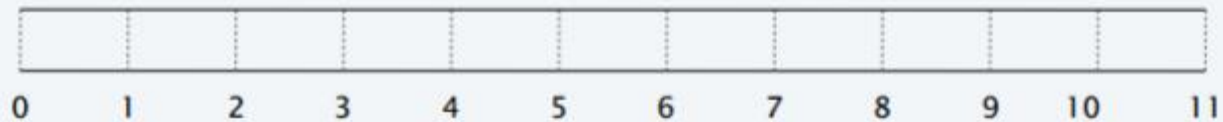
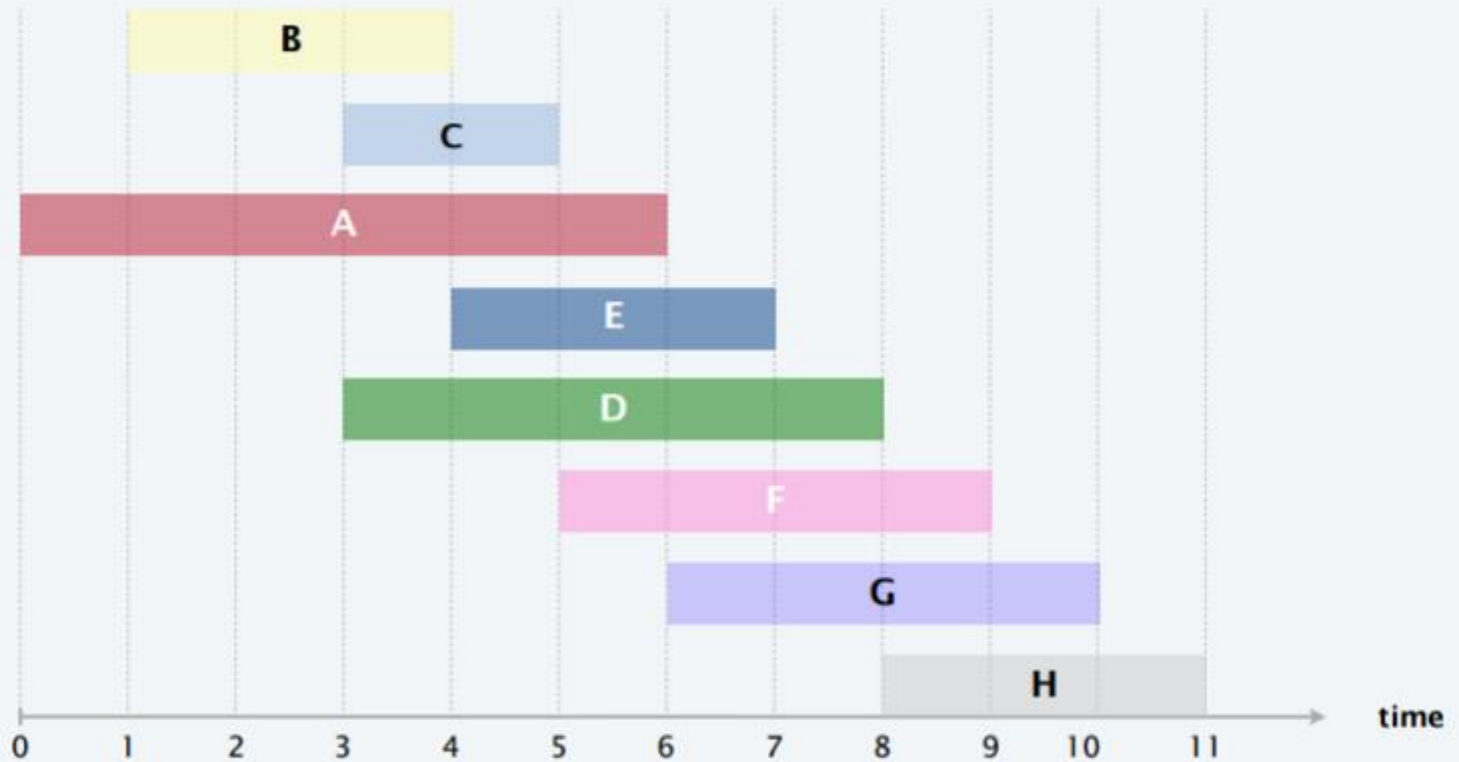
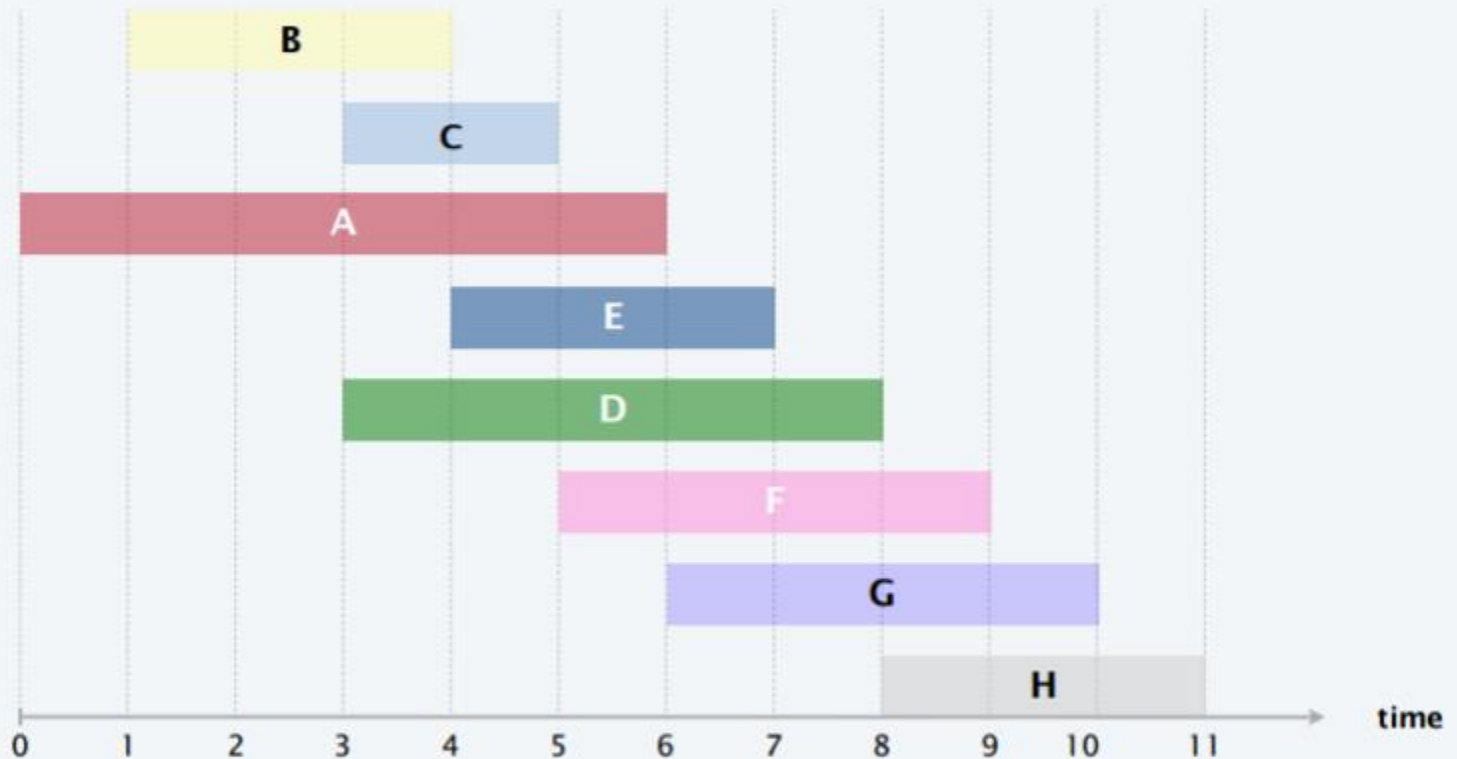


1. 给定以下多个任务及相应的开始和终止时间，求任务的
最大组合数。



1. 给定以下多个任务及相应的开始和终止时间，求任务的
最大组合数。

Earliest-finish-time-first algorithm

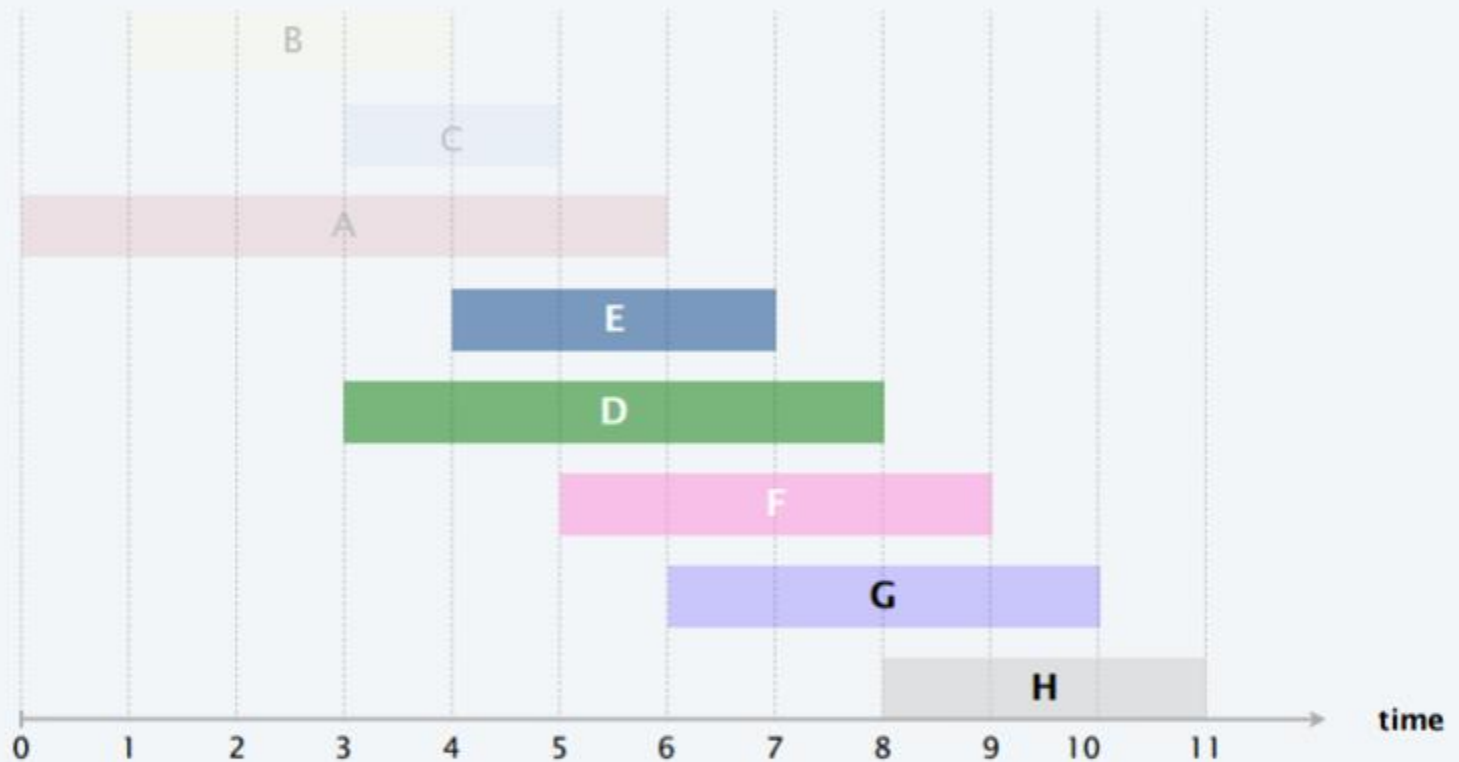


job B is compatible (add to schedule)

