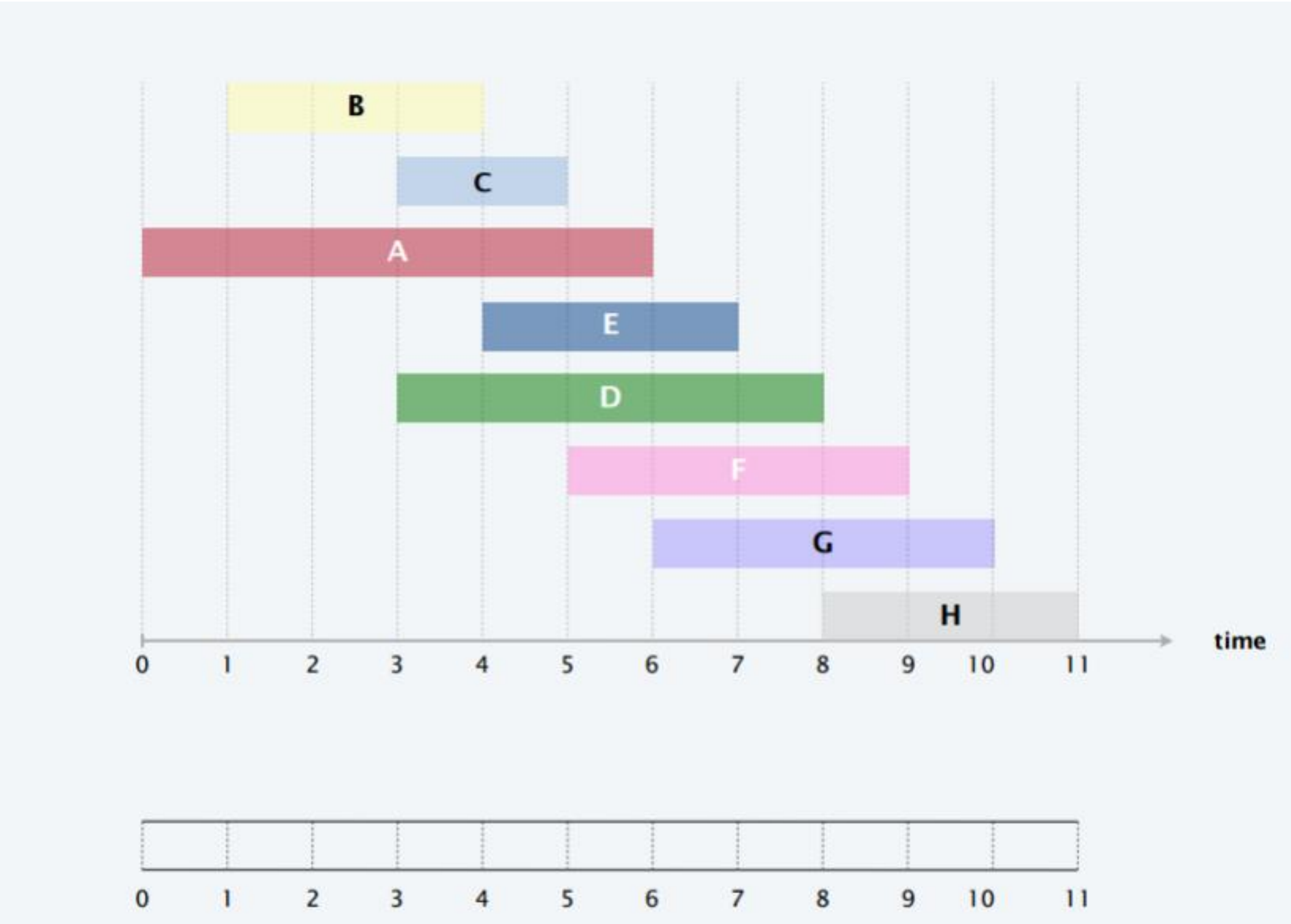
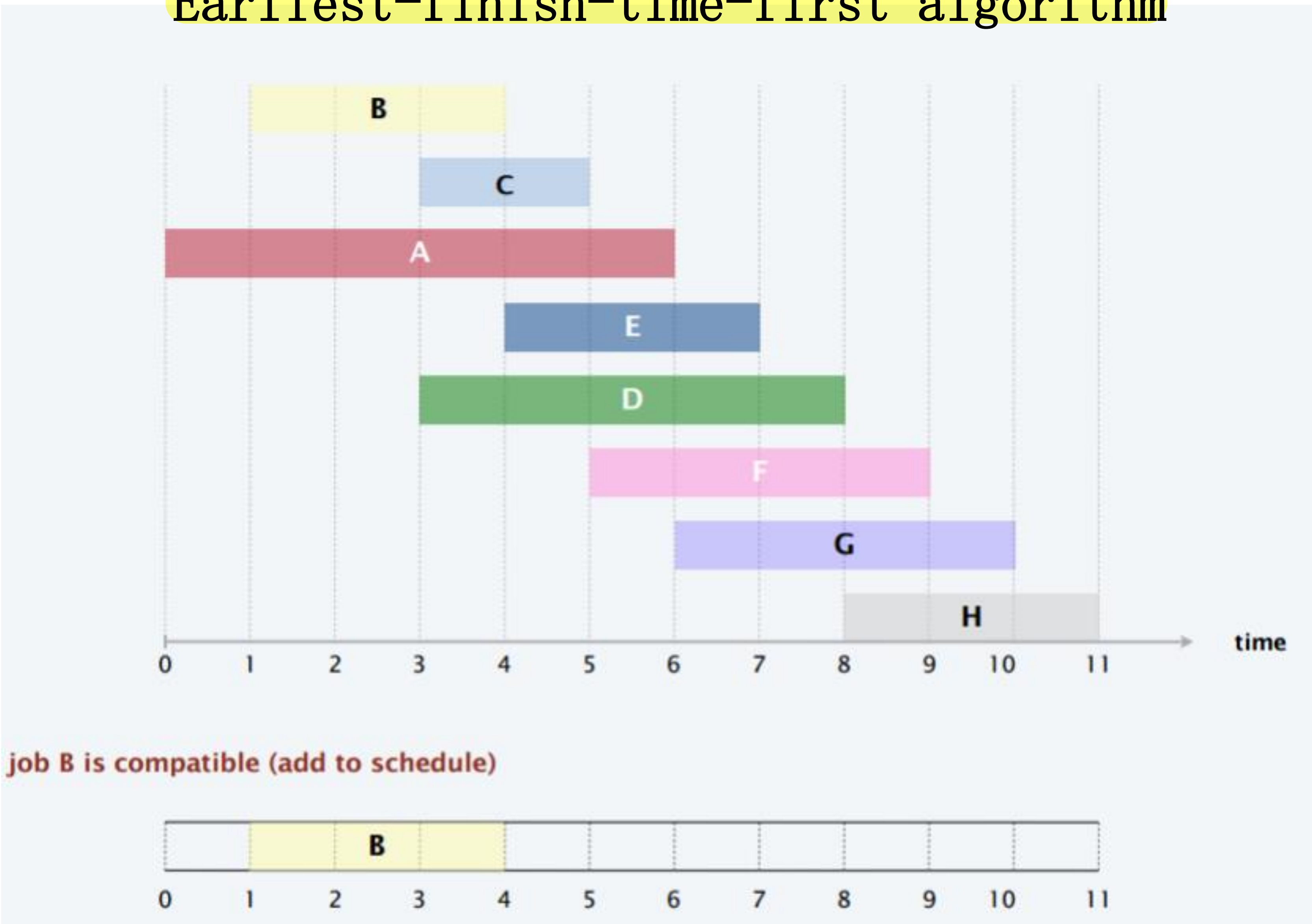
1. 给定以下多个任务及相应的开始和终止时间，求任务的最大组合数。



