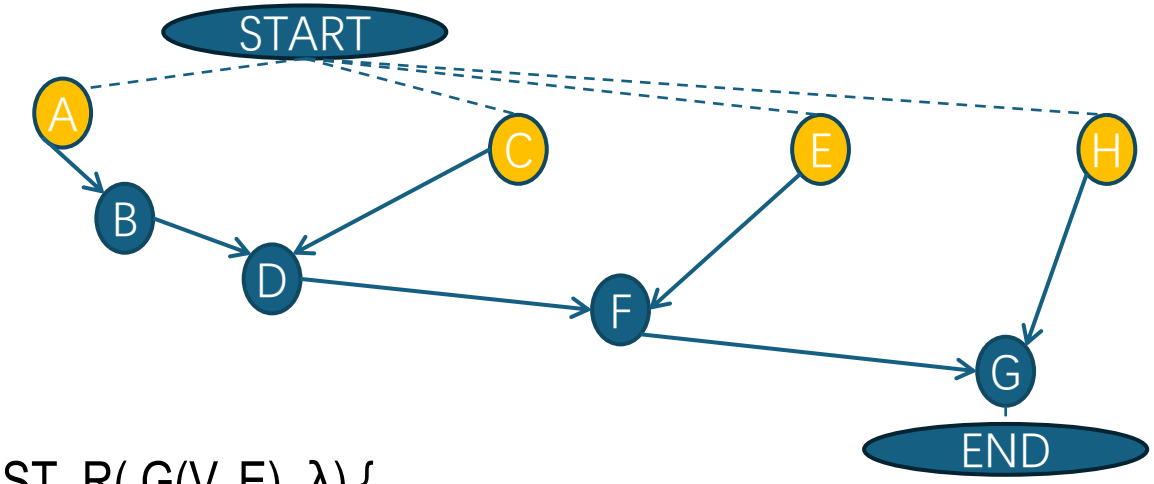
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点， 计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作， 并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	3	
k	2	
U	{E,H}	
S	{s_E=0, s_H=1}	



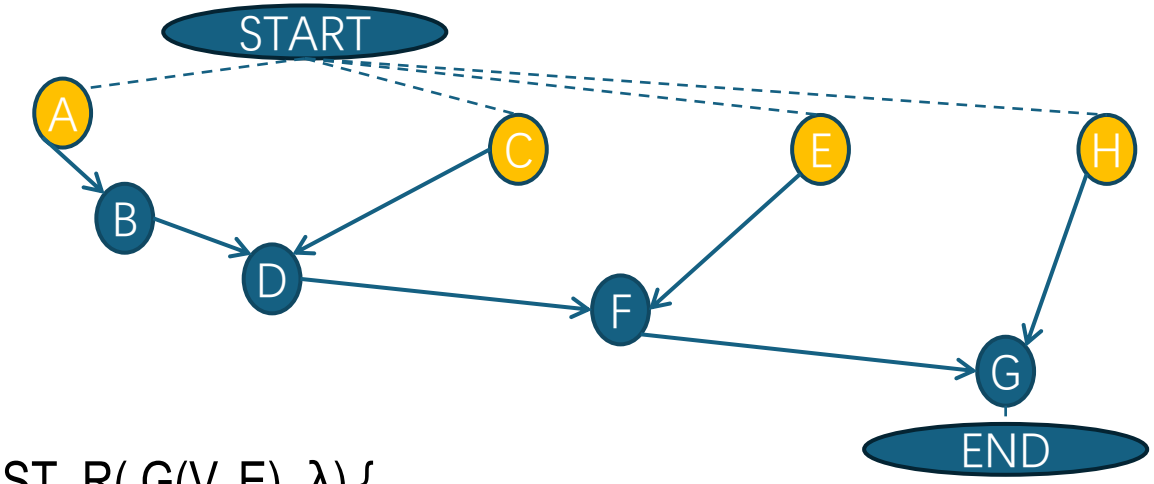
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	4	
k	2	
U	{E,H}	
S	{s_E=0, s_H=1}	



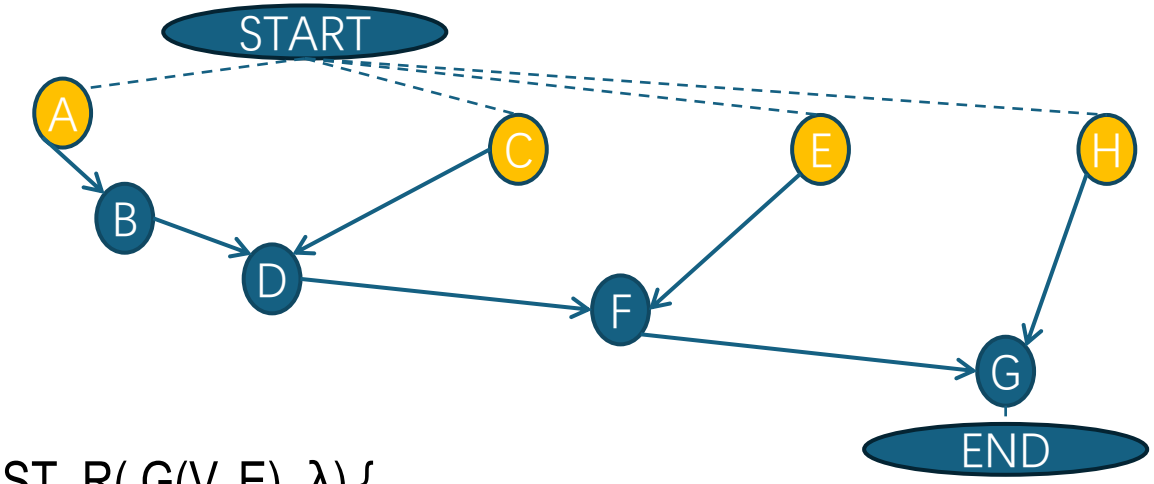
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	4	
k	1	
U	{E,H}	
S	{s_E=0, s_H=1}	



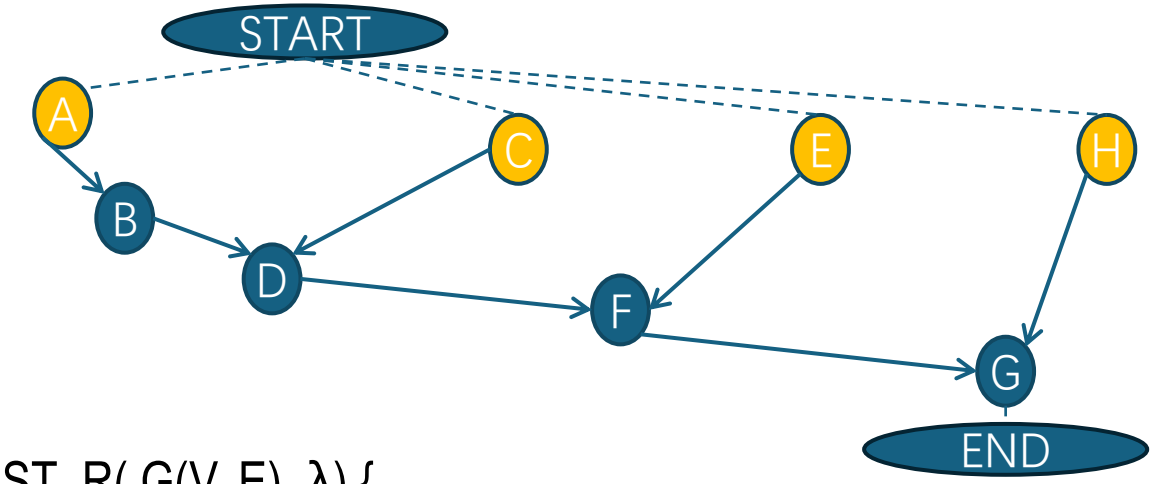
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	4	
k	1	
U	{D}	
S	$\{s_D=0\}$	



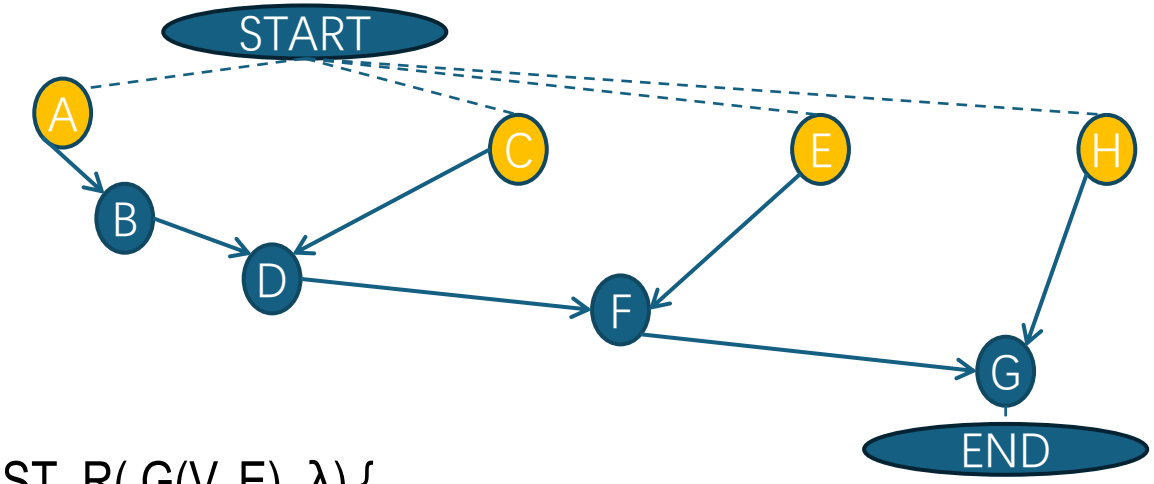
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3
D 4

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点， 计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作， 并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	4	
k	1	
U	{D}	
S	{s_D=0}	



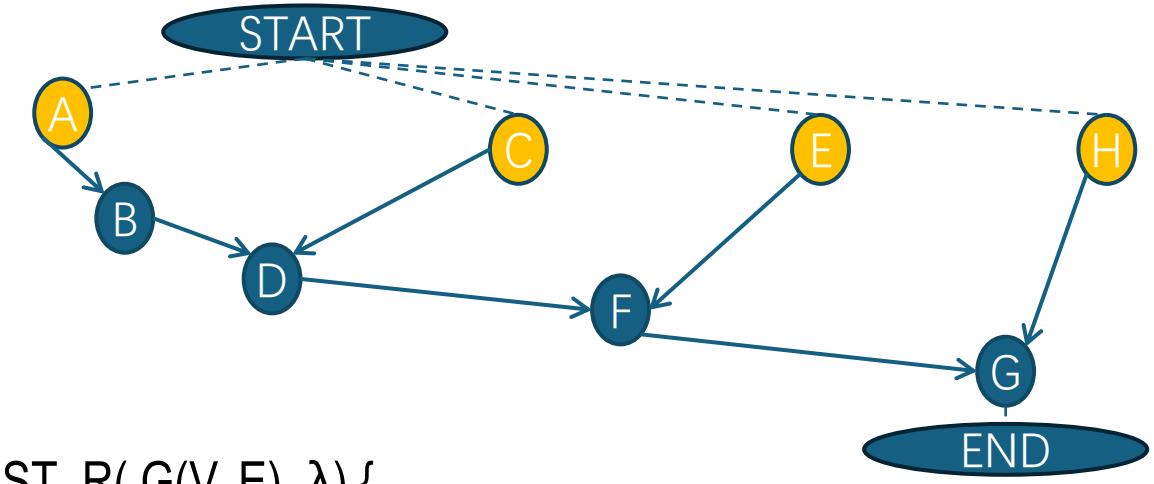
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3
D 4

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	4	
k	2	
U	{D}	
S	{s_D=0}	



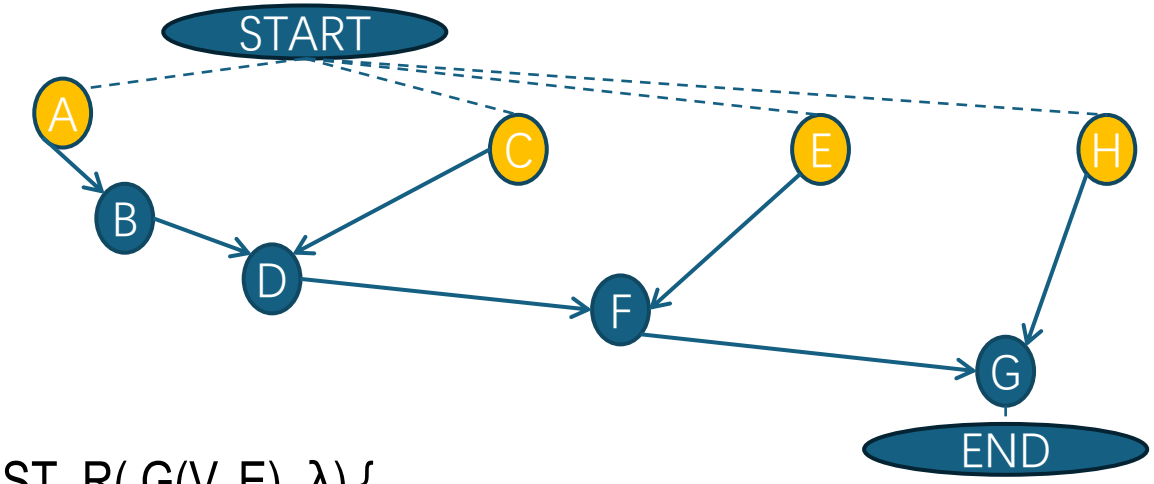
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3
D 4
H 4