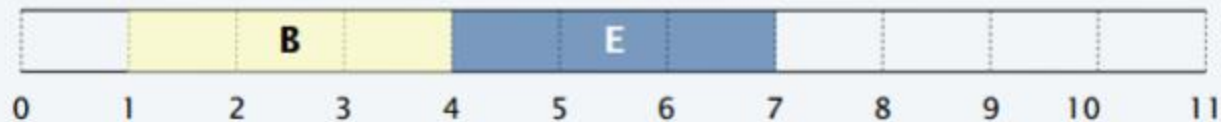


1. 给定以下多个任务及相应的开始和终止时间，求任务的
最大组合数。

Earliest-finish-time-first algorithm



job E is compatible (add to schedule)

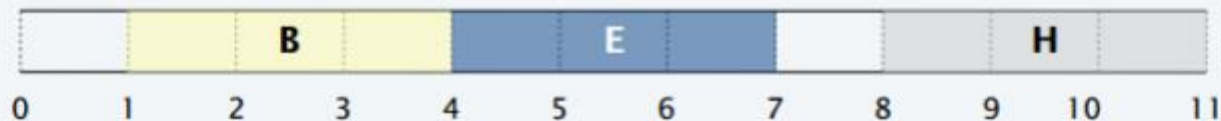


1. 给定以下多个任务及相应的开始和终止时间，求任务的
最大组合数。

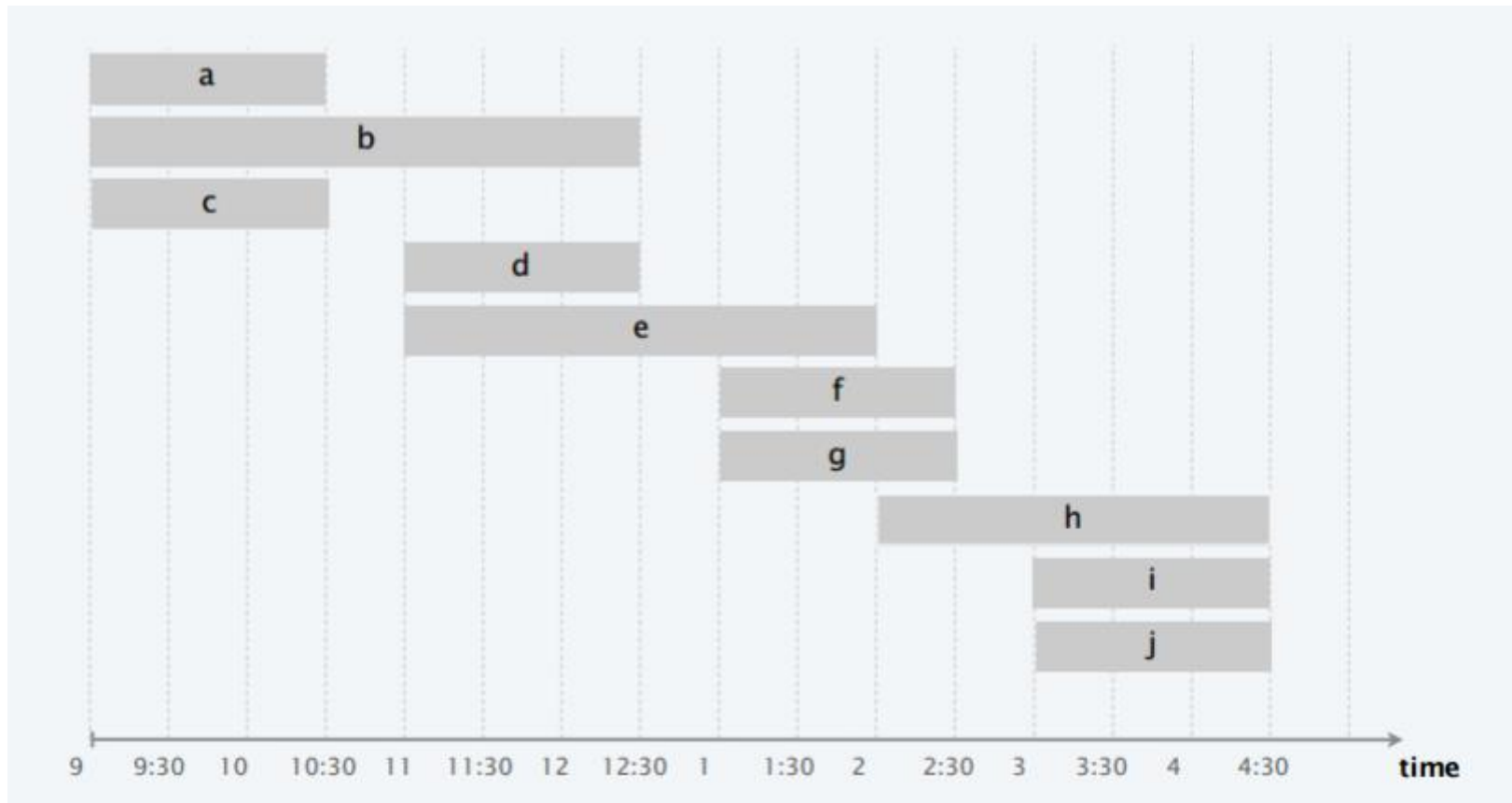
Earliest-finish-time-first algorithm



job H is compatible (add to schedule)



2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。



2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

Earliest-start-time-first algorithm

Consider lectures in order of start time:

- Assign next lecture to any compatible classroom.
- Otherwise, open up a new classroom.

no compatible classroom: open up a new classroom and assign lecture to it

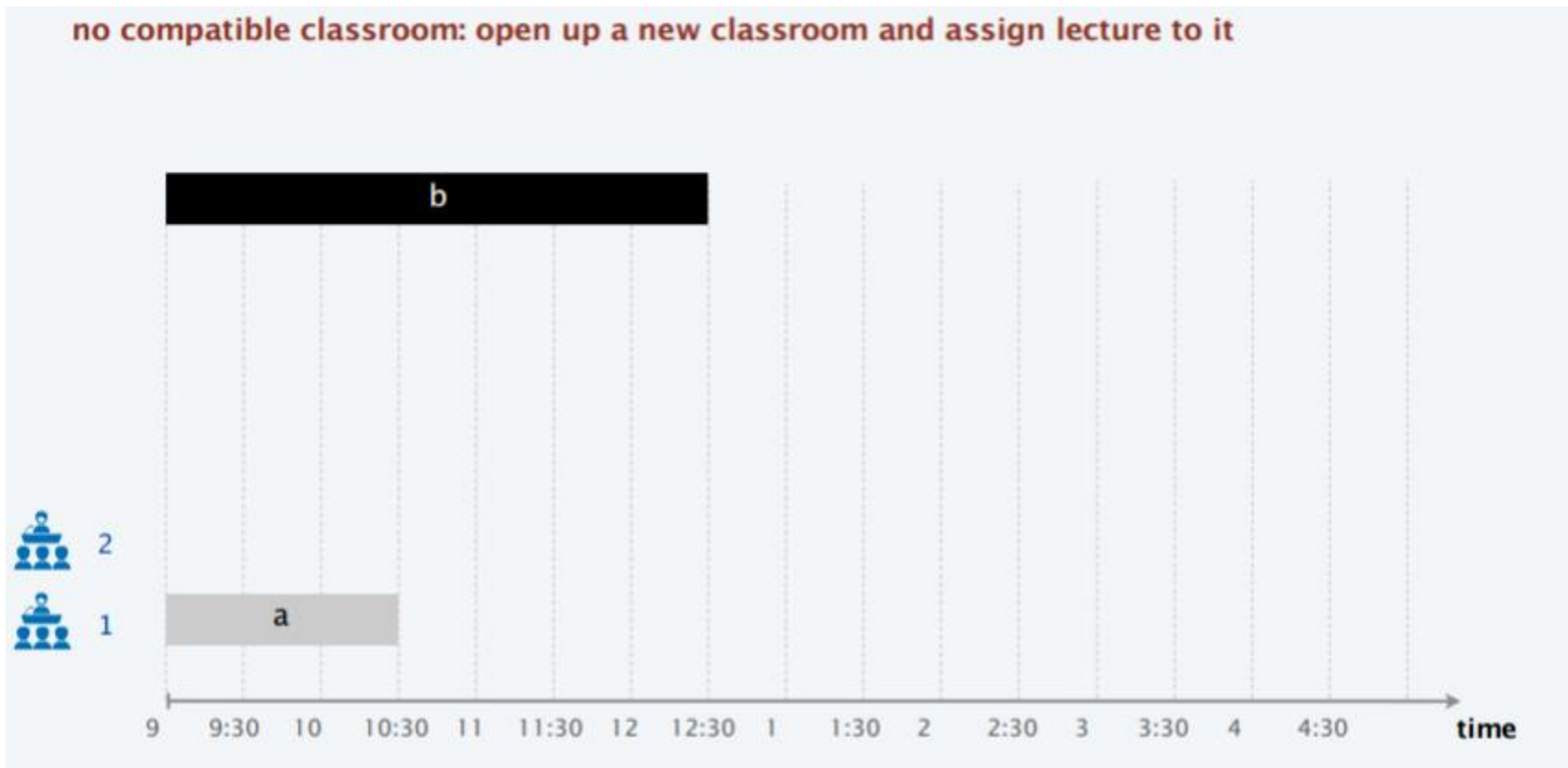


2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

Earliest-start-time-first algorithm

Consider lectures in order of start time:

- Assign next lecture to any compatible classroom.
- Otherwise, open up a new classroom.