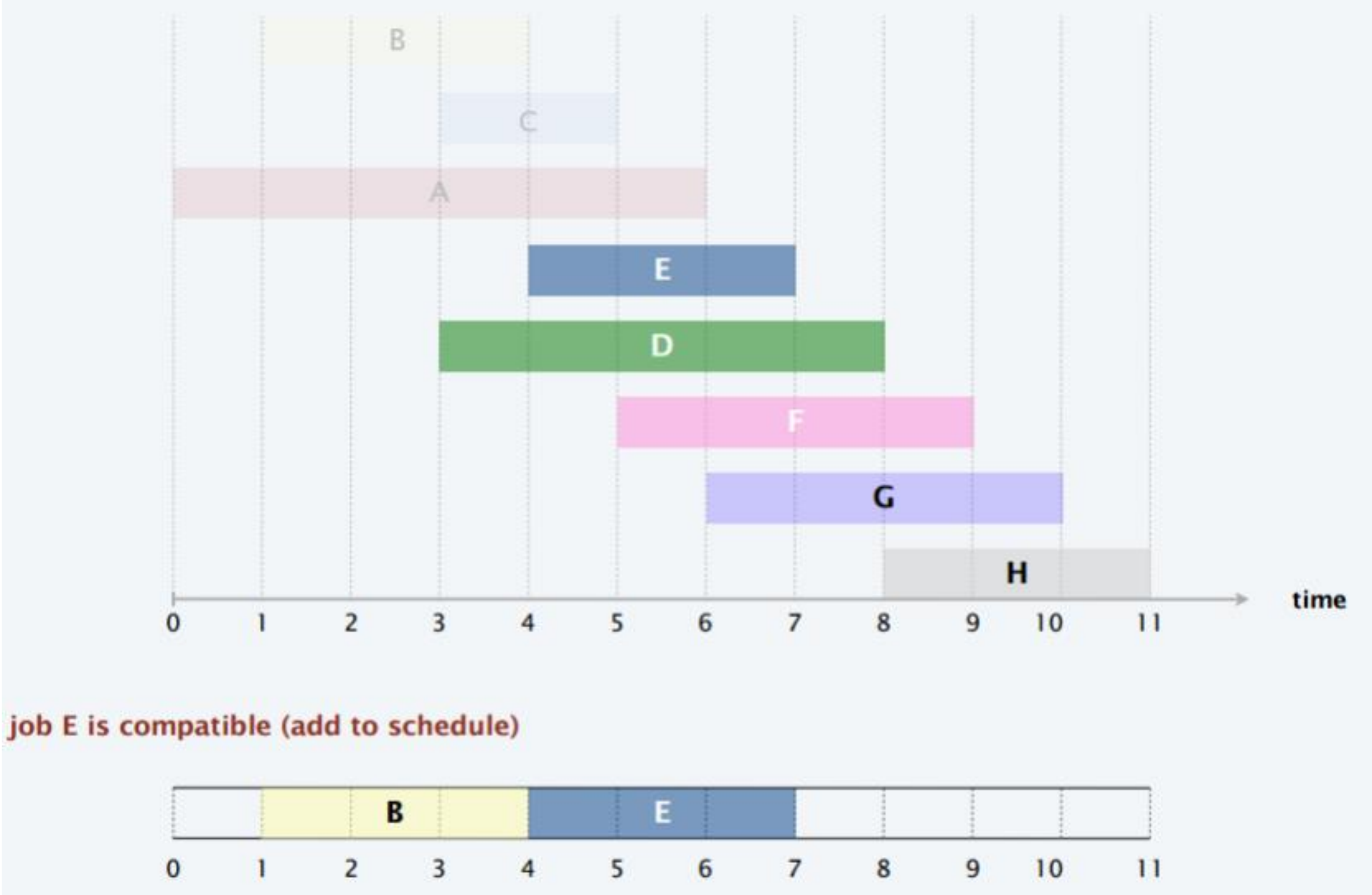
1. 给定以下多个任务及相应的开始和终止时间，求任务的最大组合数。

Earliest-finish-time-first algorithm



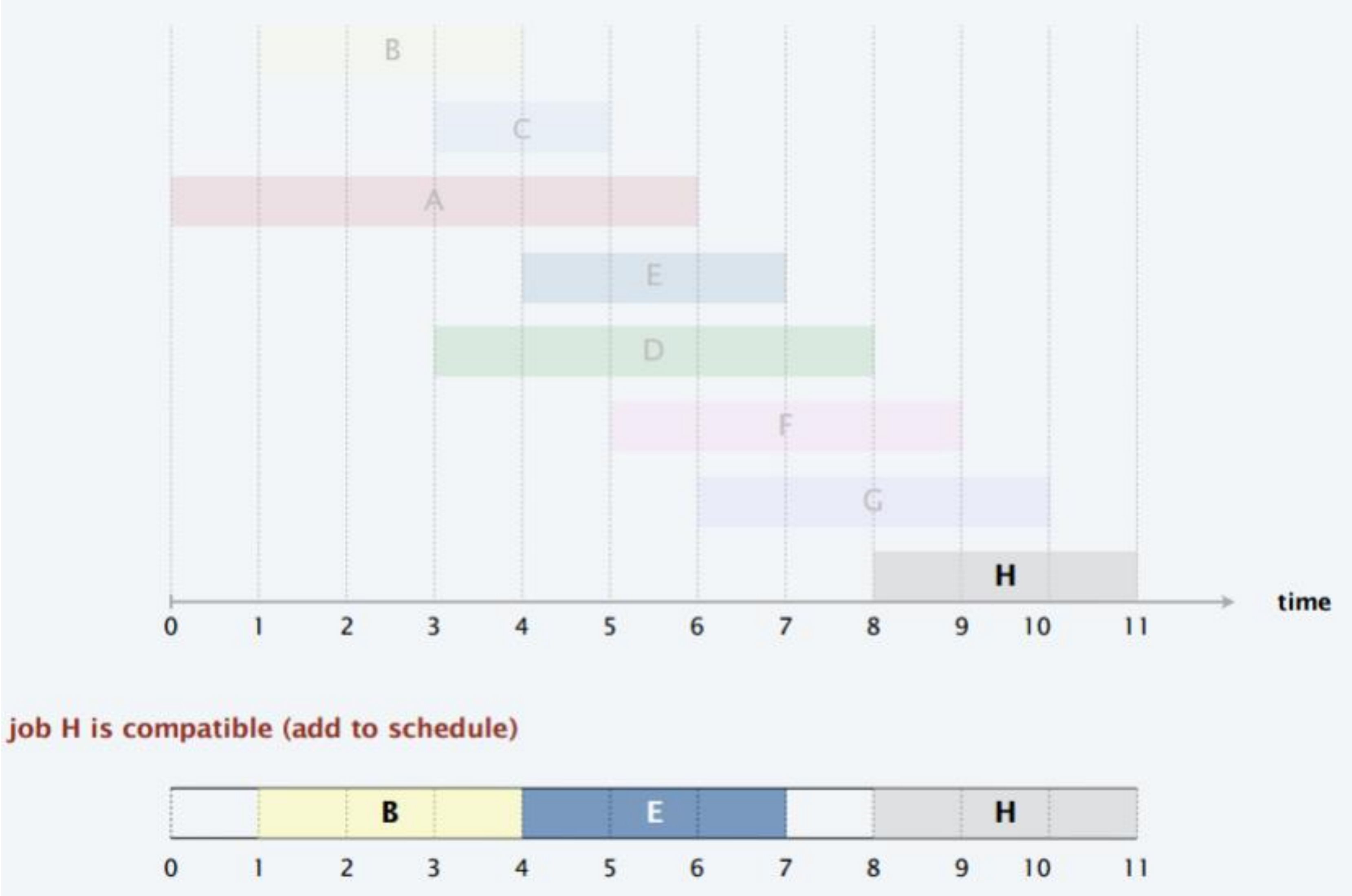
1. 给定以下多个任务及相应的开始和终止时间，求任务的最大组合数。