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	4	
k	2	
U	{H}	
S	$\{s_H=0\}$	



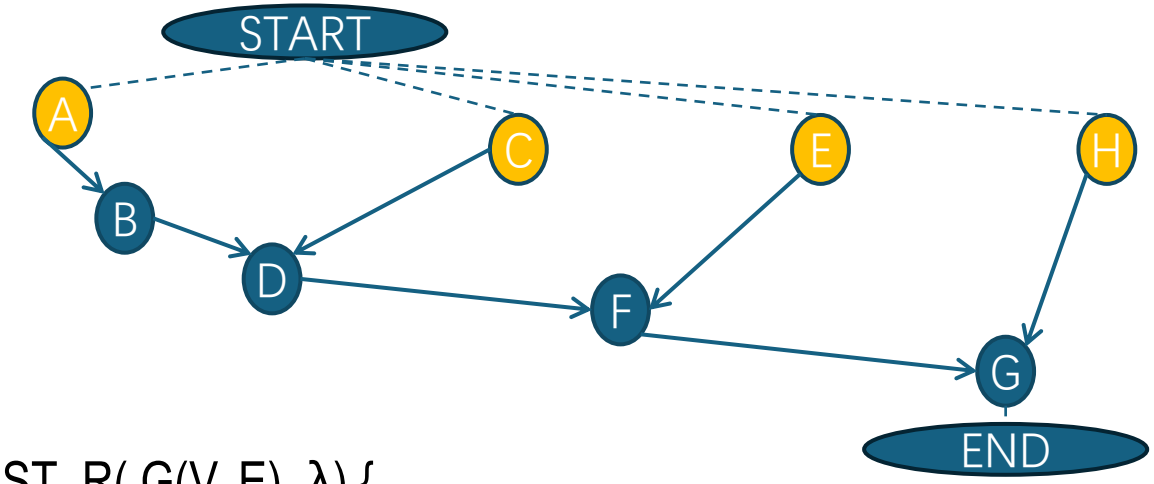
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3
D 4
H 4

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	5	
k	2	
U	{H}	
S	{s_H=0}	



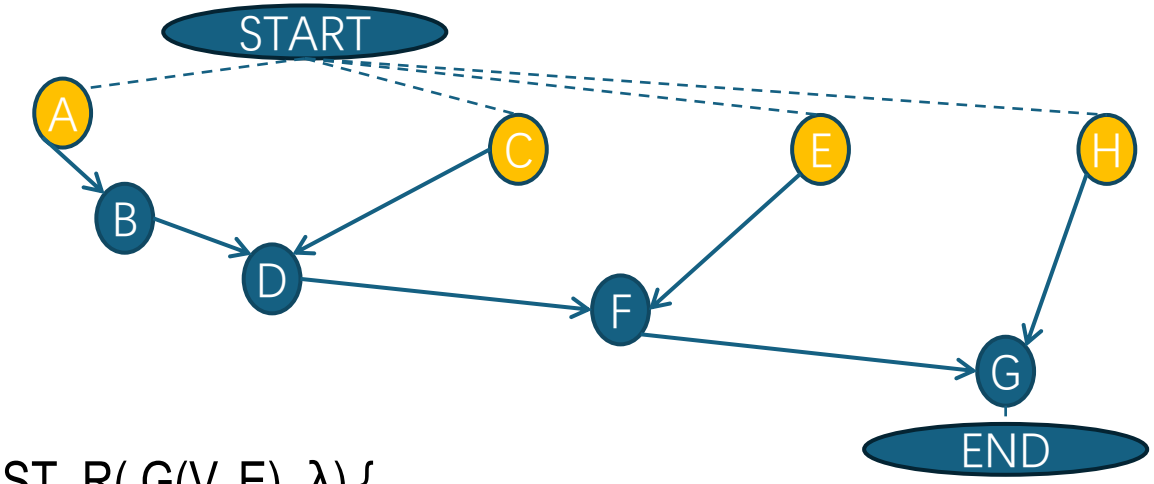
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3
D 4
H 4

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	5	
k	1	
U	{H}	
S	{s_H=0}	



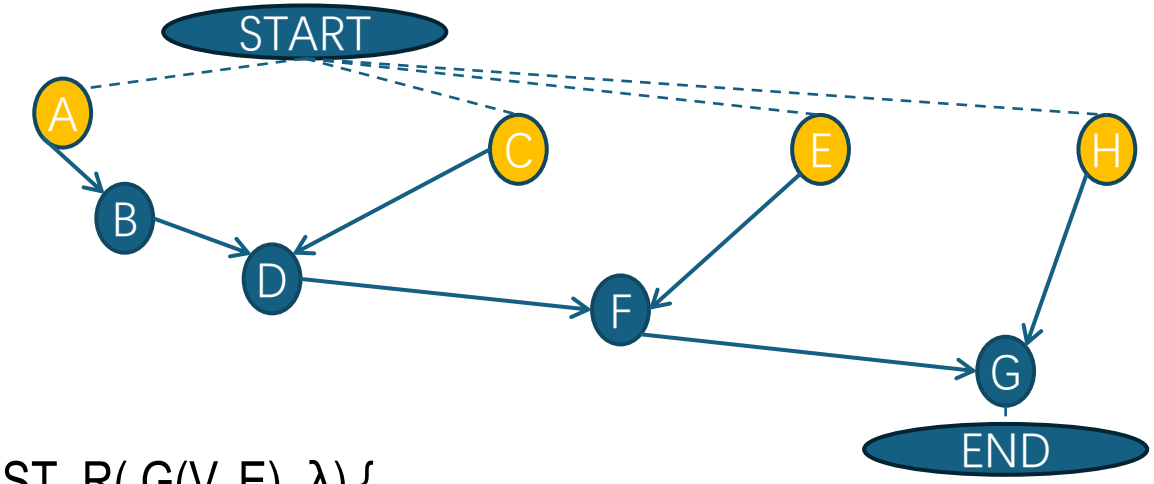
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3
D 4
H 4
F 5

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	5	
k	1	
U	{F}	
S	{s_F=0}	



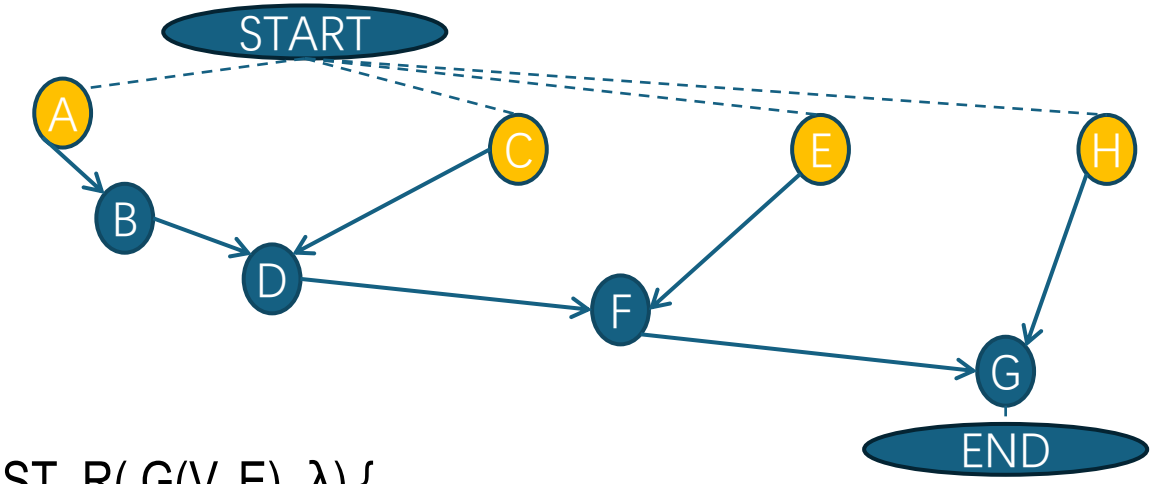
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3
D 4
H 4
F 5

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, ..., n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	5	
k	2	
U	{}	
S	{}	



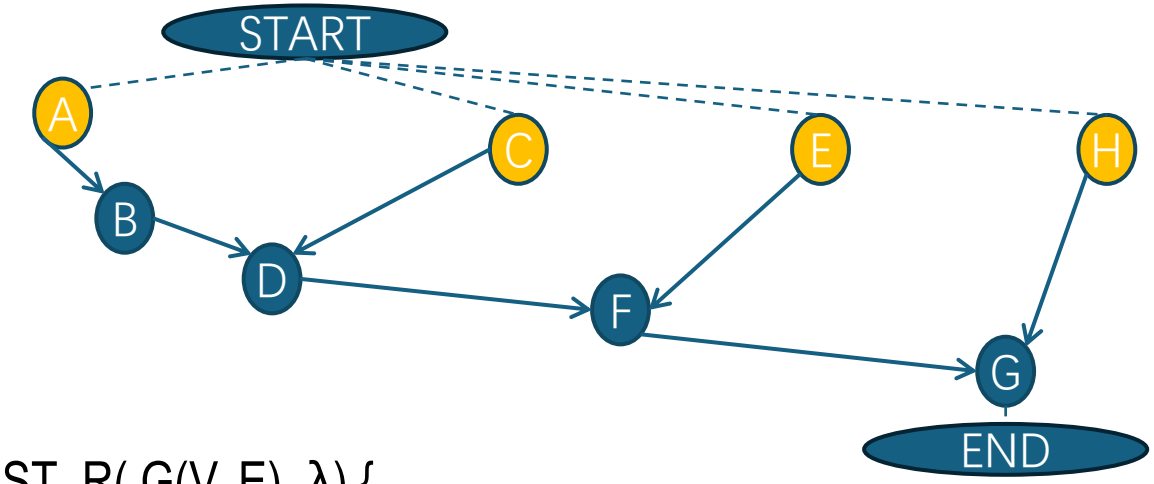
A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3
D 4
H 4
F 5

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	6	
k	1	
U	{F}	
S	{s_F=0}	



A 1
B 3
C 2
D 4
E 3
F 5
H 4
G 6

START 0
A 1
C 2
B 3
E 3
D 4
H 4
F 5

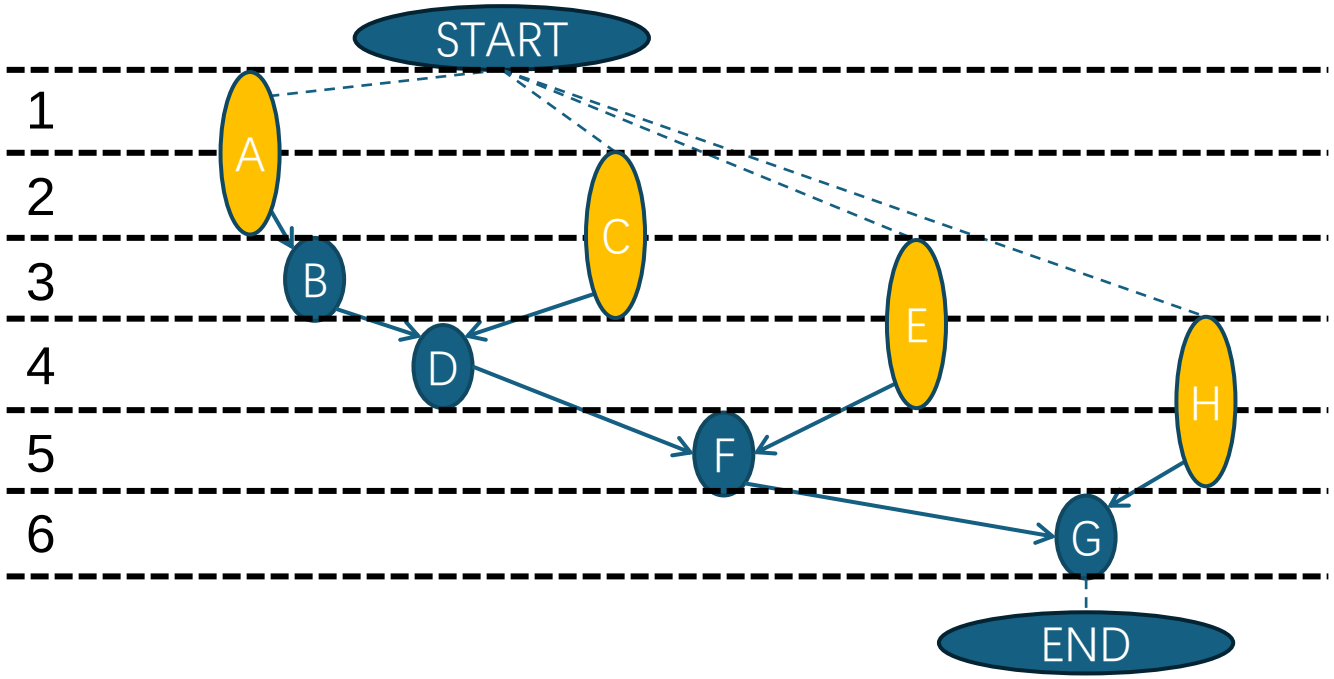
G 6
END 7

LIST_R(G(V, E), λ) {
a = 1; （a代表资源数）
通过 ALAP (G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
if (t^L_0 < 0)
return (Ø);
t_0 = 0; l = 1;
重复执行以下步骤{
对于每种资源类型 k = 1, 2, …, n_{res} {
确定就绪的操作集 $U_{l,k}$;
对$U_{l,k}$中的每个顶点，计算其slacks $s_i = t^L_i - l$;
调度所有松弛度为零的候选操作，并更新 a;
调度不需要额外资源的候选操作;
}
$l = l + 1$;}
直到 vn 被调度完成;
return (t, a);}