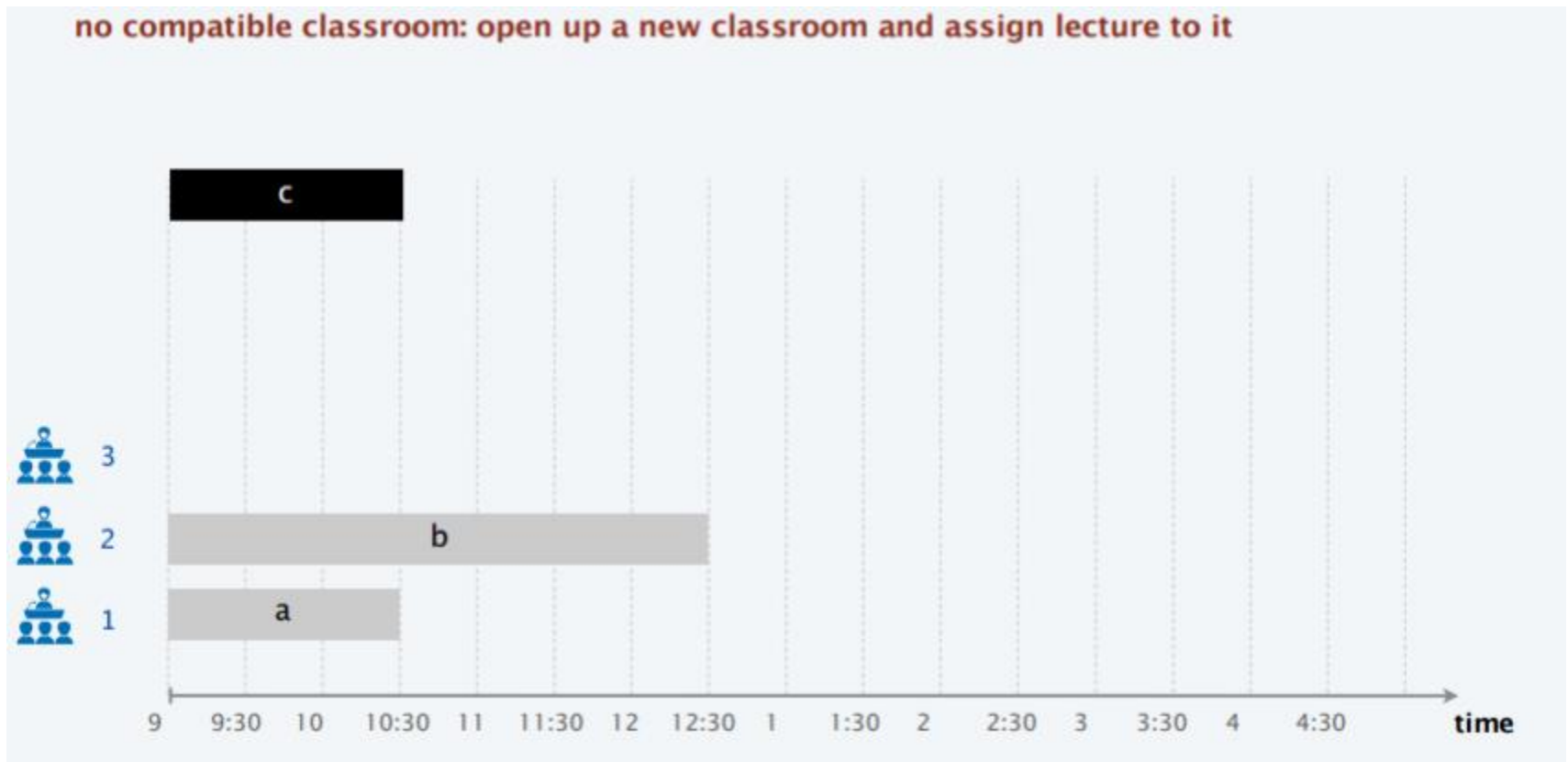


2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

Earliest-start-time-first algorithm

Consider lectures in order of start time:

- Assign next lecture to any compatible classroom.
- Otherwise, open up a new classroom.

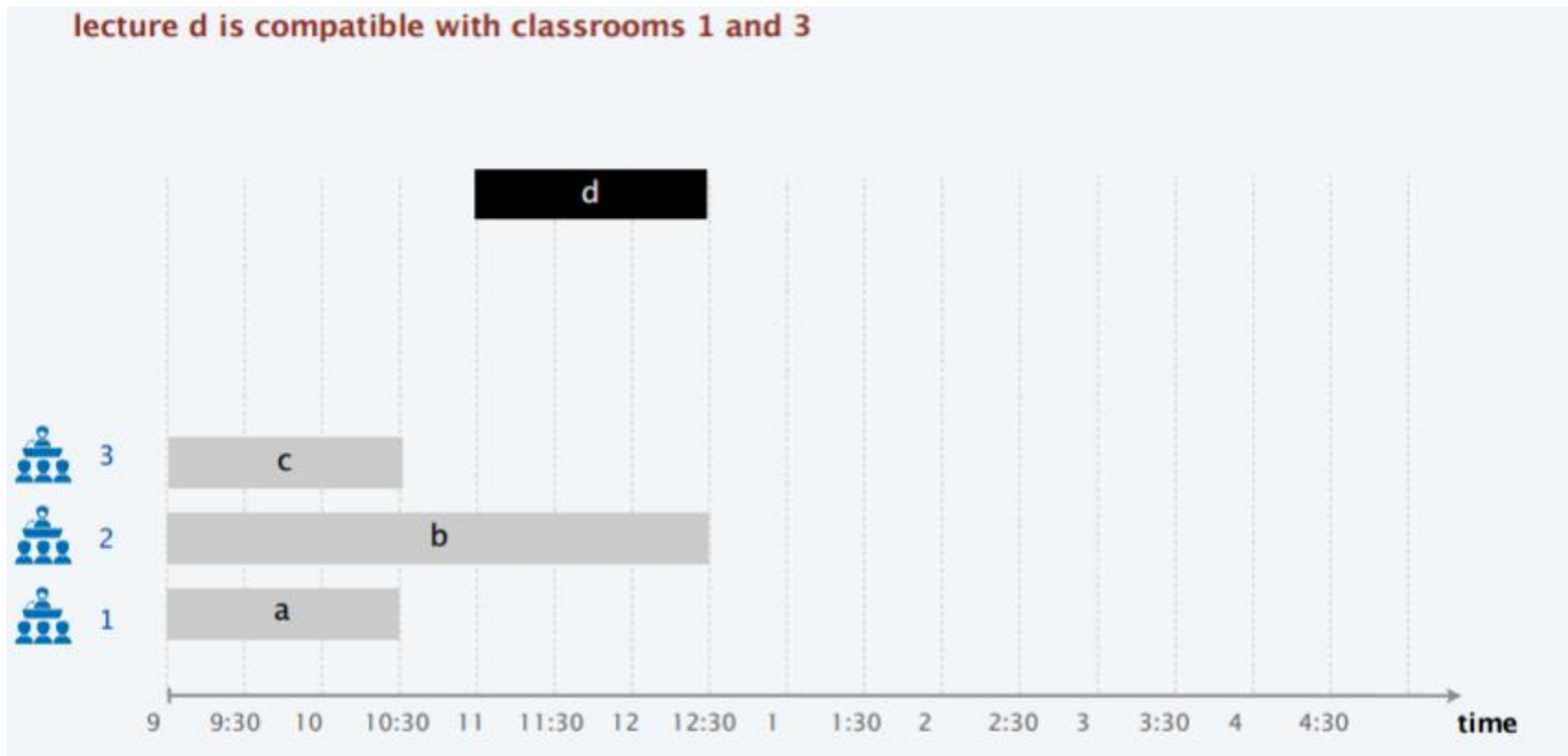


2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

Earliest-start-time-first algorithm

Consider lectures in order of start time:

- Assign next lecture to any compatible classroom.
- Otherwise, open up a new classroom.

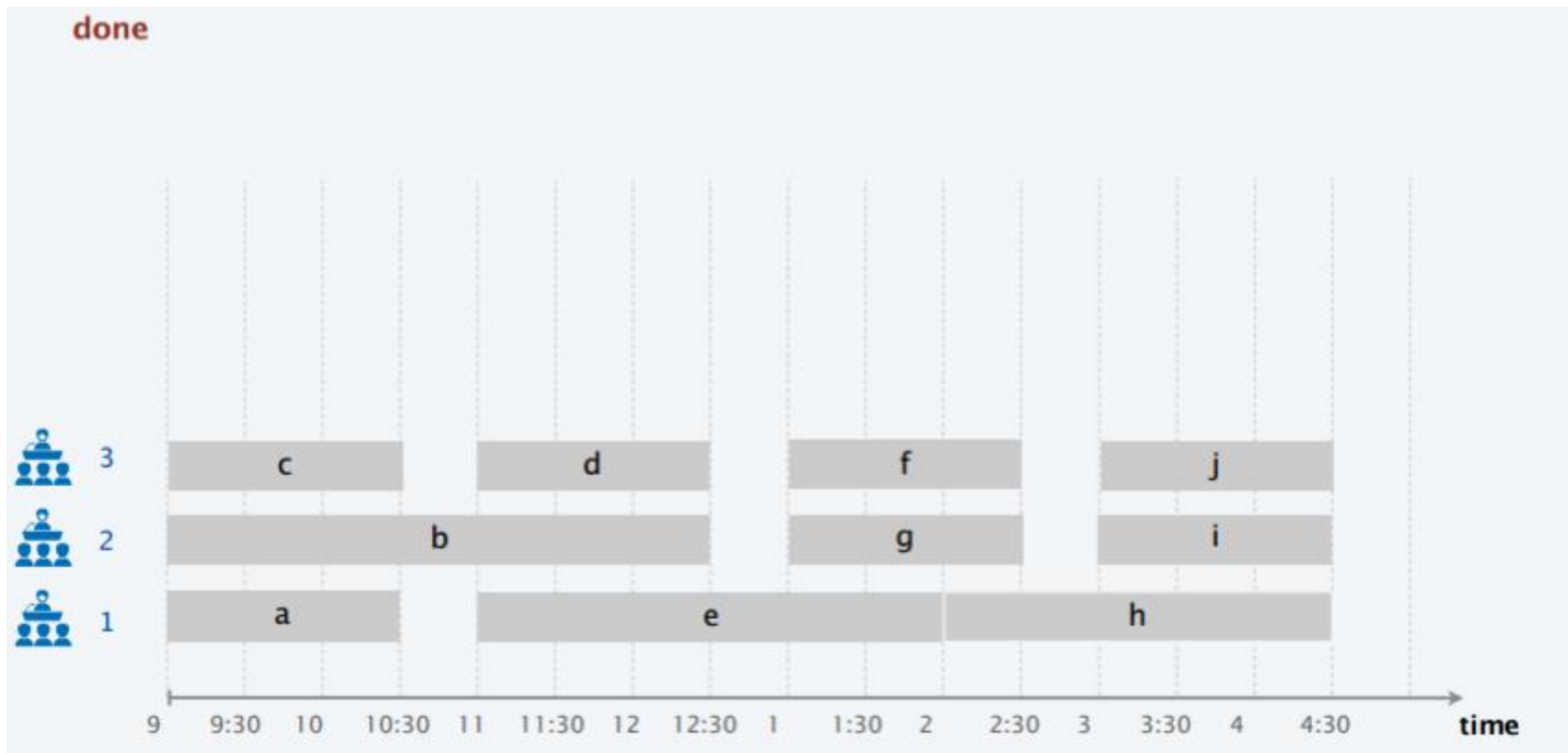


2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

Earliest-start-time-first algorithm

Consider lectures in order of start time:

- Assign next lecture to any compatible classroom.
- Otherwise, open up a new classroom.