Earliest-finish-time-first algorithm

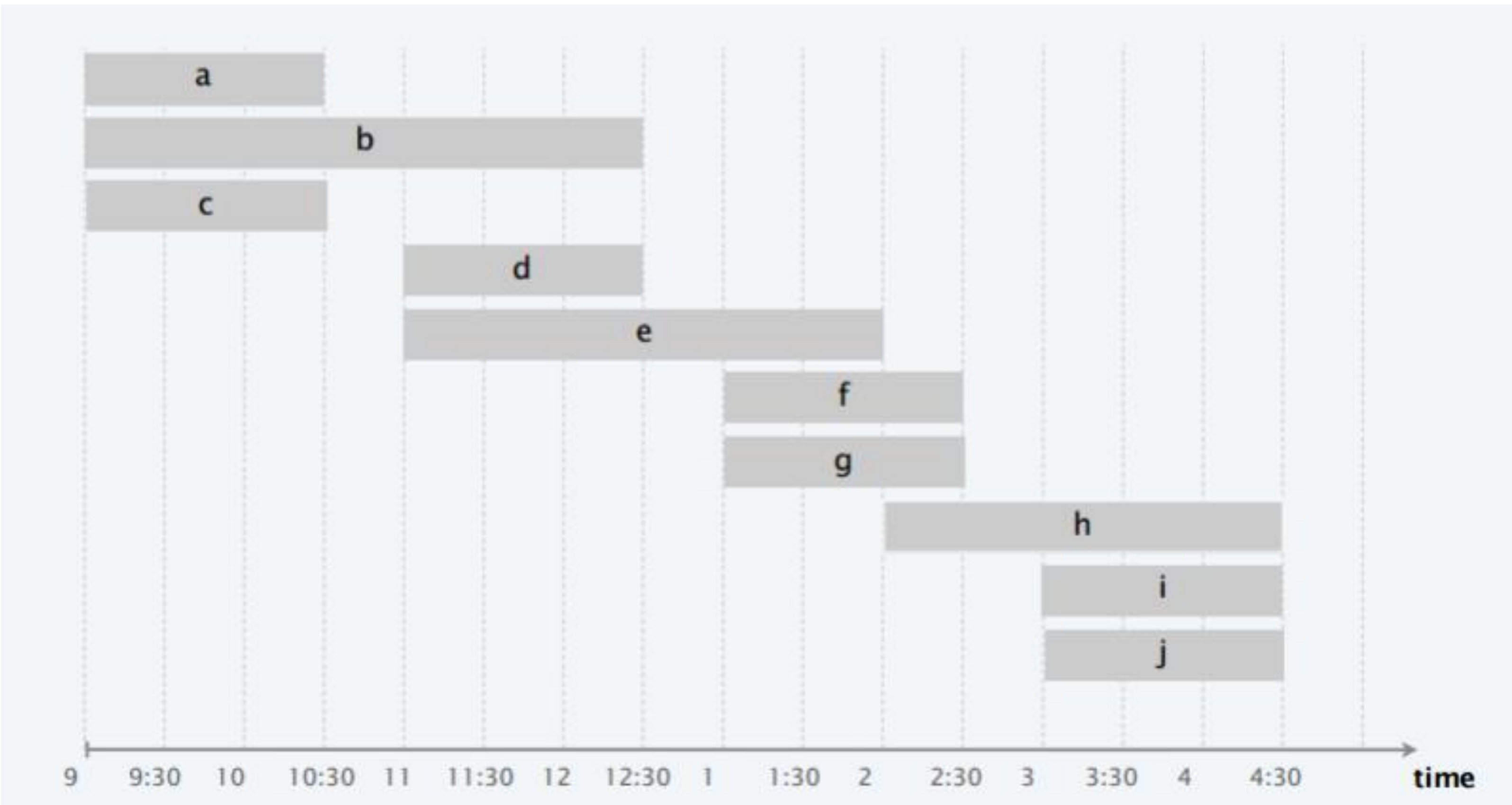


1. 给定以下多个任务及相应的开始和终止时间，求任务的最大组合数。

Earliest-finish-time-first algorithm