k=1为加法操作
k=2为乘操作

变量	当前值	
a	a1	1
	a2	2
l	6	
k	1	
U	{F}	
S	{s_F=0}	

START 0
A 1
C 2
B 3
E 3
D 4
H 4
F 5
G 6
END 7

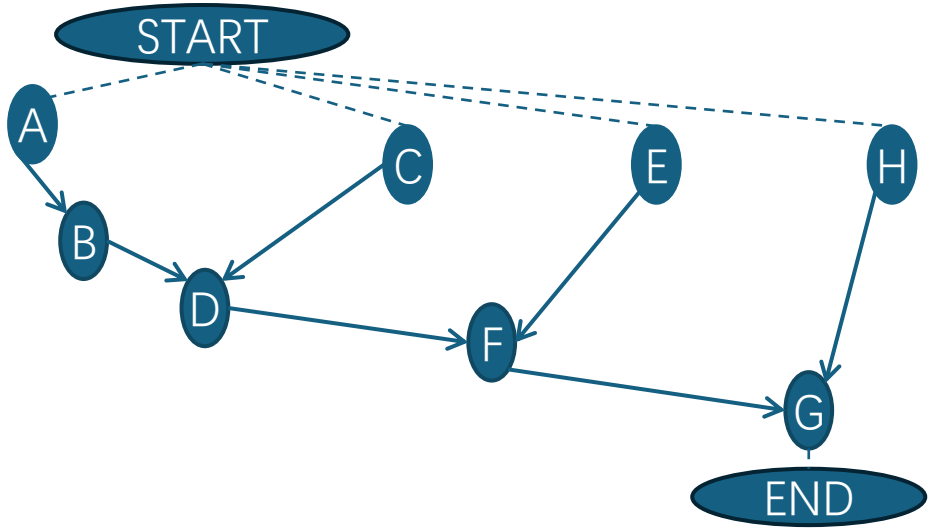


MR-LCS-随堂作业

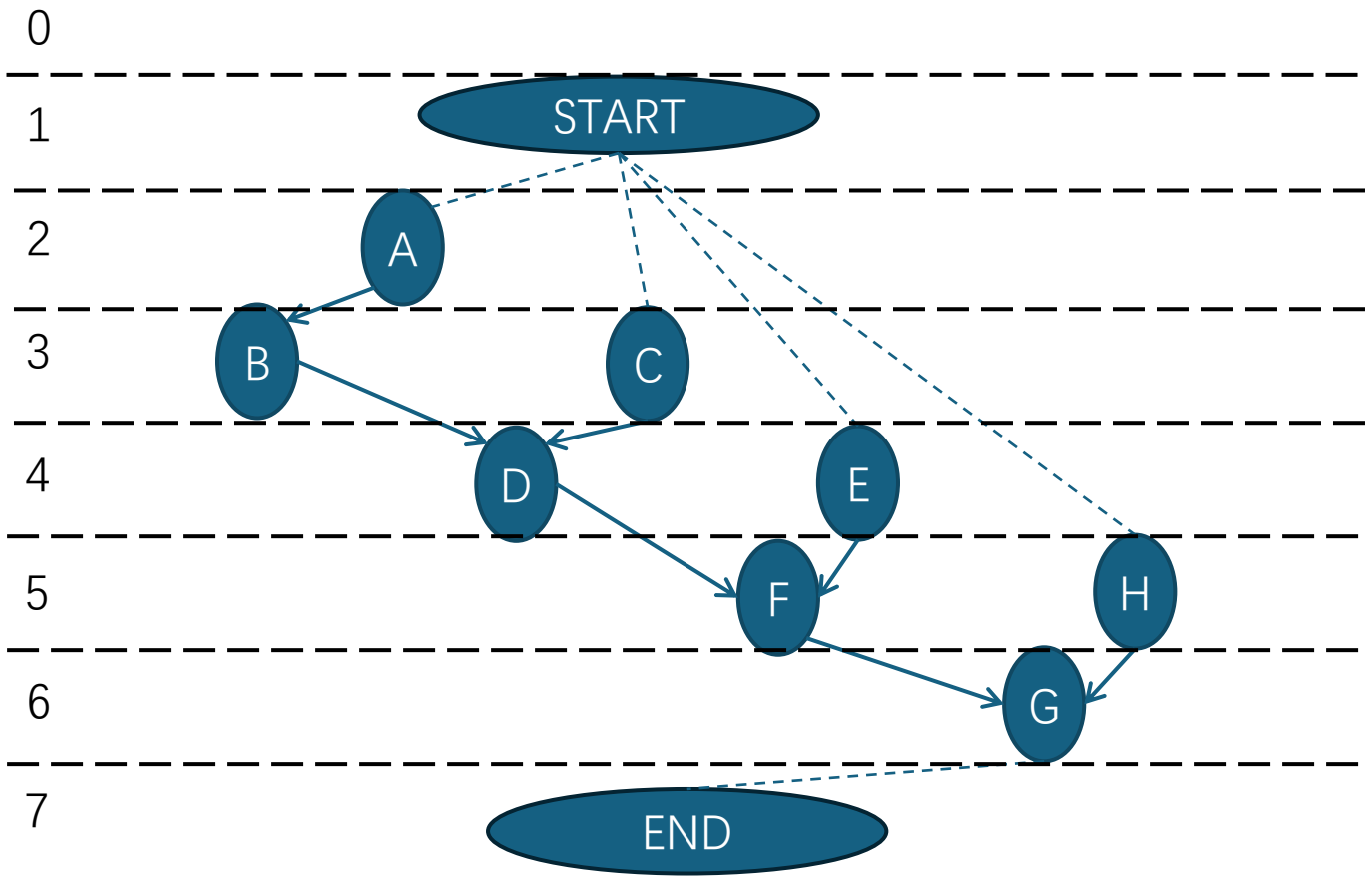
in-class assignment

```
LIST_R( G(V, E), λ) {
        a = 1;  （a代表资源数）
        通过 ALAP ( G(V, E), λ) 计算所有操作的最晚开始时间 t^L ;
        if (t^L_0 < 0)
            return (Ø);
        t_0 = 0;   l = 1;
        重复执行以下步骤{
            对于每种资源类型 k = 1, 2, …, n_res {
                确定就绪的操作集 U_l,k;
                对U_l,k中的每个顶点，计算其slacks s_i = t^L_i – l ;
                调度所有松弛度为零的候选操作，并更新 a;
                调度不需要额外资源的候选操作;
            }
            l = l + 1;}
        直到 vn 被调度完成;
        return (t, a);}
```

若图中所有结点都为加法器（一个周期完成），要求在六个周期完成全部调度（不包括虚拟节点）。若使用列表调度算法，下图对应的最终调度图应该如何，请画出



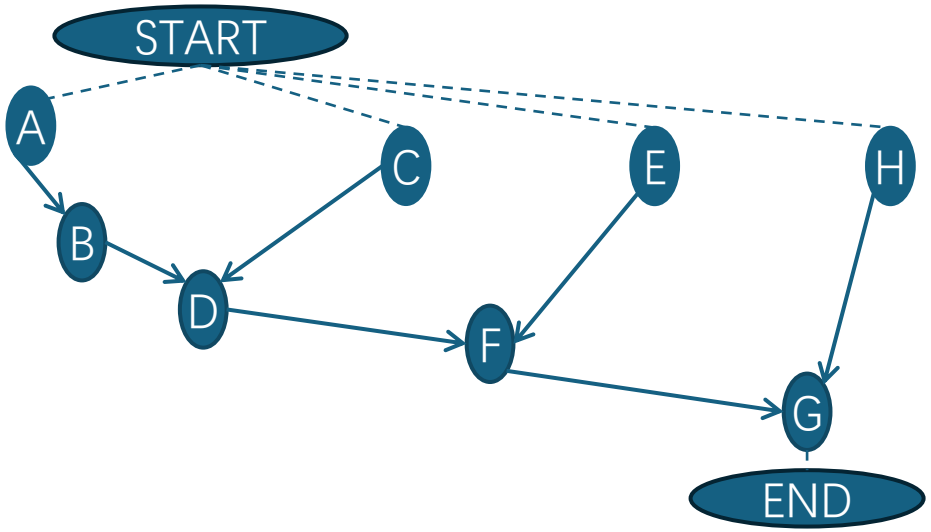
ALAP调度结果



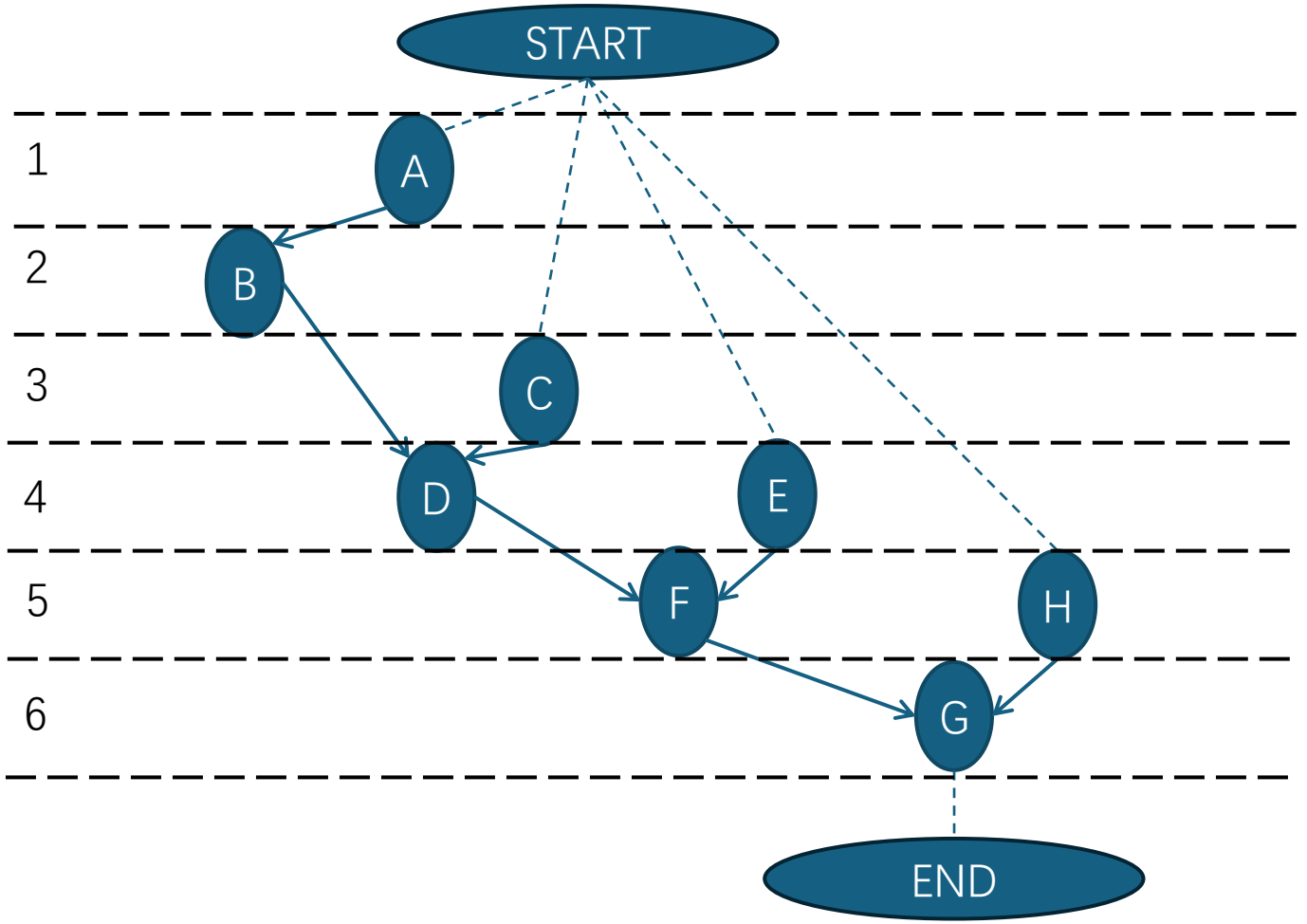
A 2
B 3
C 3
D 4
E 4
F 5
H 5
G 6

ALAP调度结果:

A 2
B 3
C 3
D 4
E 4
F 5
H 5
G 6

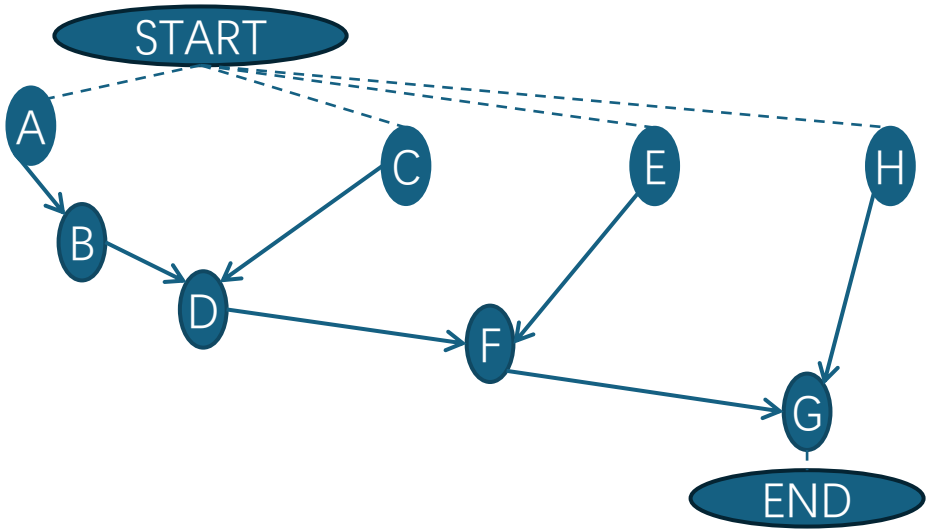


列表调度结果:

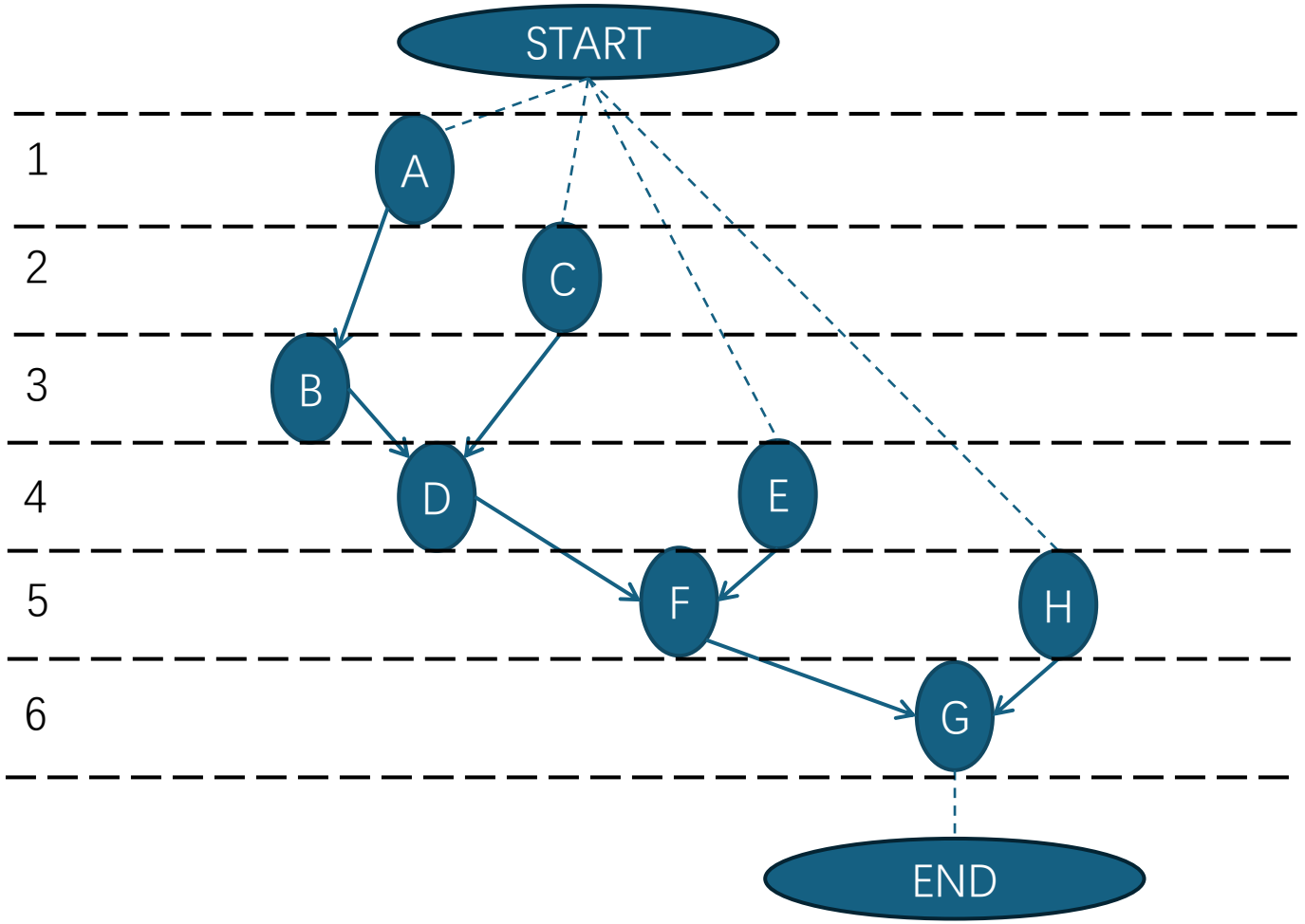


ALAP调度结果:

A 2
B 3
C 3
D 4
E 4
F 5
H 5
G 6

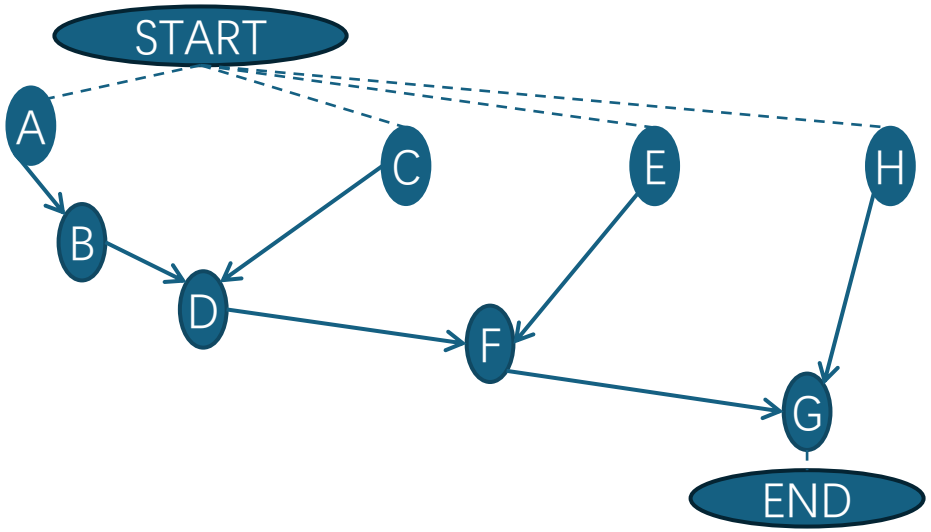


列表调度结果:

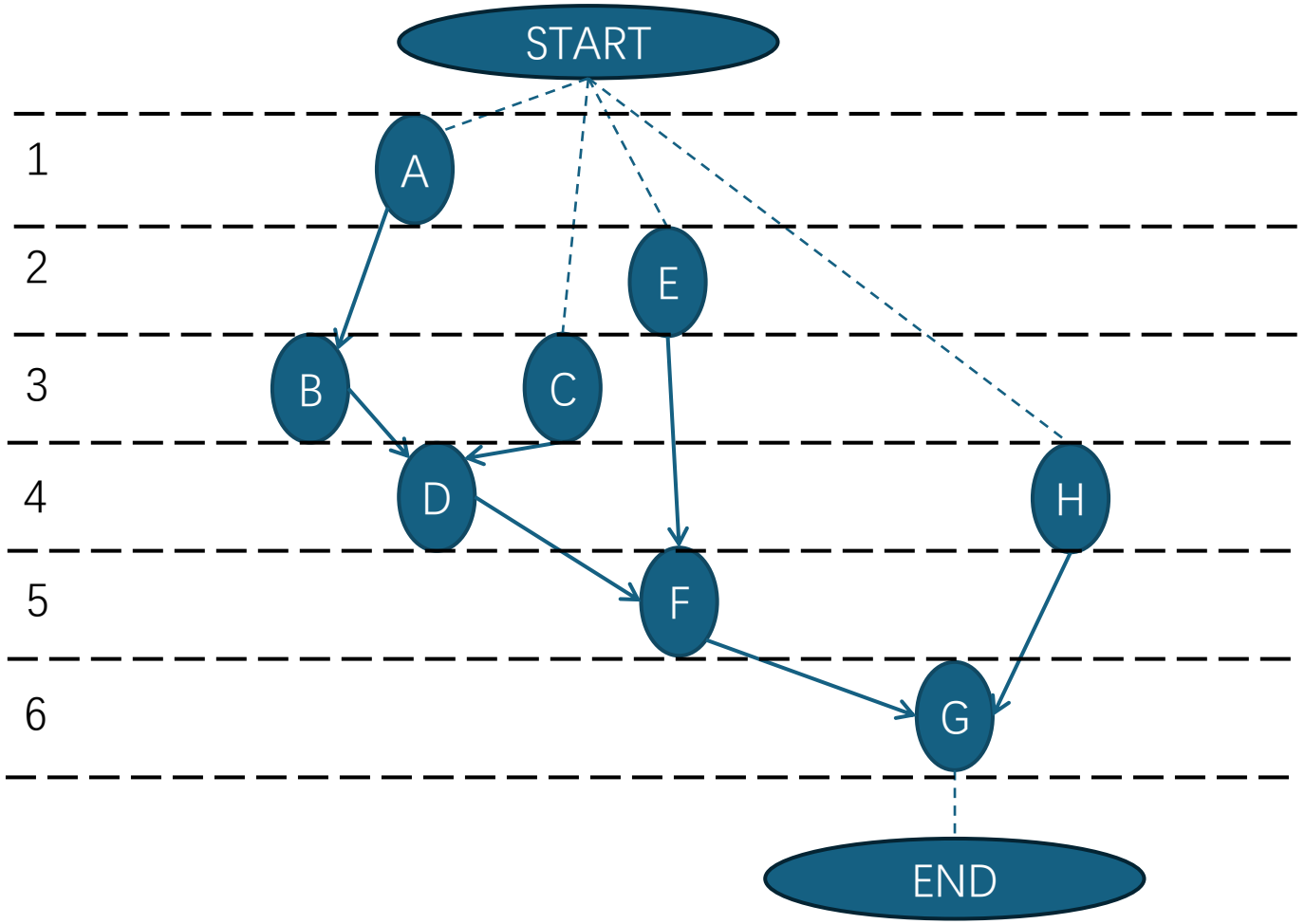


ALAP调度结果:

A 2
B 3
C 3
D 4
E 4
F 5
H 5
G 6

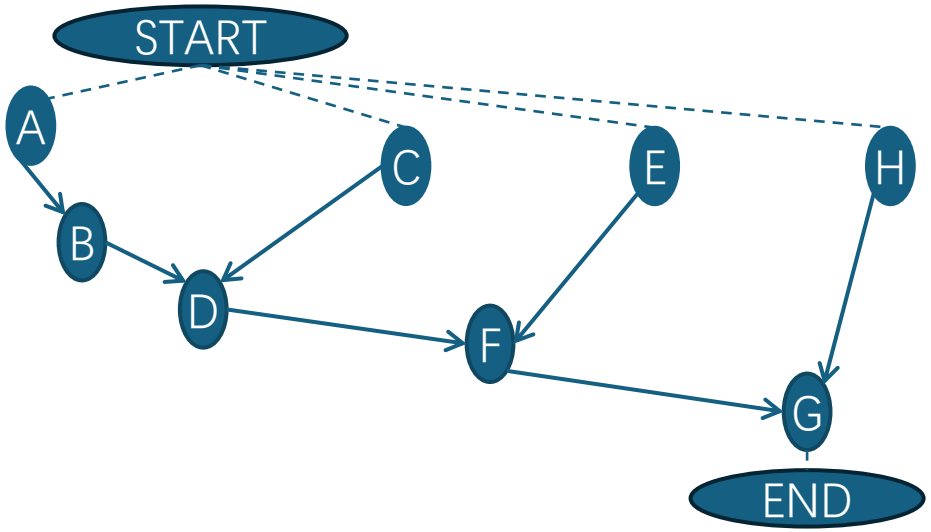


列表调度结果:

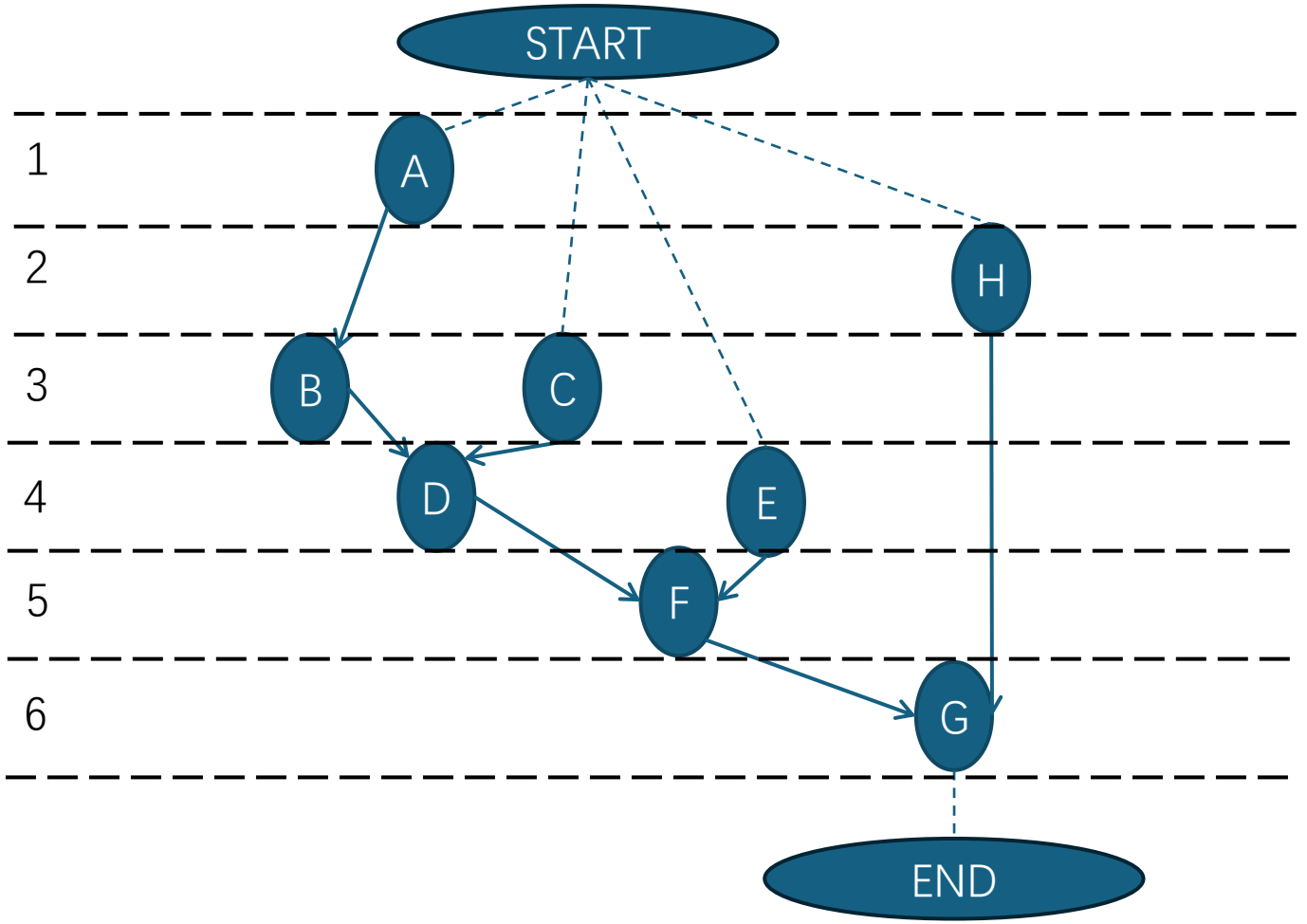


ALAP调度结果:

A 2
B 3
C 3
D 4
E 4
F 5
H 5
G 6

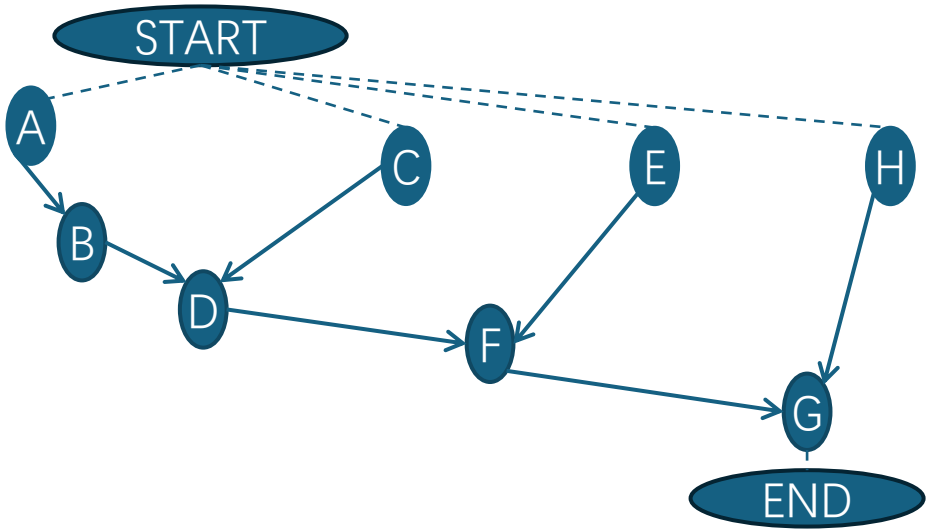


列表调度结果:

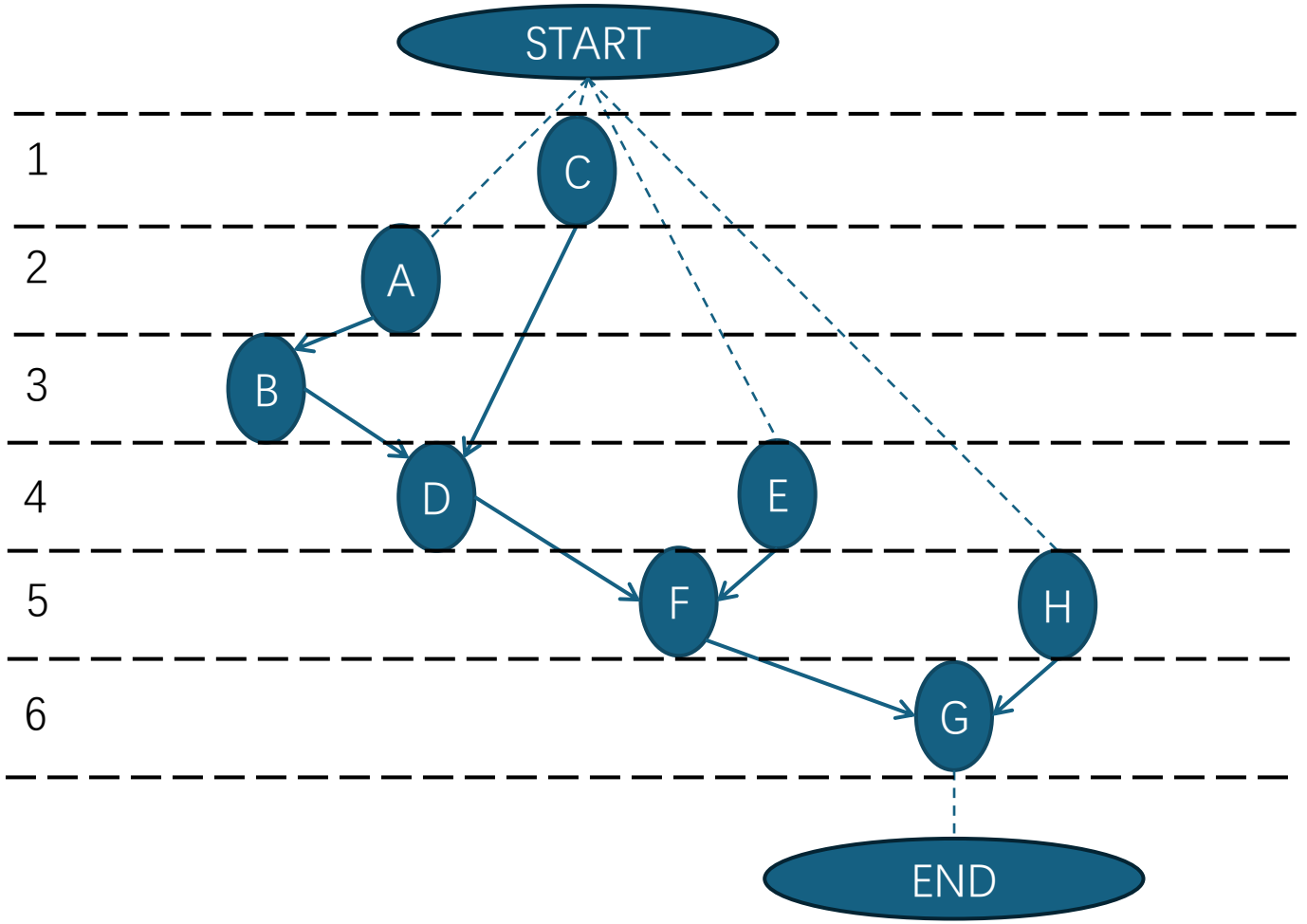


ALAP调度结果:

A 2
B 3
C 3
D 4
E 4
F 5
H 5
G 6

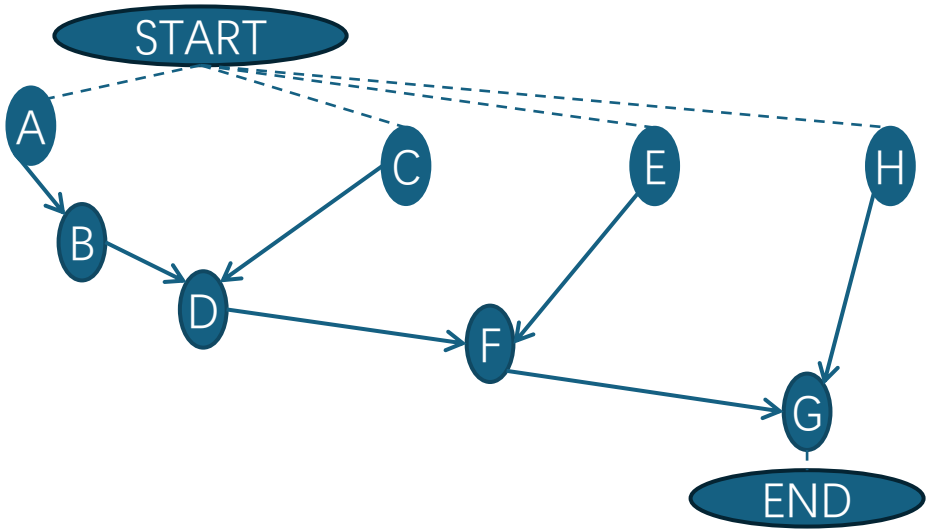


列表调度结果:

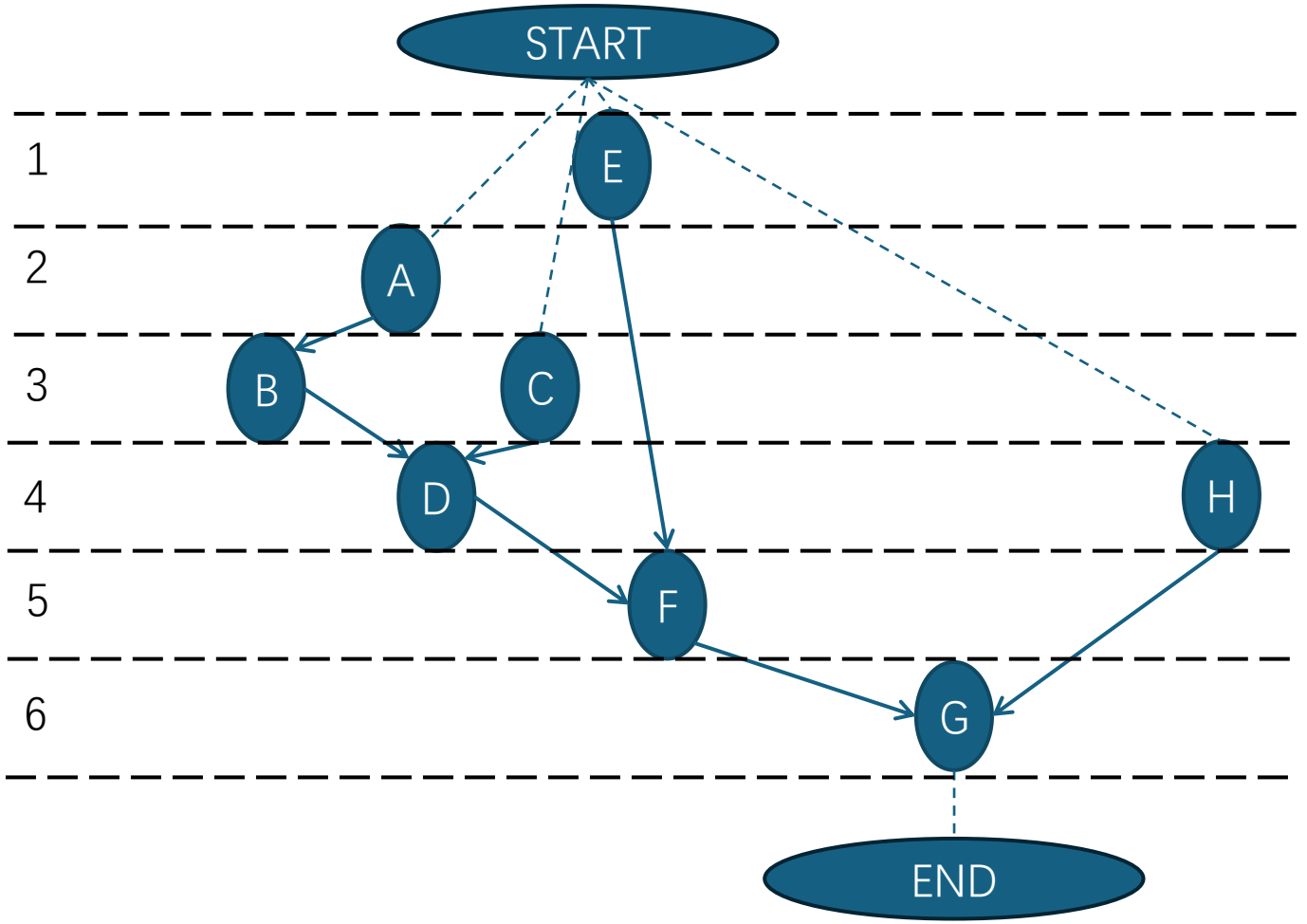


ALAP调度结果:

A 2
B 3
C 3
D 4
E 4
F 5
H 5
G 6

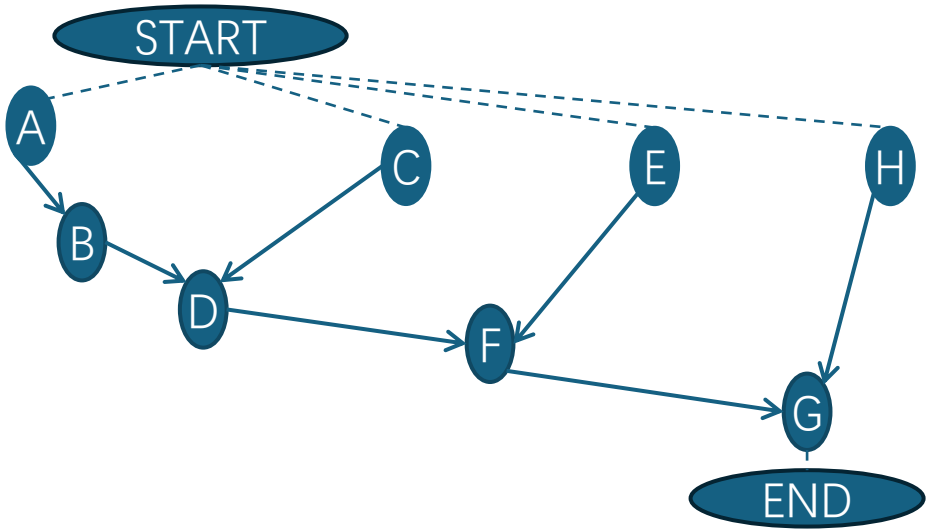


列表调度结果:

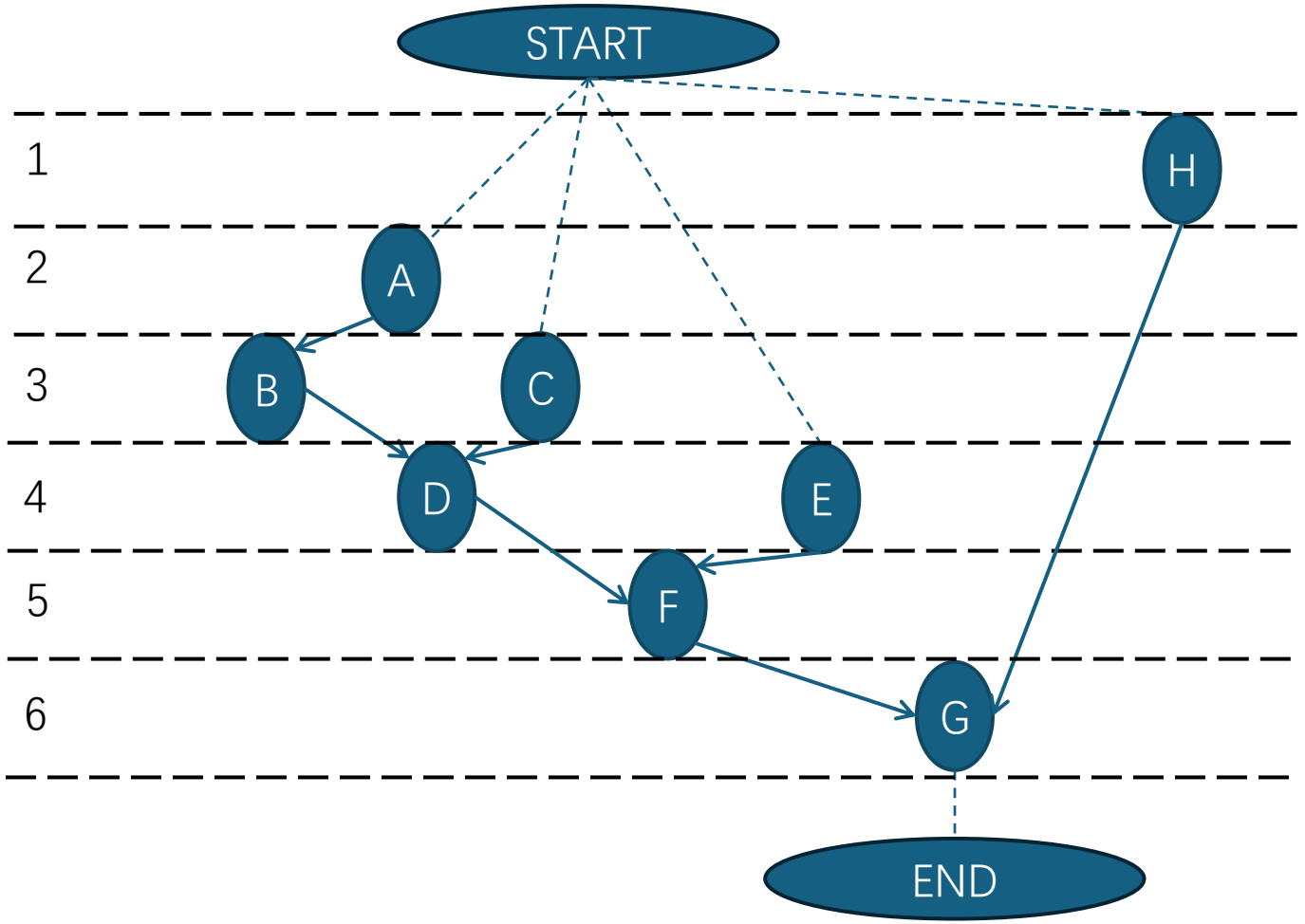


ALAP调度结果:

A 2
B 3
C 3
D 4
E 4
F 5
H 5
G 6



列表调度结果:



静态时序分析

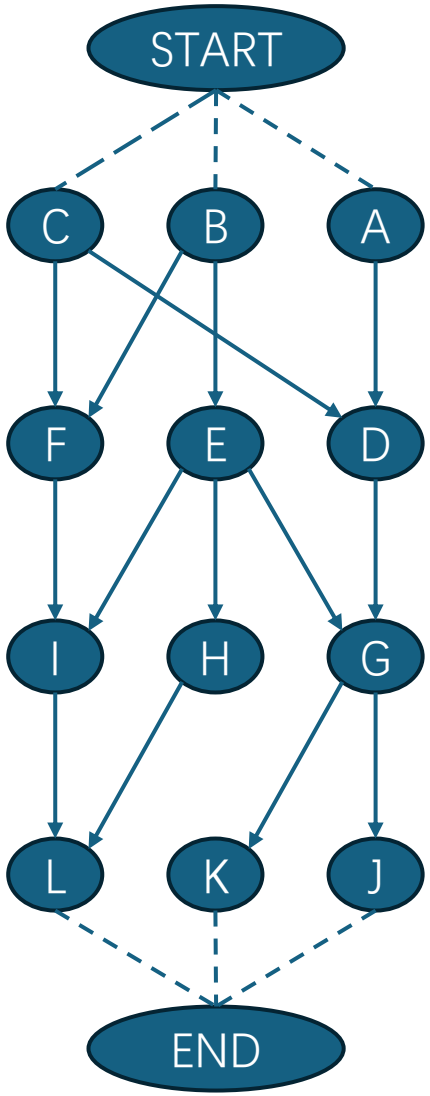
相关术语

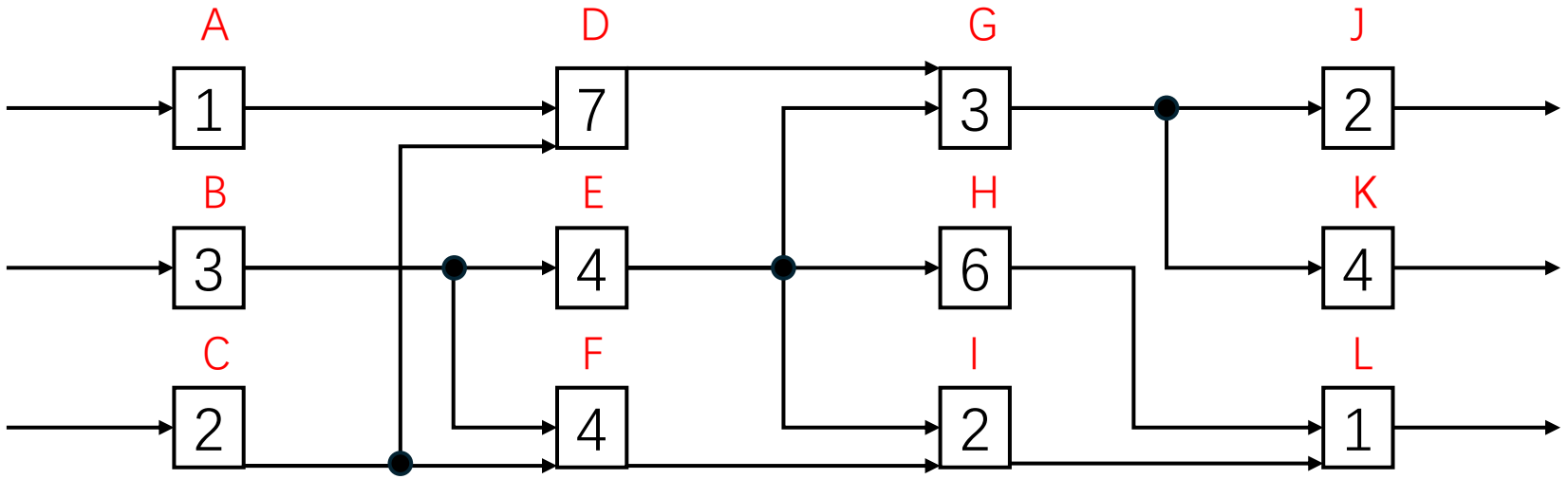
Related Terminologies

- 静态时序分析（Static Timing Analysis）
  - 到达时间（Arrival Time）
  - 要求时间（Required Time）
  - 裕量（Slack）
  - 关键路径（critical node）
  - 关键节点（critical node）
  - 关键性（Criticality）