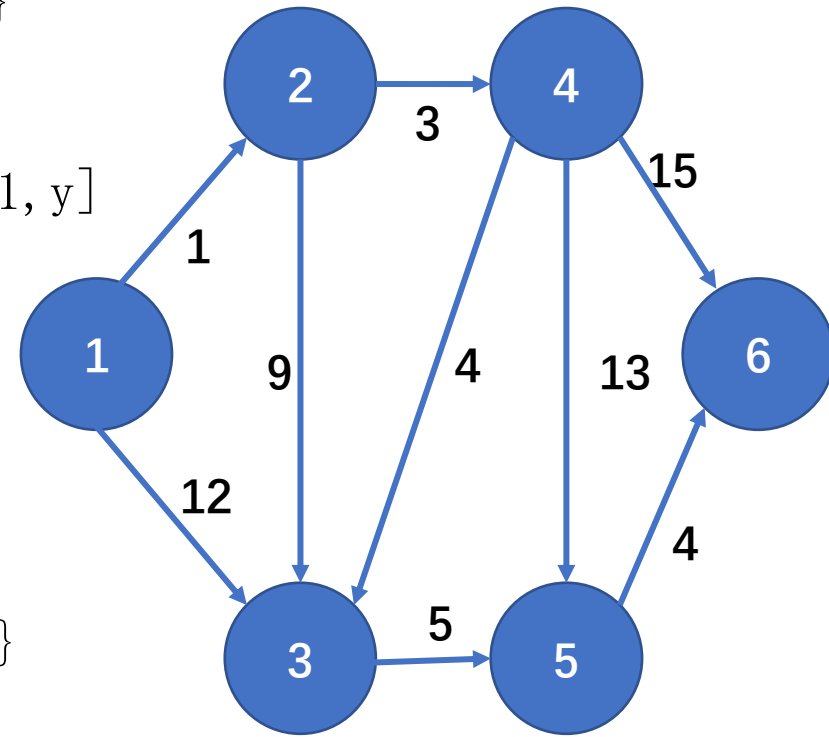
3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

Dijkstra算法

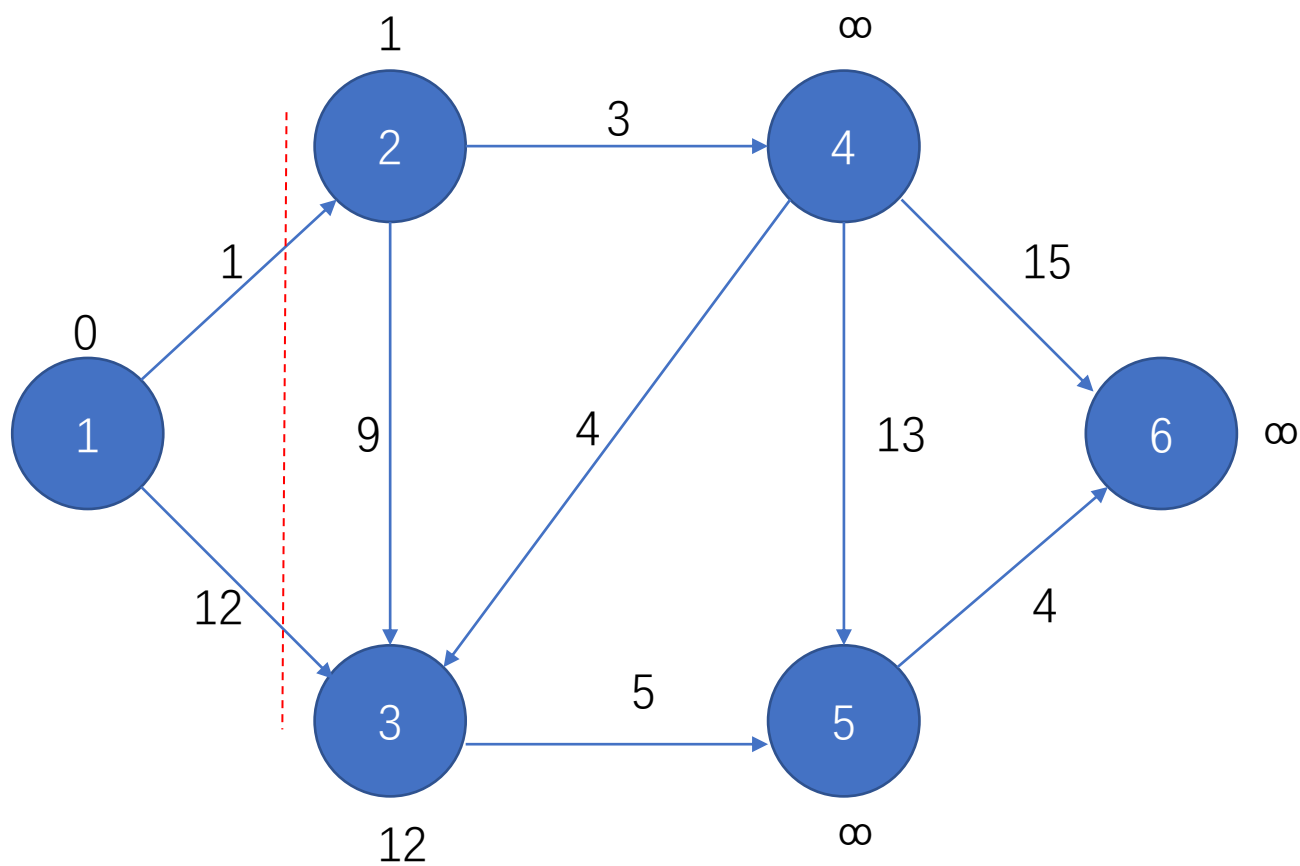
输入：含权有向图 $G=\{V, E\}$, $V=\{1, 2, \dots, n\}$

输出：G中顶点1到其他顶点的距离

```
1. for  $y \leftarrow 2$  to  $n$ :
2.   if  $y$  相邻于 1 then  $\lambda[y] \leftarrow \text{length}[1, y]$ 
3.   else  $\lambda[y] = \infty$ 
4.   end if
5. end for
6. for  $j \leftarrow 1$  to  $n-1$ 
7.   令  $y \in Y$ , 使得  $\lambda[y]$  为最小
8.    $X \leftarrow X \cup \{y\}$       {将顶点  $y$  加入  $X$ }
9.    $Y \leftarrow Y - \{y\}$     {将顶点  $y$  从  $Y$  中删除}
11.  for 每条边  $(y, w)$ 
12.    if  $w \in Y$  and  $\lambda[y] + \text{length}[y, w] < \lambda[w]$  then
13.       $\lambda[w] \leftarrow \lambda[y] + \text{length}[y, w]$ 
14.    end if
15.  end for
16. end for
```



3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

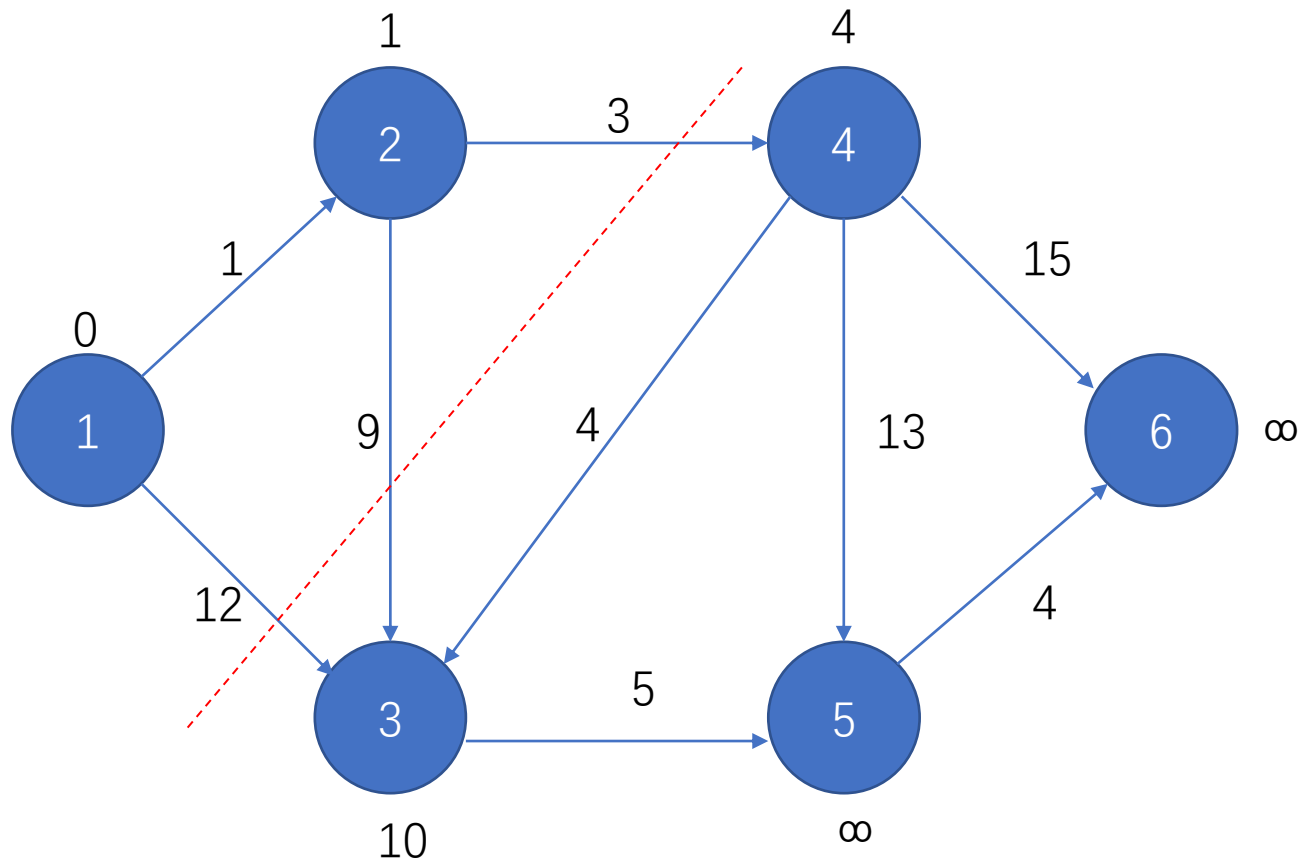


3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

加入顶点2:

$\lambda[3]=12$, $\lambda[2]+\text{length}(2,3)=10$, $10 < 12$, 更新顶点3到1的最短路径。

$\lambda[4]=\infty$, $\lambda[2]+\text{length}(2,4)=4$, $4 < \infty$, 更新顶点4到1的最短路径。