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。

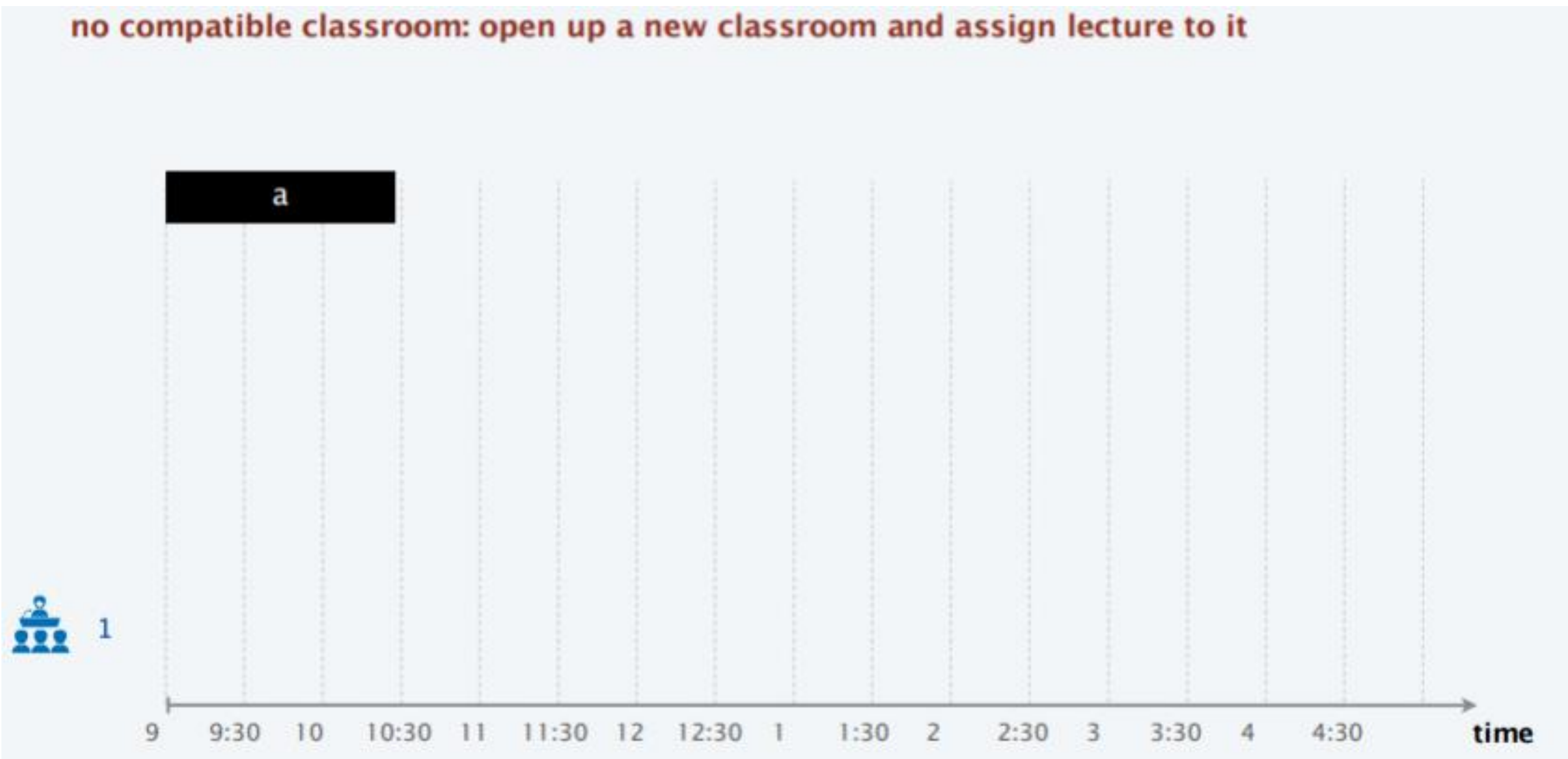


2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

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.