时间分析

Timing Analysis



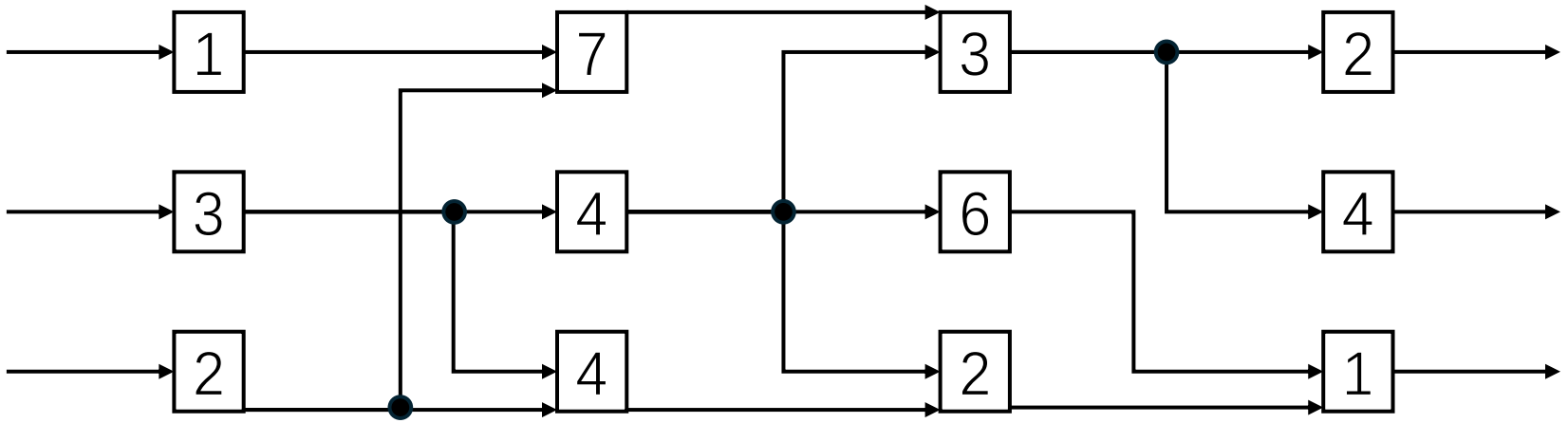


时间分析

Timing Analysis

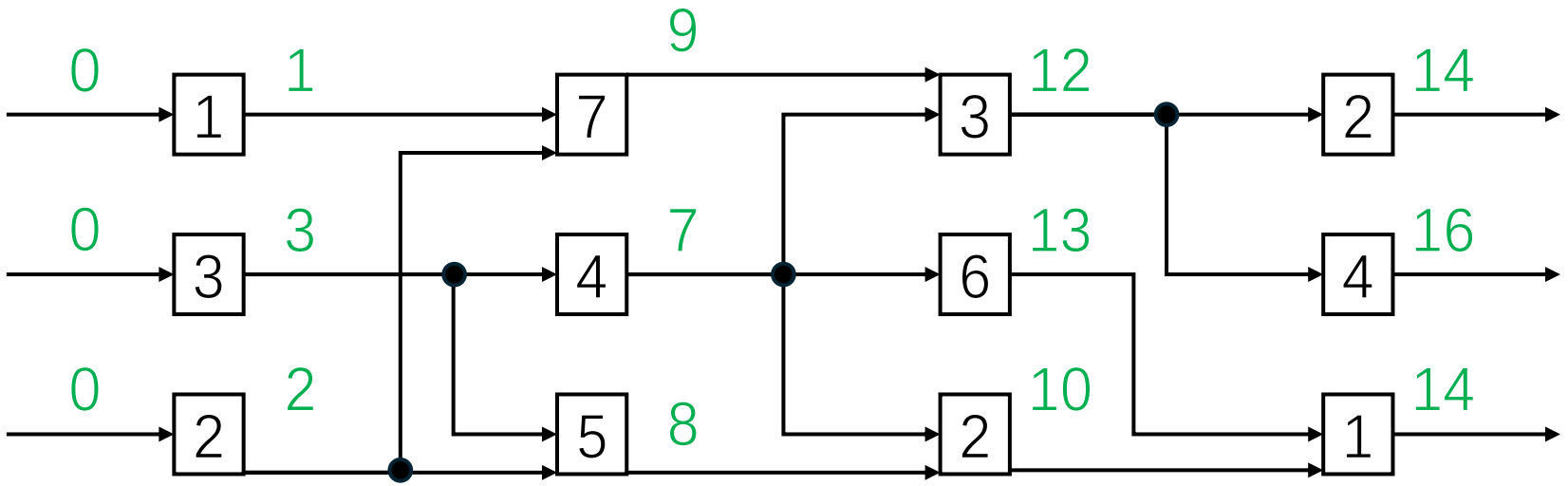
原始顺序图

（包含延迟信息）



到达时间

（Arrival Time）

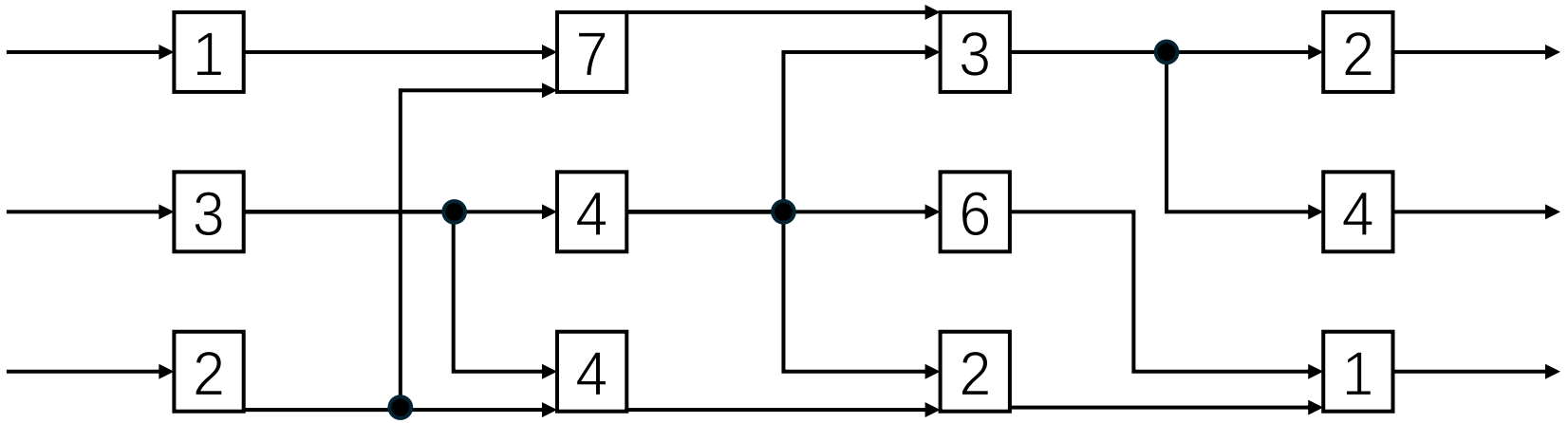


时间分析

Timing Analysis

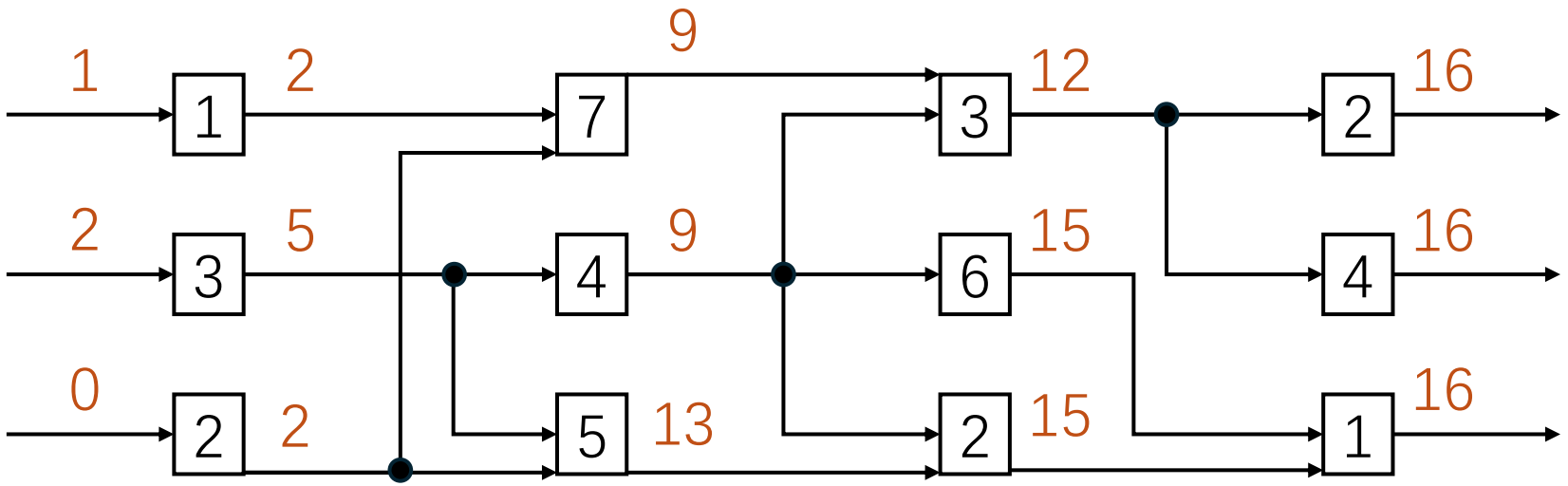
原始顺序图

（包含延迟信息）



要求时间

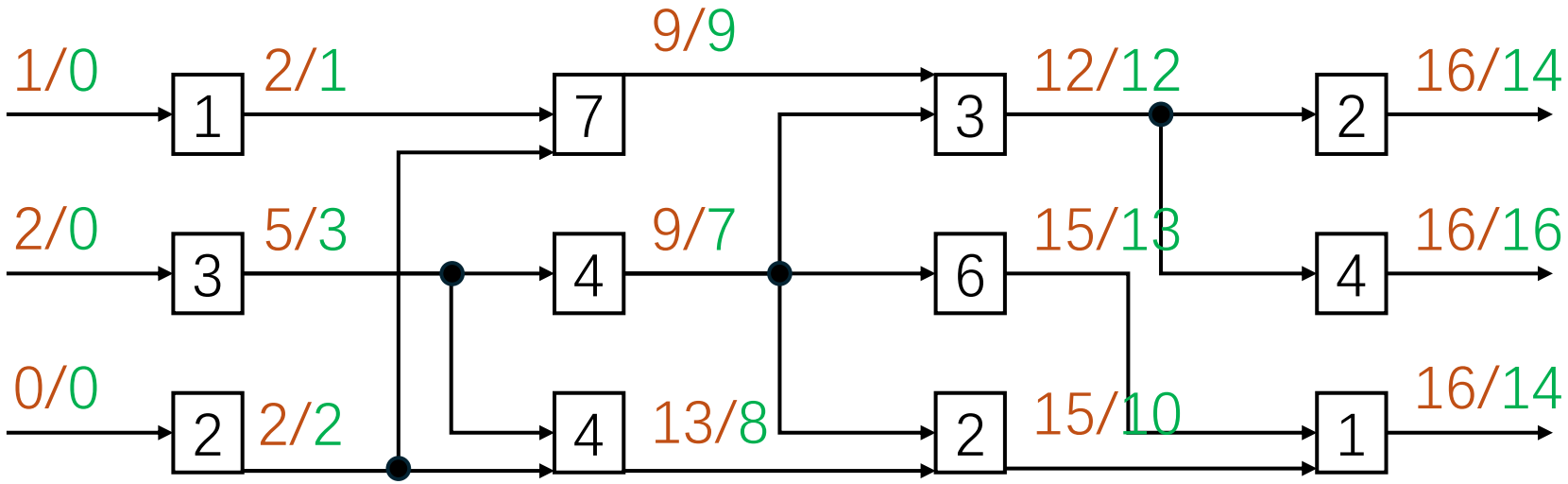
(Required Time)