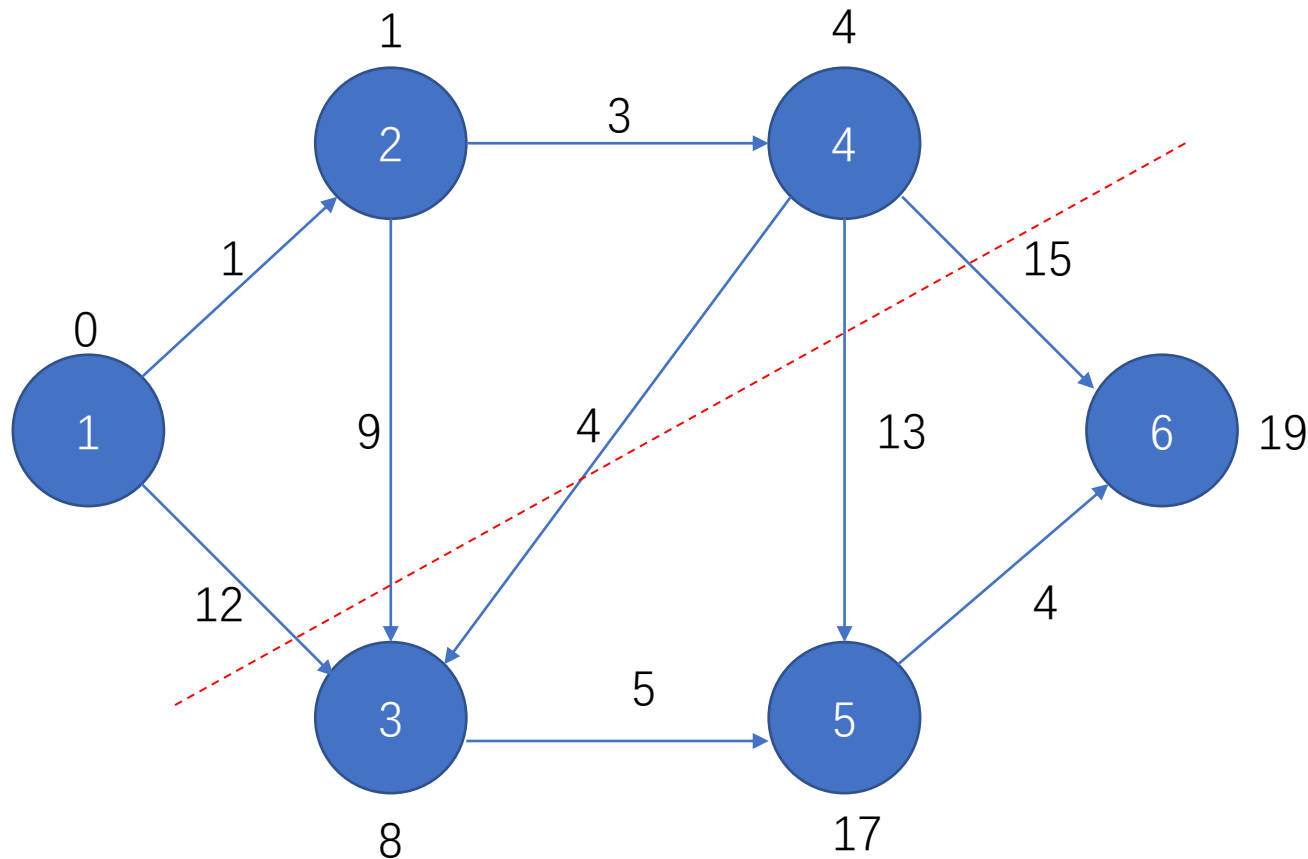
3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

加入顶点4:

$\lambda[6] = \infty$, $\lambda[4] + \text{length}(4,6) = 19$, $19 < \infty$, 更新顶点6到1的最短路径。

$\lambda[5] = \infty$, $\lambda[4] + \text{length}(4,5) = 17$, $17 < \infty$, 更新顶点5到1的最短路径。

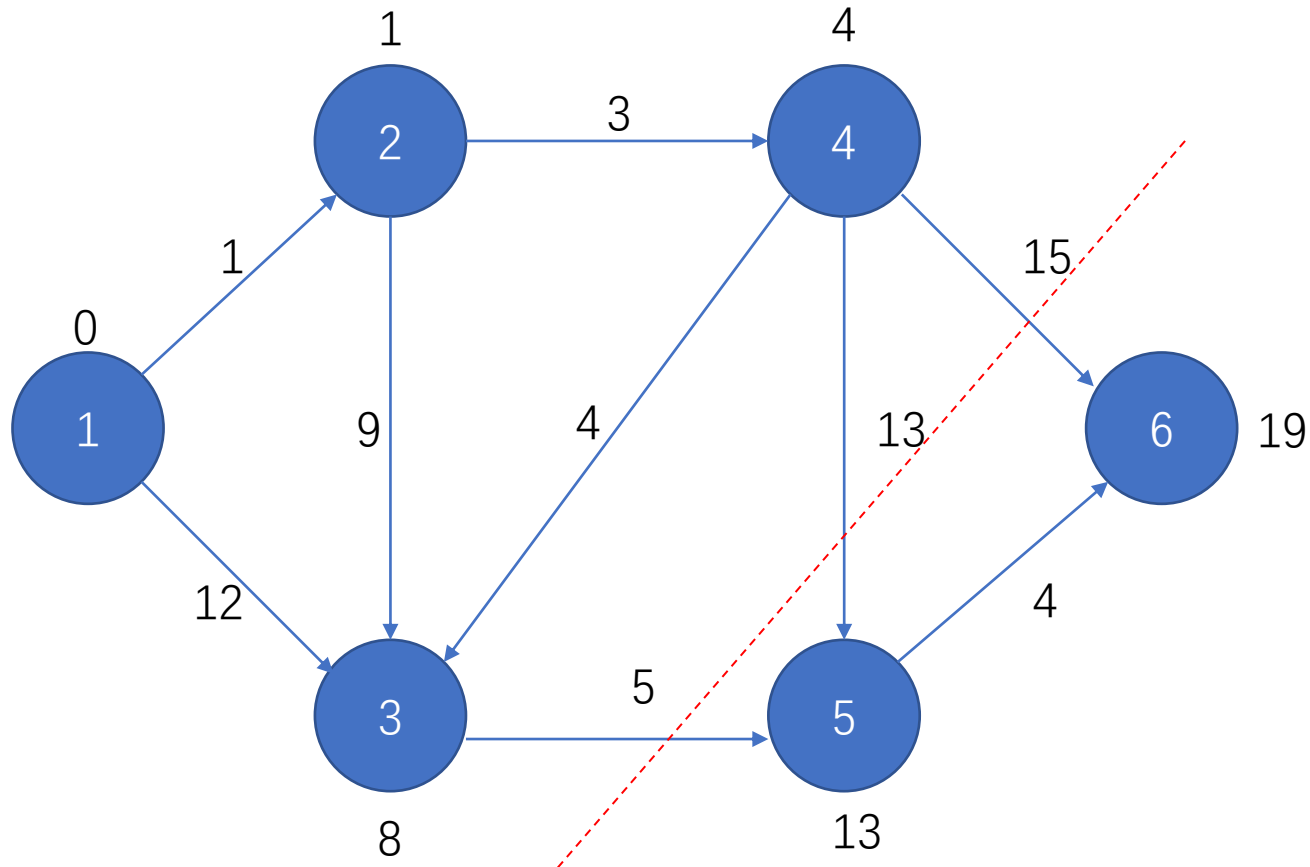
$\lambda[3] = 10$, $\lambda[4] + \text{length}(4,3) = 8$, $8 < 10$, 更新顶点3到1的最短路径。



3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

加入顶点3:

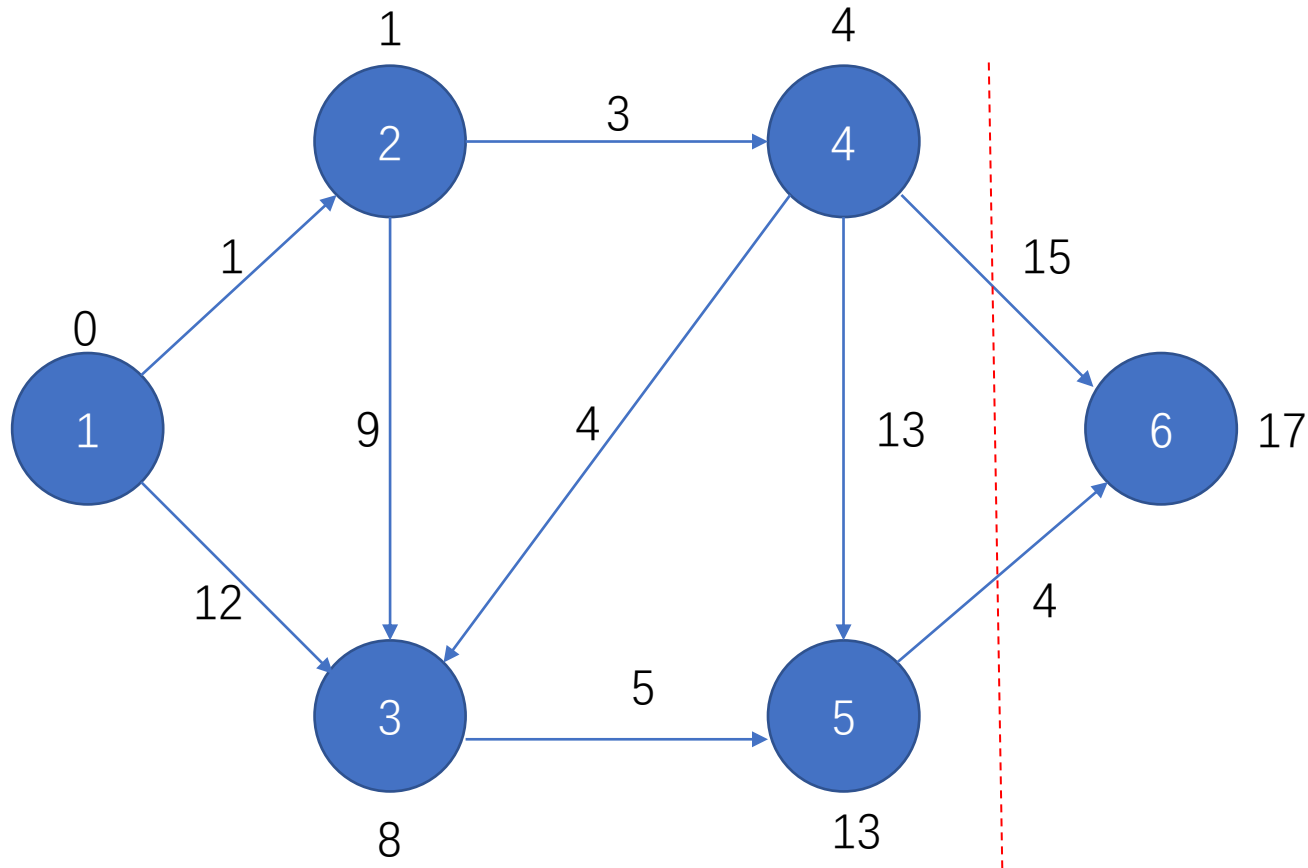
$\lambda[5] = 17$, $\lambda[3] + \text{length}(3,5) = 13$, $13 < 17$, 更新顶点5到1的最短路径。