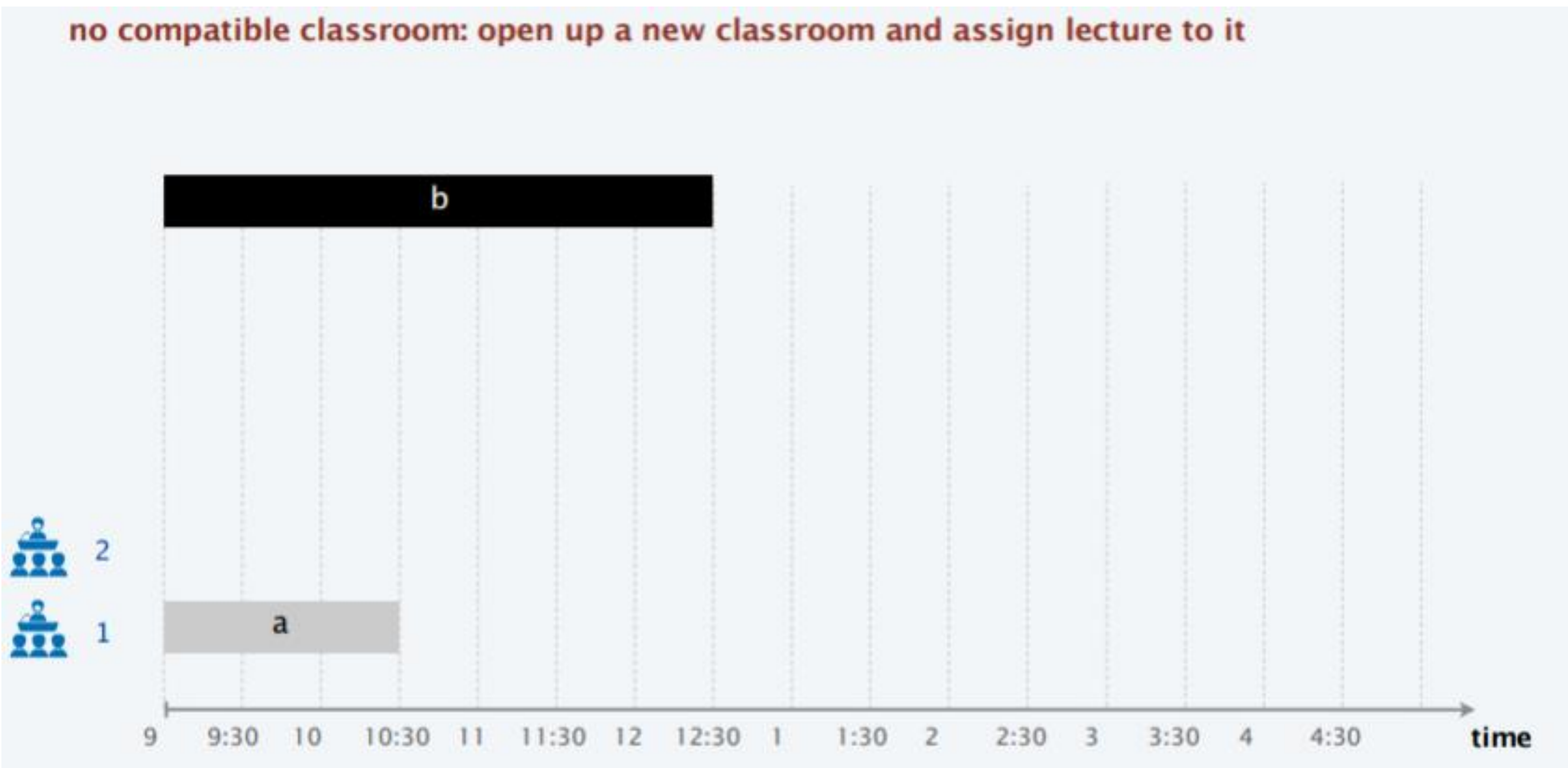


2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

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.

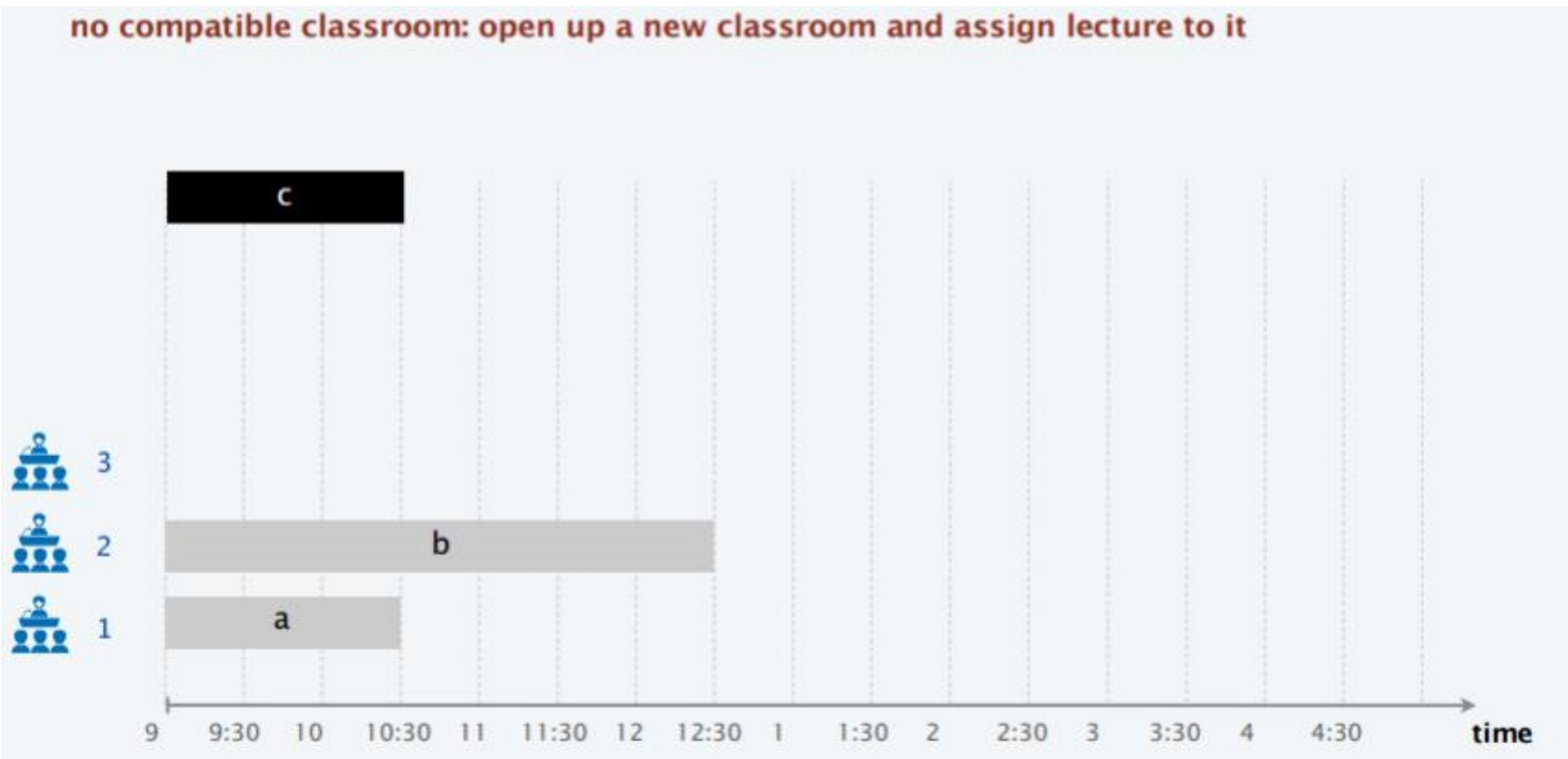


2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

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.