时间分析

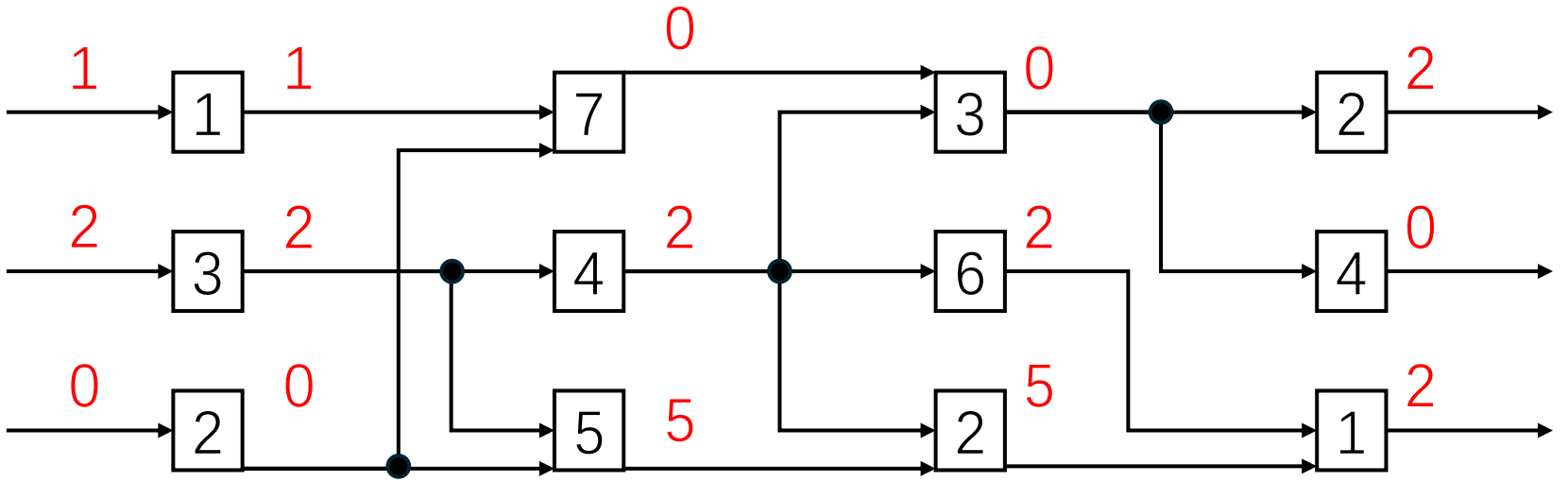
Timing Analysis

要求时间/到达时间



裕量（Slack）=

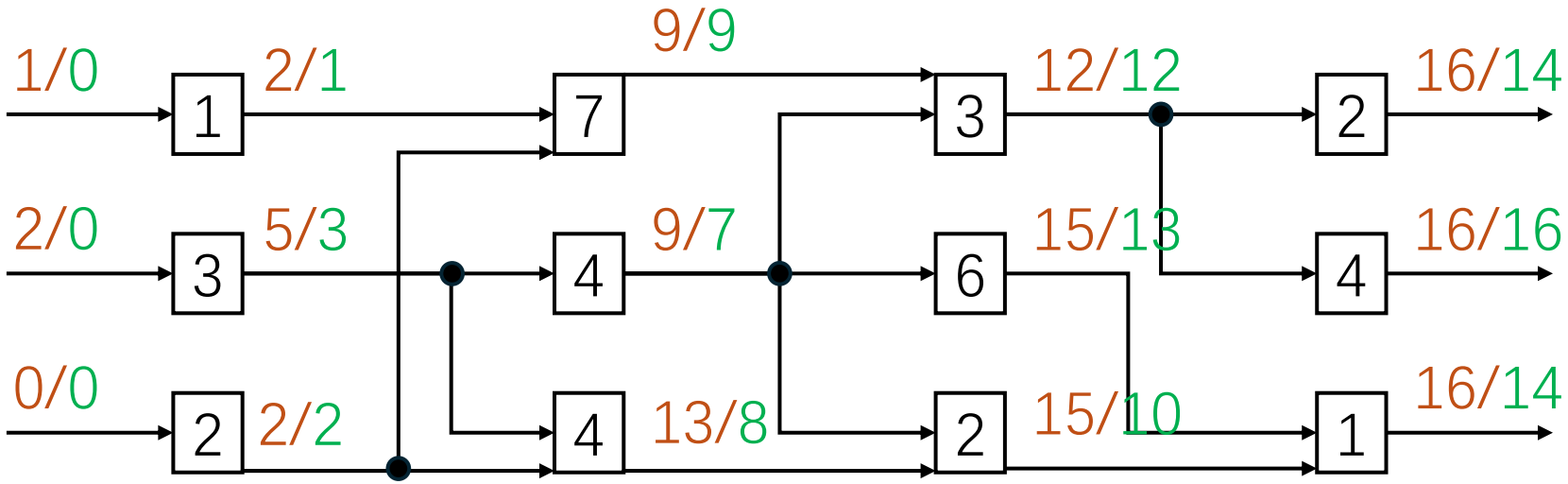
要求时间-到达时间



时间分析

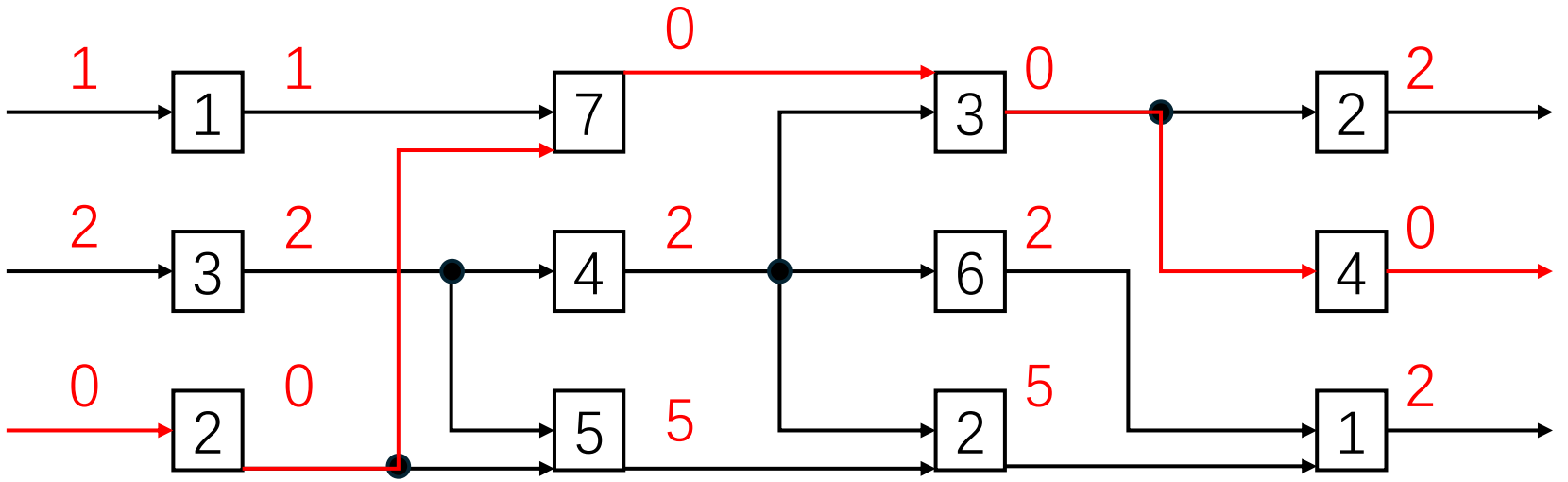
Timing Analysis

要求时间/到达时间



关键路径

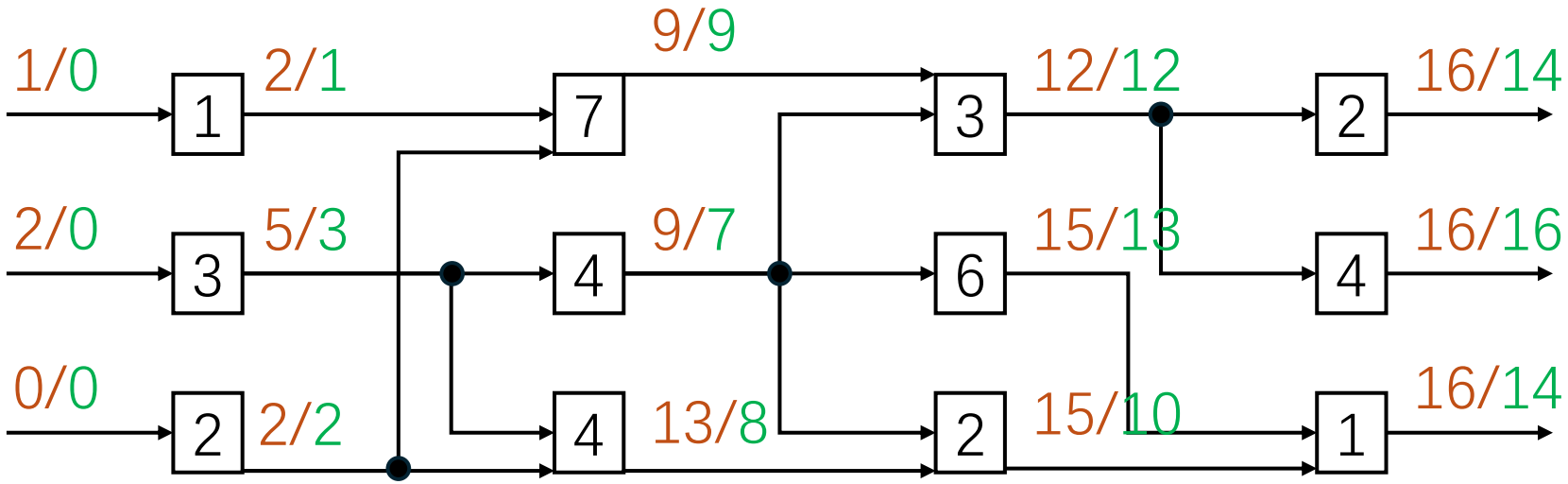
（Critical Path）



时间分析

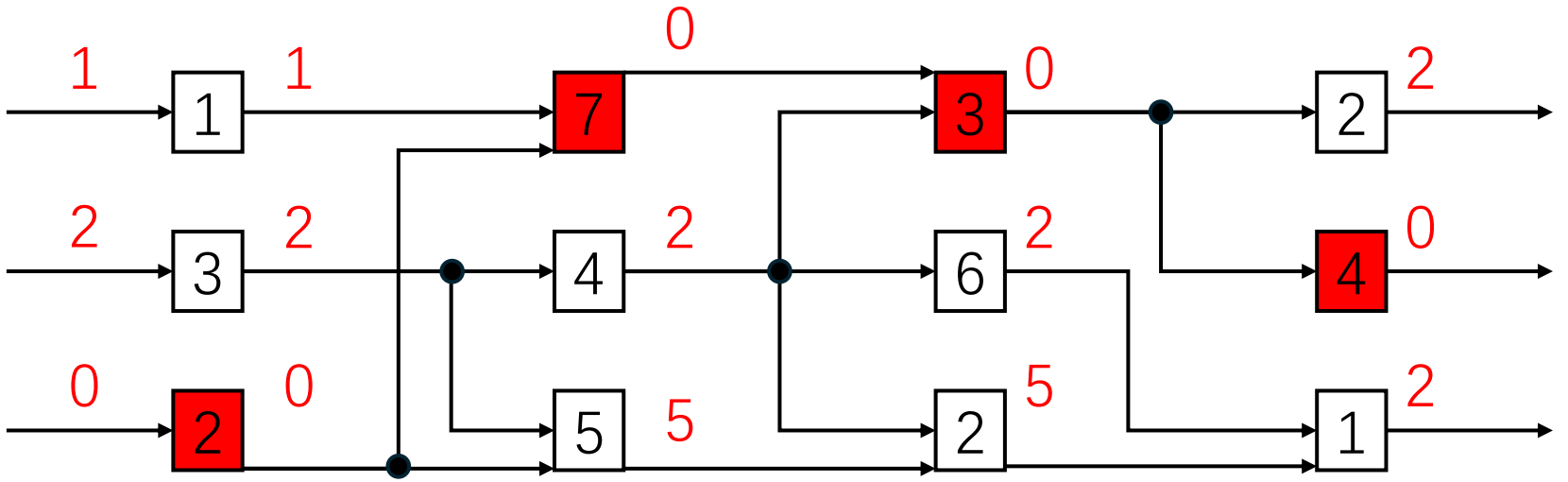
Timing Analysis

要求时间/到达时间



关键结点

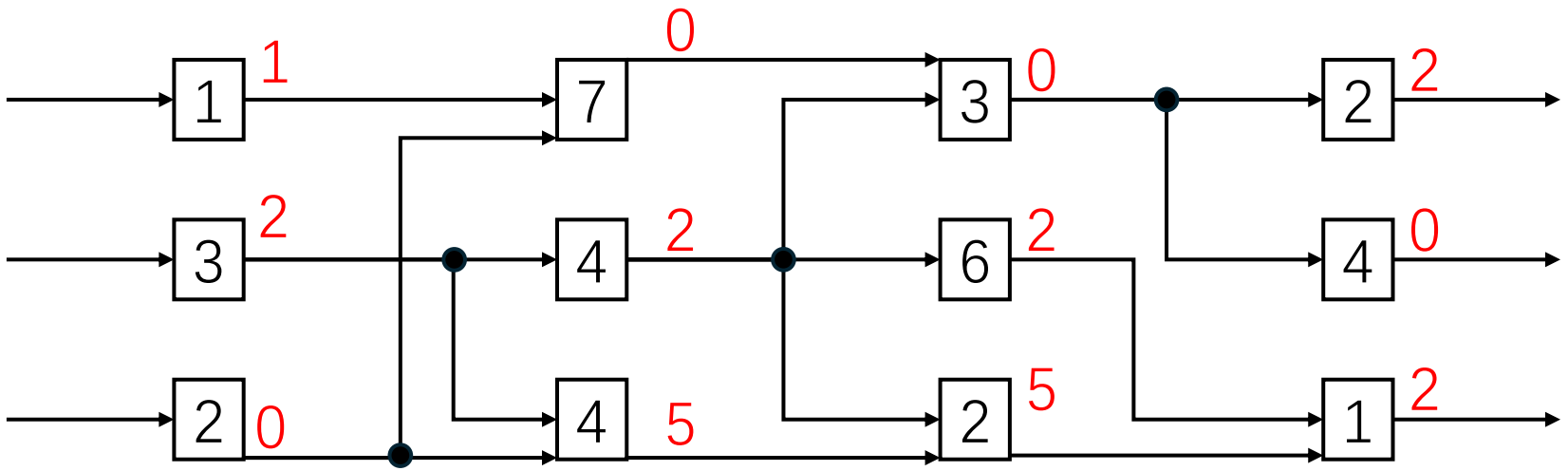
（Critical Node）



时间分析

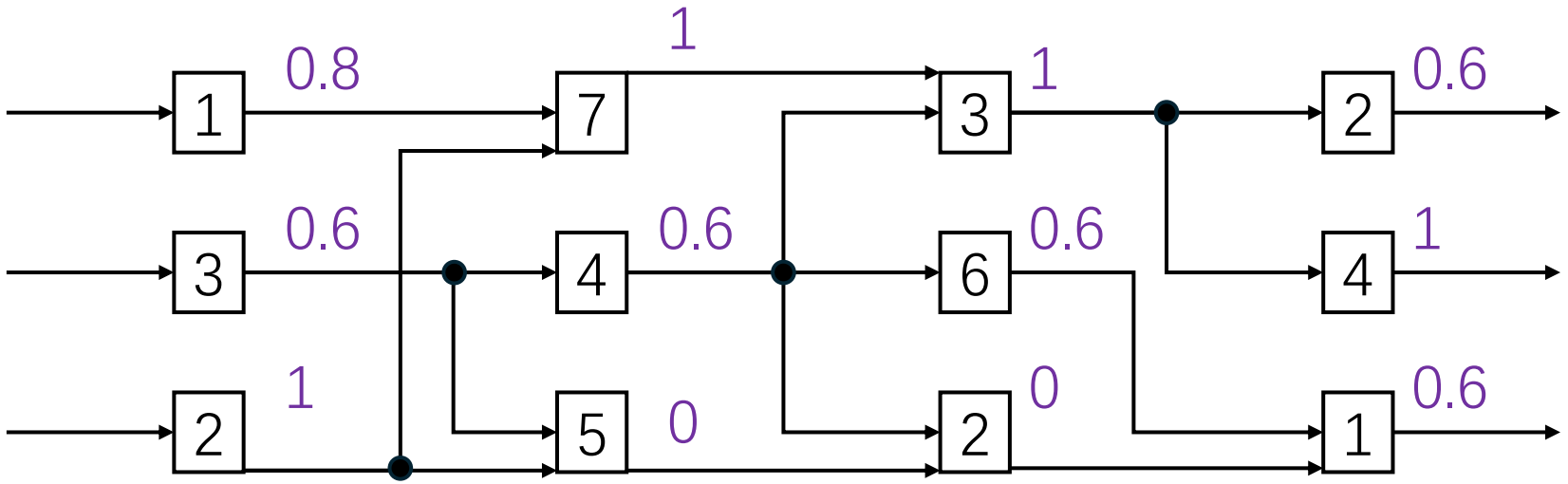
Timing Analysis

裕量（Slack）



关键性（Criticality）

=1-(slack/max_slack)



随堂作业

in-class assignment

请写出以下调度图的关键结点，以及代表每个结点关键性的值