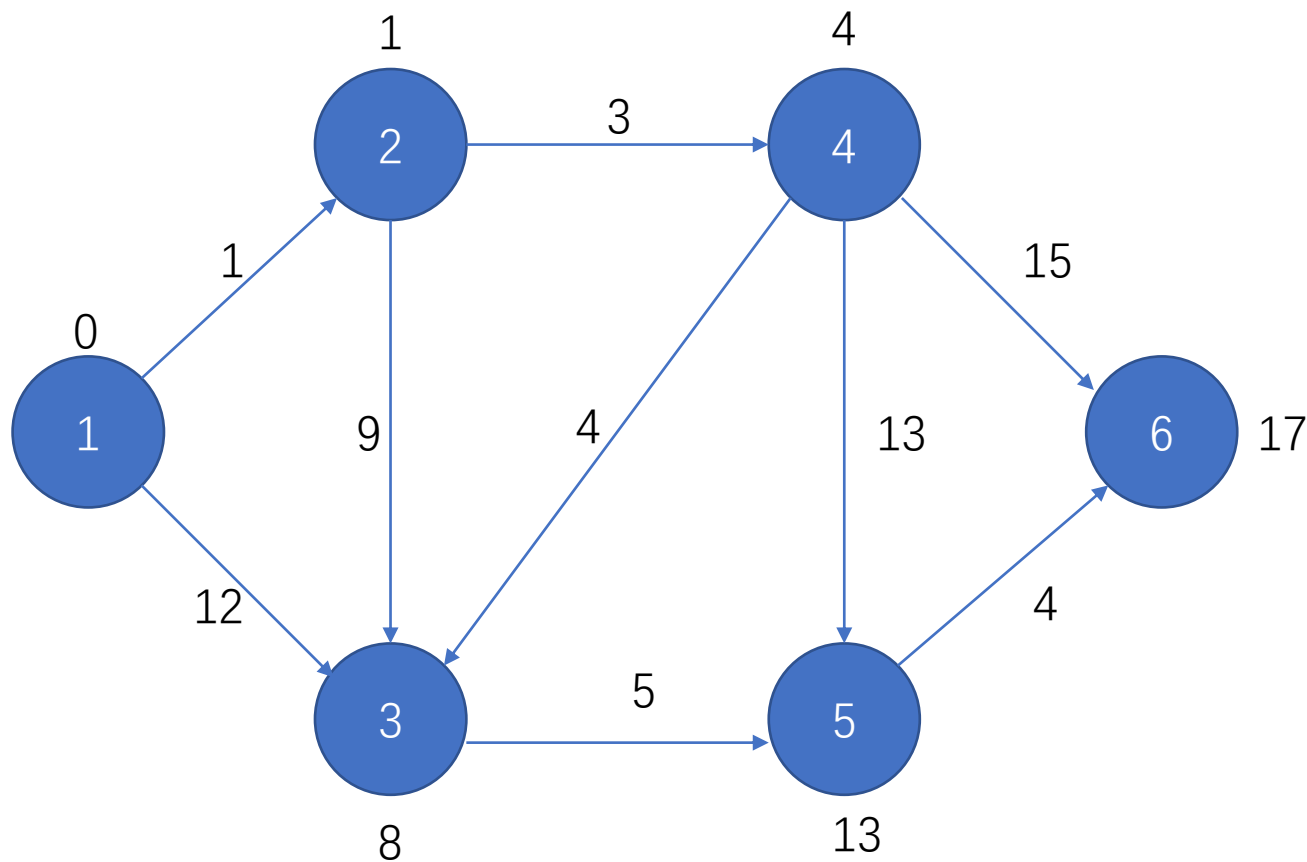
3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

加入顶点5:

$\lambda[6] = 19$, $\lambda[5] + \text{length}(5,6) = 17$, $17 < 19$, 更新顶点6到1的最短路径。



3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。



4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。

Kruskal算法

输入：包含 n 个顶点的含权连通无向图 $G = \{V, E\}$

输出：由 G 生成的最小耗费生成树 T 组成的边的集合

1. 按非降序权重将 E 中的边排序

2. $T = \{ \}$

3. while $|T| < n-1$

4. $e \leftarrow E$ 中第一个元素

5. if e 不会造成回路

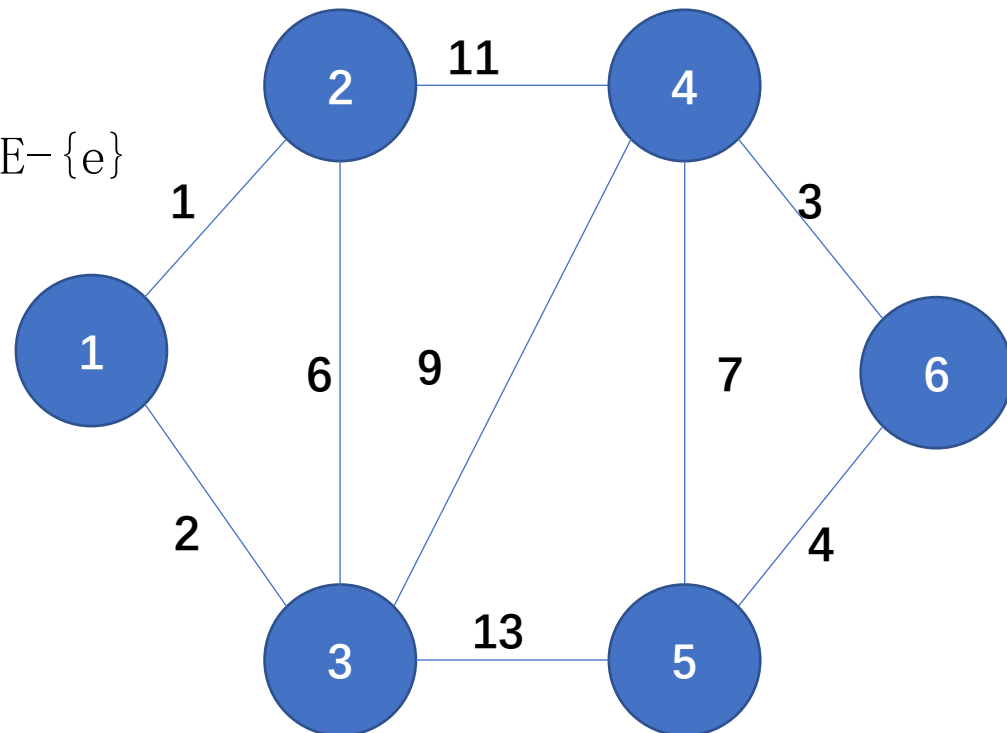
6. then $T \leftarrow T + \{e\}$, $E \leftarrow E - \{e\}$

6. end if

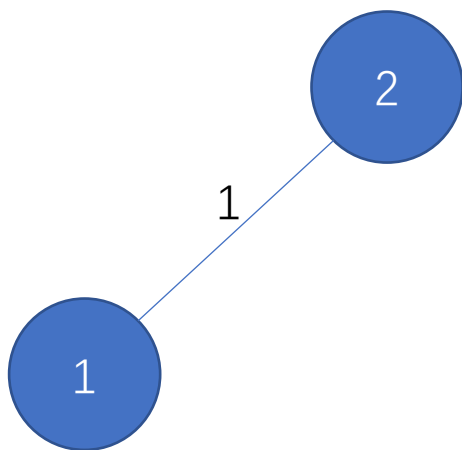
7. if $E = \{ \}$ then 算法结束

8. end if

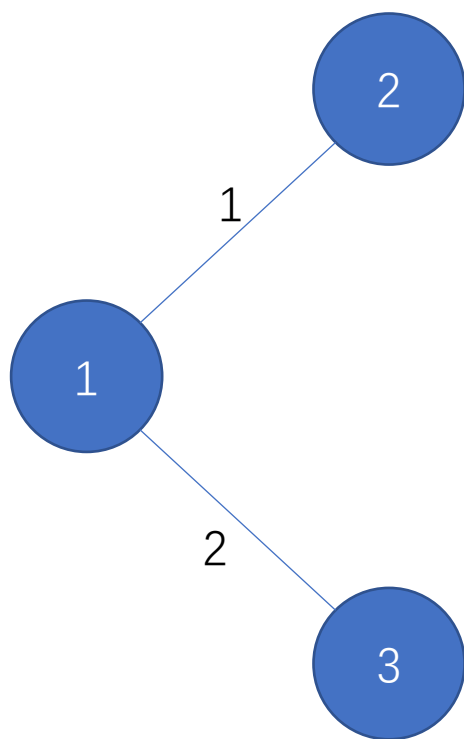
9. end while



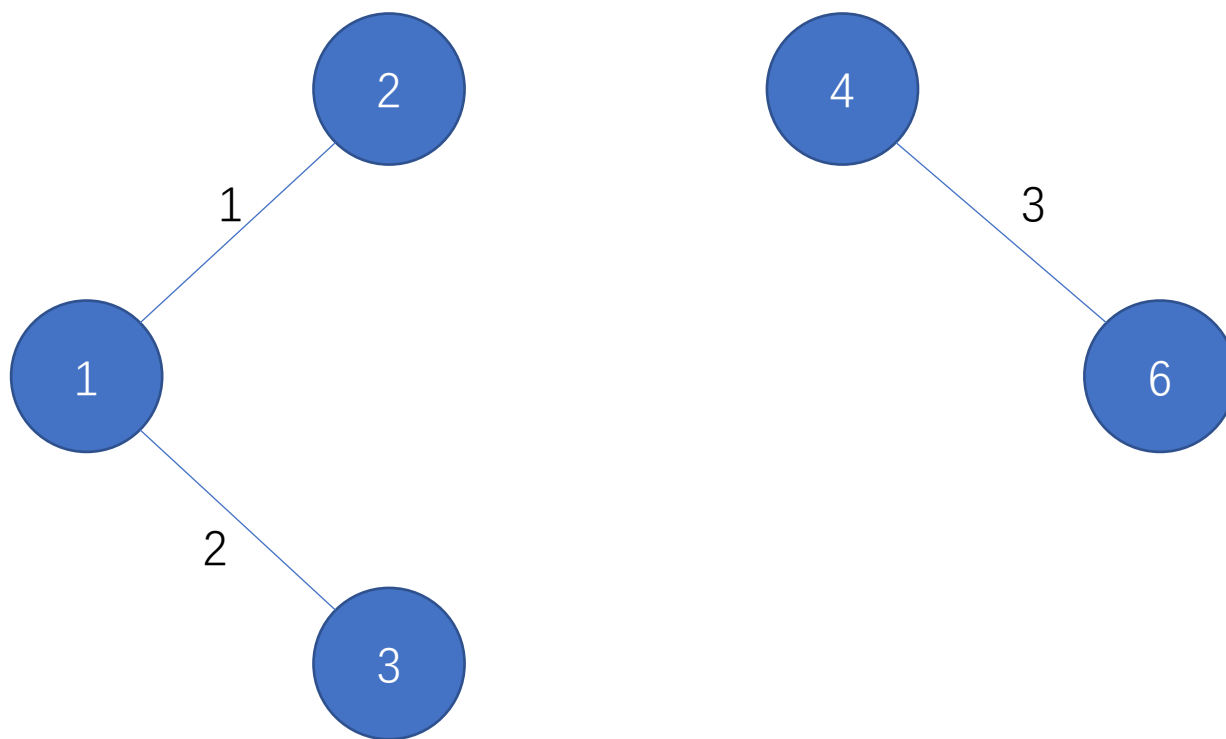
4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。