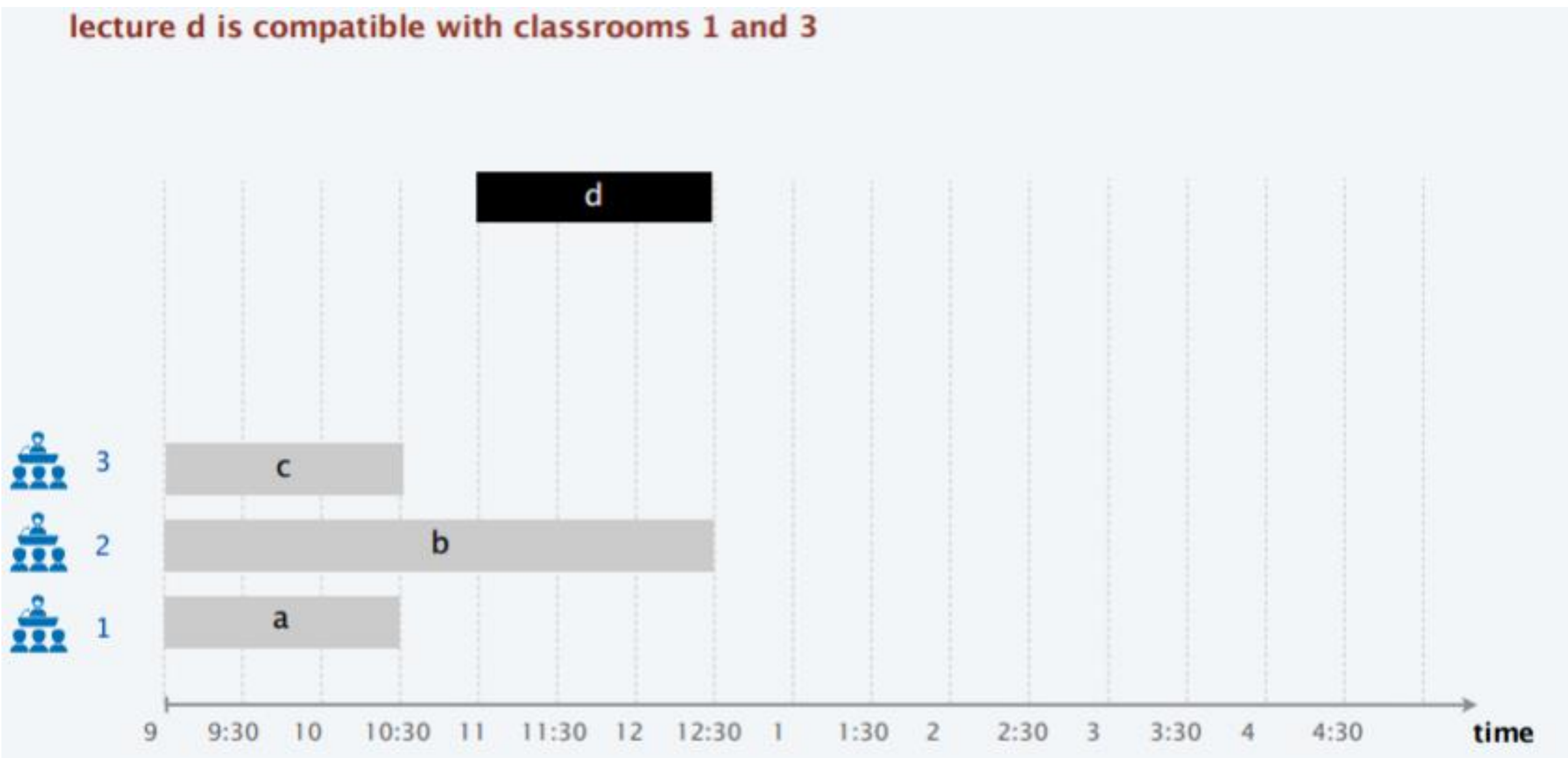


2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

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.