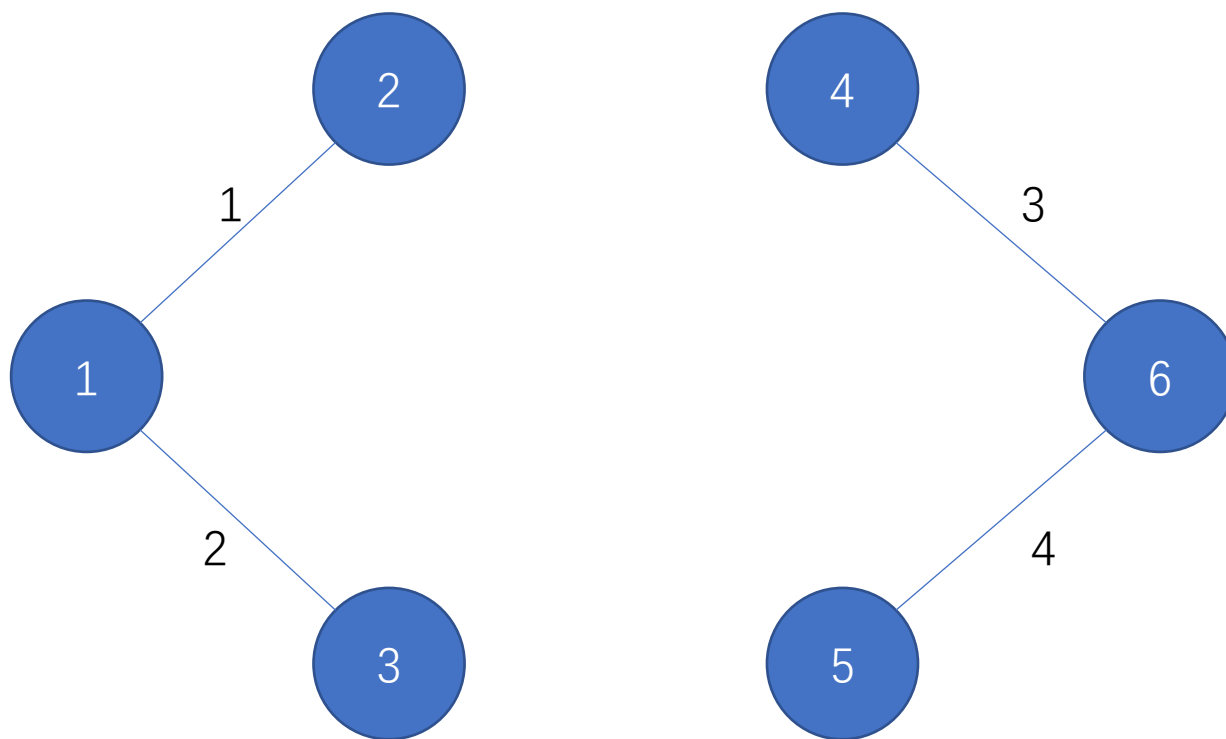
4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。



4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。

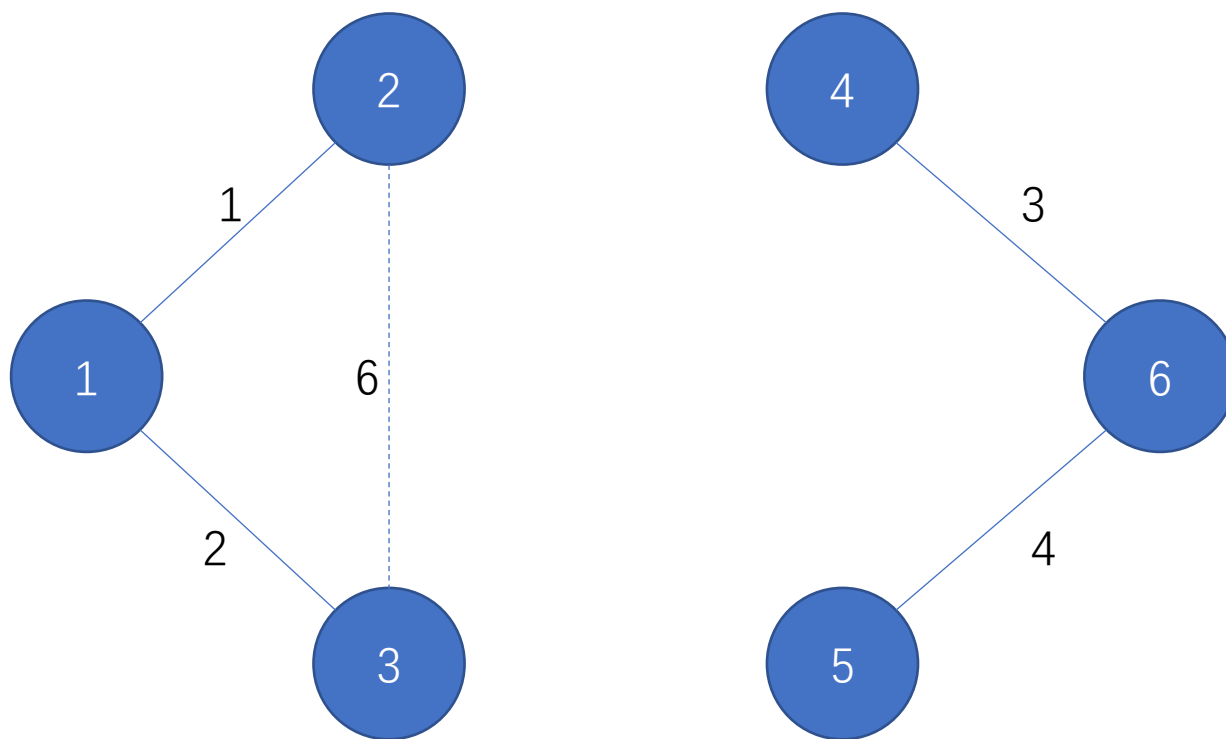


4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。



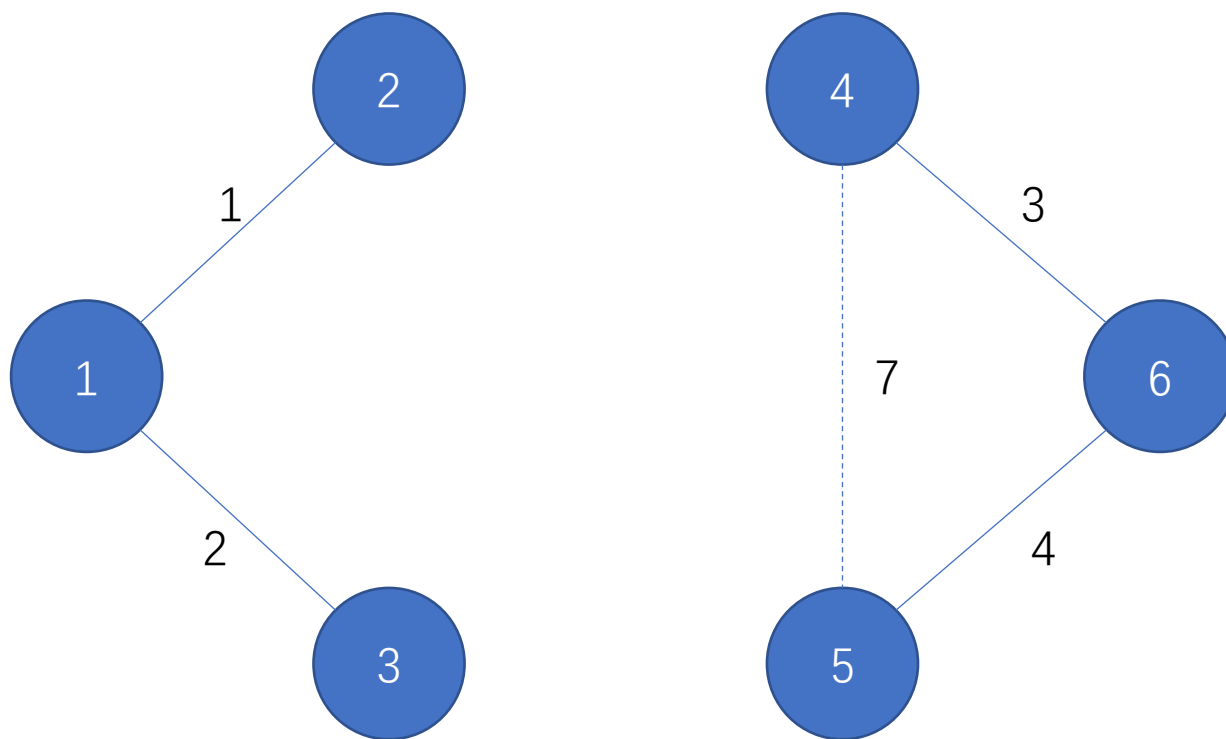
4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。

由于边(2,3)构成回路，舍弃

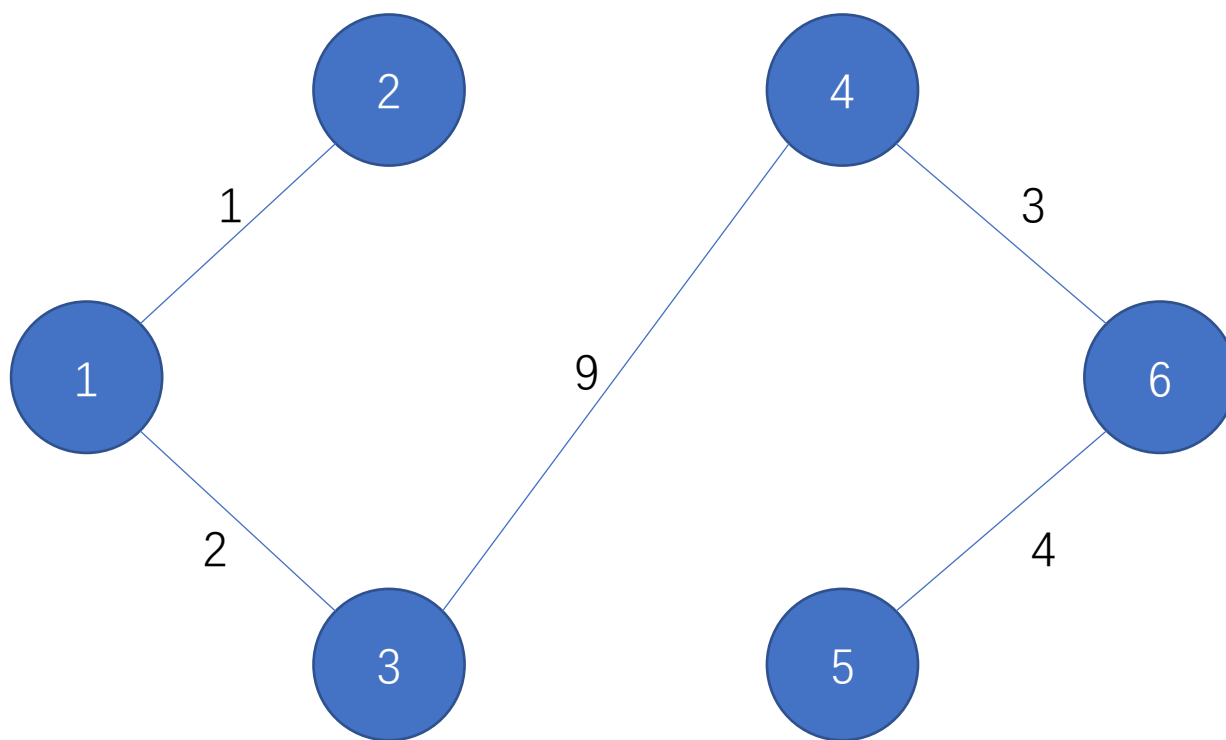


4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。

由于边(4,5)构成回路，舍弃



4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。



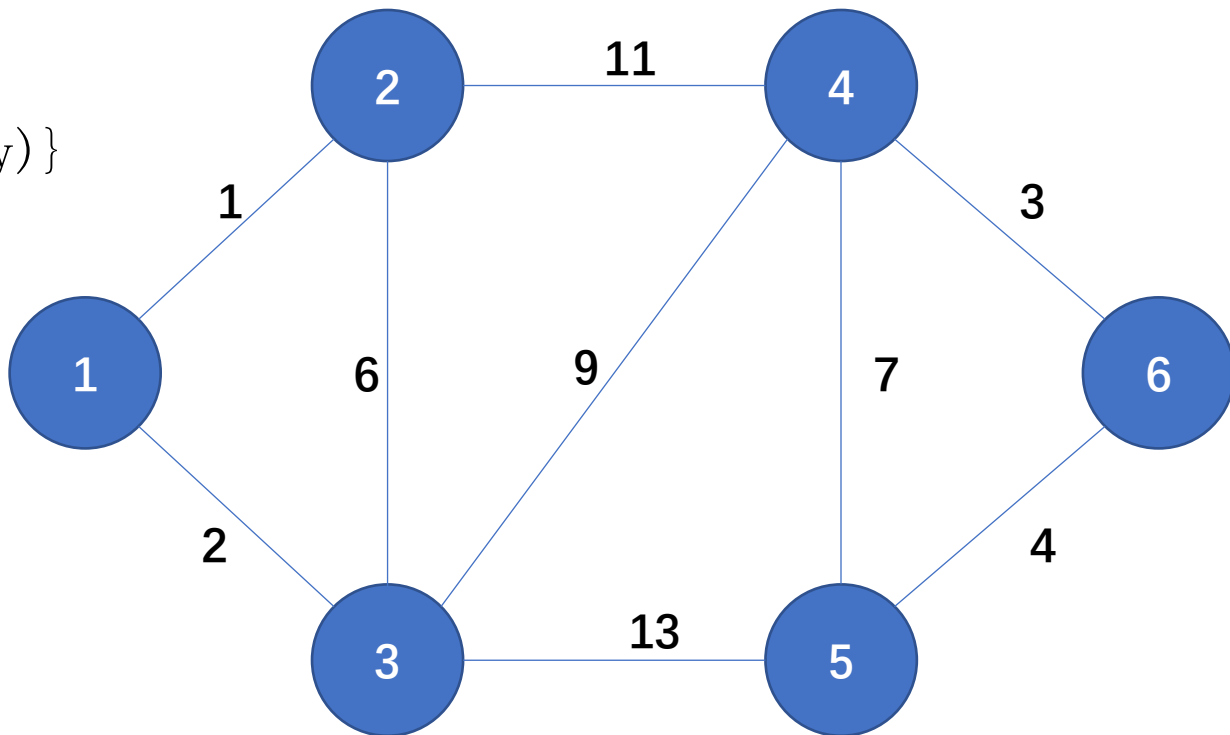
5. 已知有权无向图如下，根据Prim算法画出最小生成树。

Prim算法

输入：包含 n 个顶点的含权连通无向图 $G=\{V, E\}$

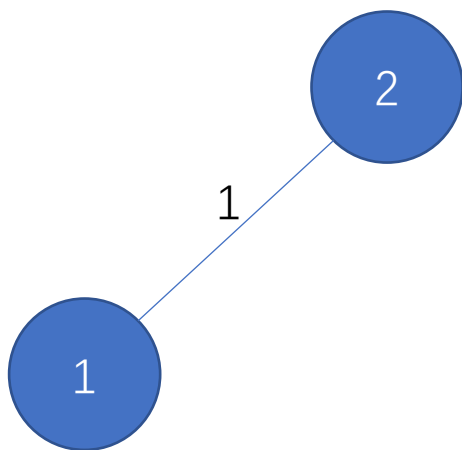
输出：由 G 生成的最小耗费生成树 T 组成的边的集合

1. $T=\{\}$, $X\leftarrow\{1\}$, $Y\leftarrow V-\{1\}$
2. while $Y\neq\{\}$
3. 设 (x, y) 是最小权重的边，其中 $x\in X$, $y\in Y$
4. $X\leftarrow X\cup\{y\}$
5. $Y\leftarrow Y-\{y\}$
6. $T\leftarrow T\cup\{(x, y)\}$
7. end while



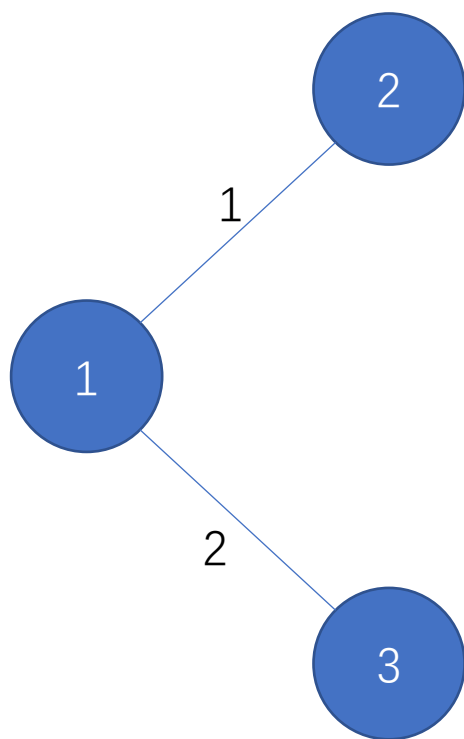
5. 已知有权无向图如下，根据Prim算法画出最小生成树。

$$X = \{1, 2\}, Y = \{3, 4, 5, 6\}$$



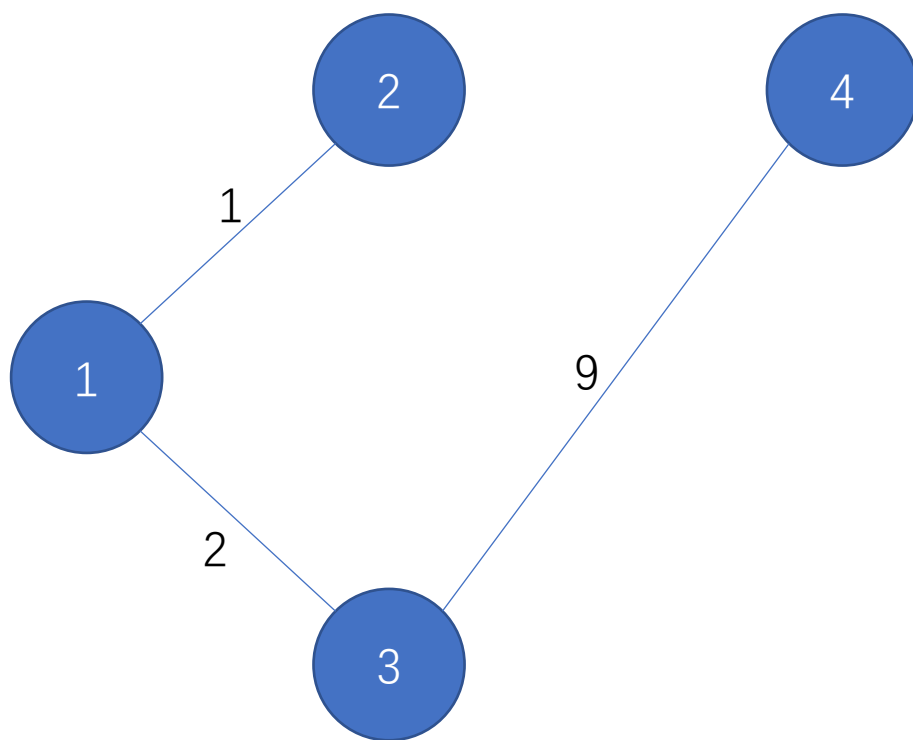
5. 已知有权无向图如下，根据Prim算法画出最小生成树。

$$X = \{1, 2, 3\} \quad Y = \{4, 5, 6\}$$



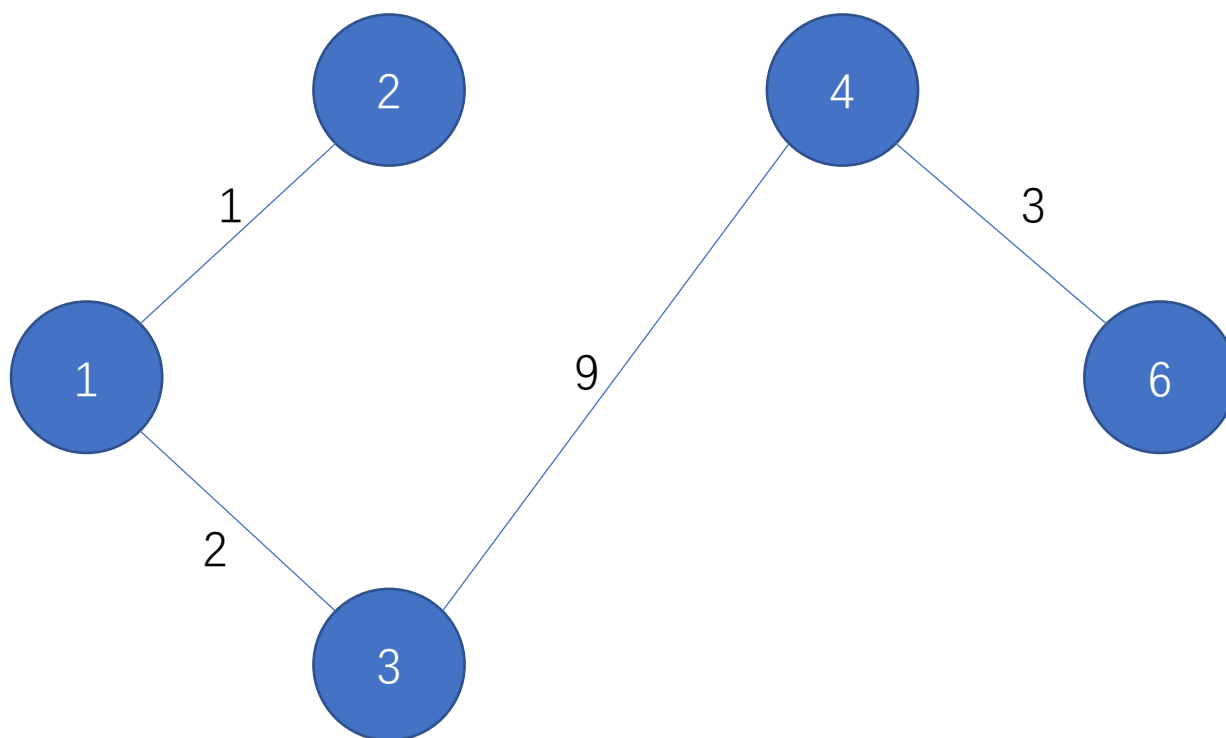
5. 已知有权无向图如下，根据Prim算法画出最小生成树。

$$X = \{1, 2, 3, 4\} \quad Y = \{5, 6\}$$



5. 已知有权无向图如下，根据Prim算法画出最小生成树。

$$X = \{1, 2, 3, 4, 6\} \quad Y = \{5\}$$



5. 已知有权无向图如下，根据Prim算法画出最小生成树。

$$X = \{1, 2, 3, 4, 5, 6\} \quad Y = \{\}$$