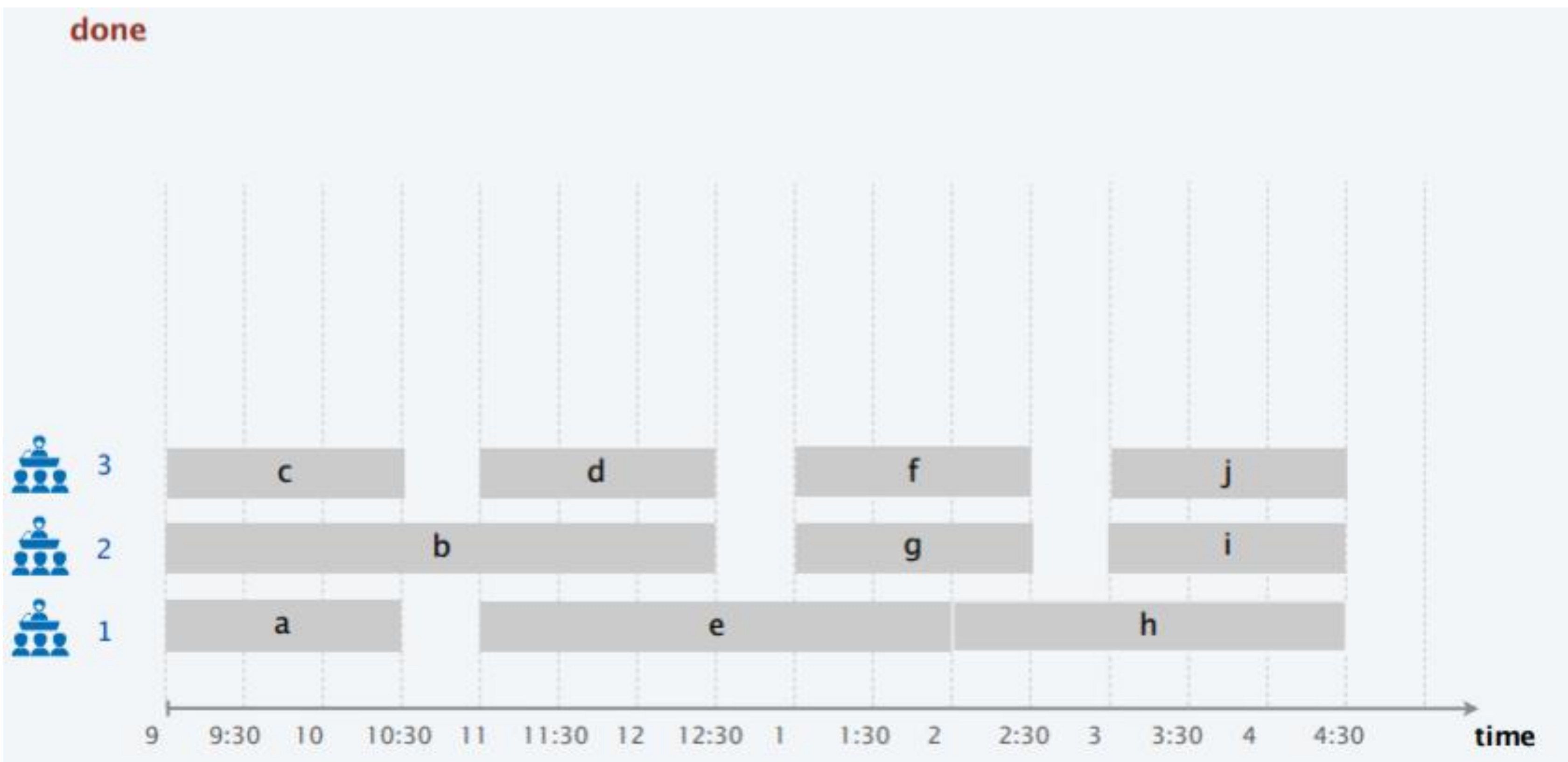


2. 给定以下讲座及相应的开始和终止时间，确定最少教室数。

Earliest-start-time-first algorithm

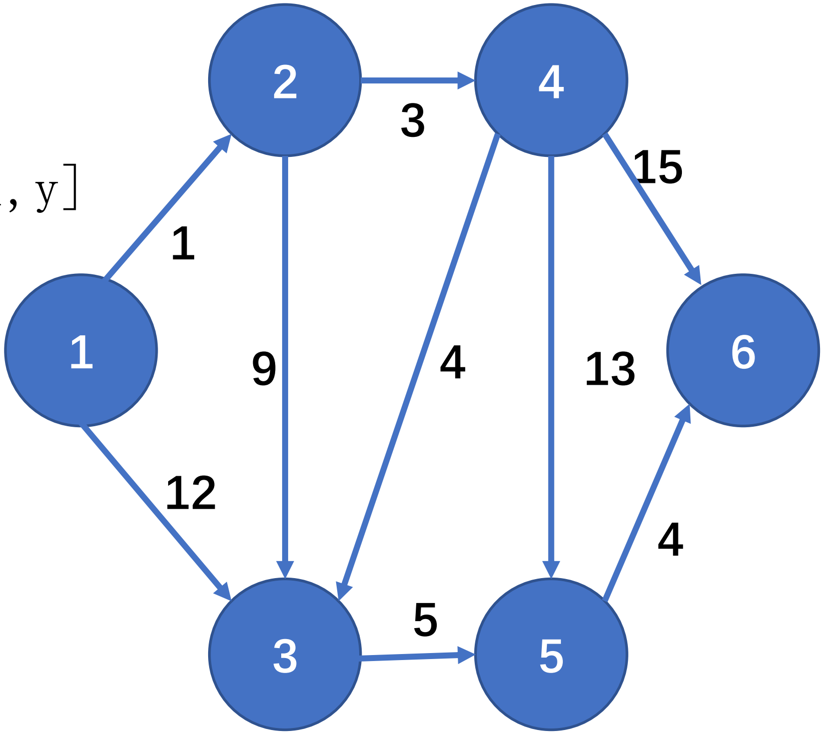
Consider lectures in order of start time:

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

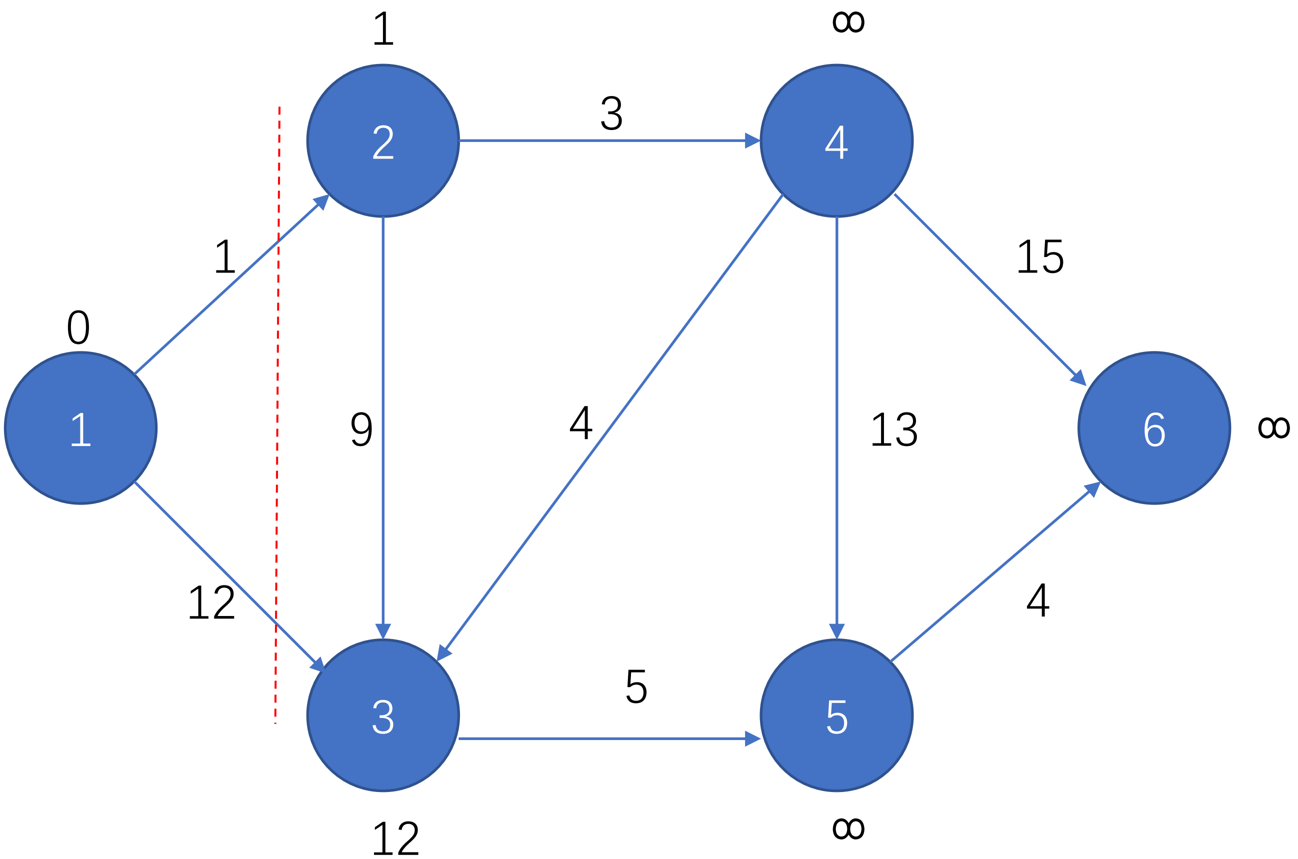


3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

```
Dijkstra算法
输入：含权有向图G={V,E},  V={1,2,…,n}
输出：G中顶点1到其他顶点的距离
1.for y←2 to n:
2.       if y相邻于1 then λ[y]←length[1,y]
3.       else λ[y]=∞
4.       end if
5.end for
6.for j←1 to n-1
7.       令y∈Y,  使得λ[y]为最小
8.       X← X∪{y}      {将顶点y加入X}
9.       Y← Y-{y}      {将顶点y从y中删除}
11.      for 每条边(y,w)
12.         if w∈Y and λ[y]+length[y,w] < λ[w] then
13.              λ[w]← λ[y]+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。

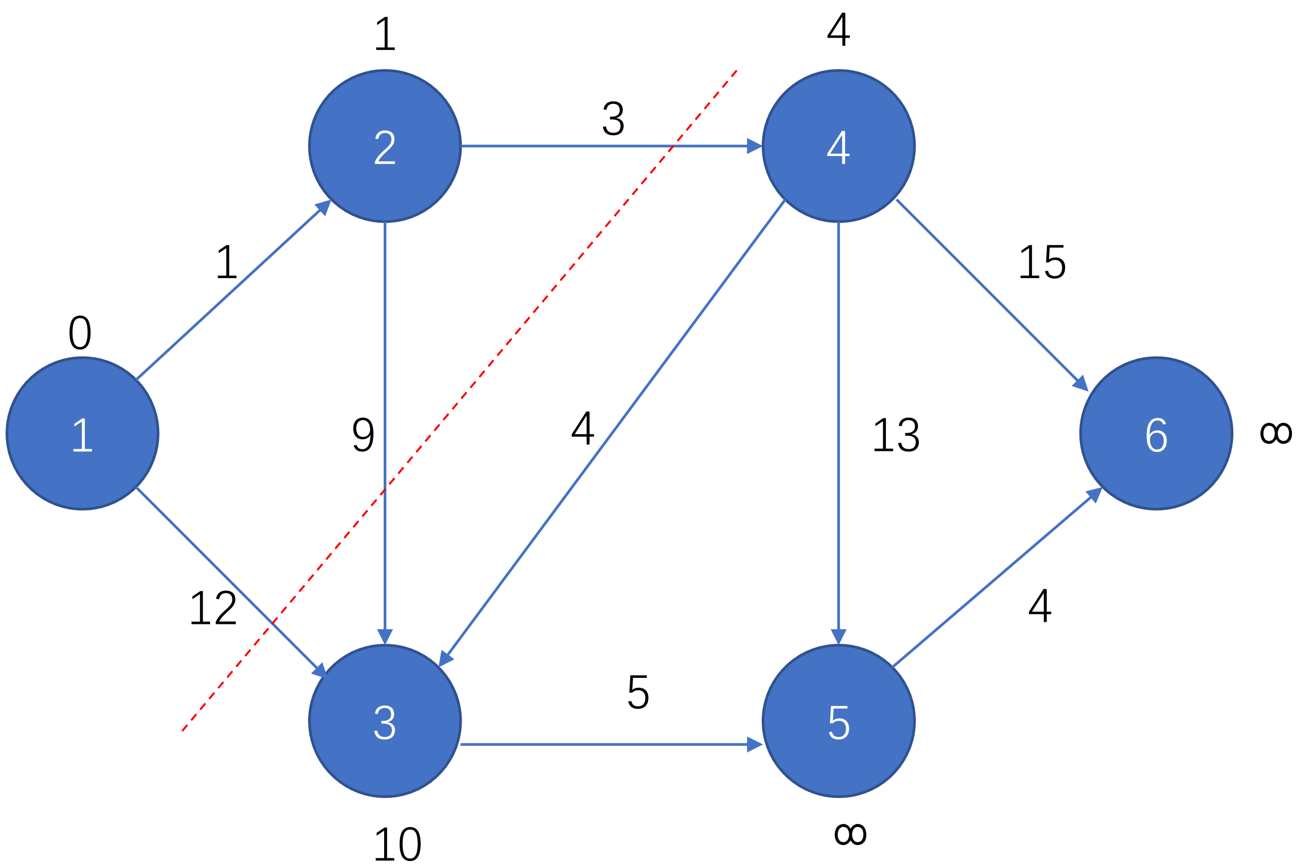


3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

加入顶点2:

λ[3]=12，λ[2]+length(2,3)=10, 10<12, 更新顶点3到1的最短路径。

λ[4]=∞，λ[2]+length(2,4)=4， 4<∞， 更新顶点4到1的最短路径。



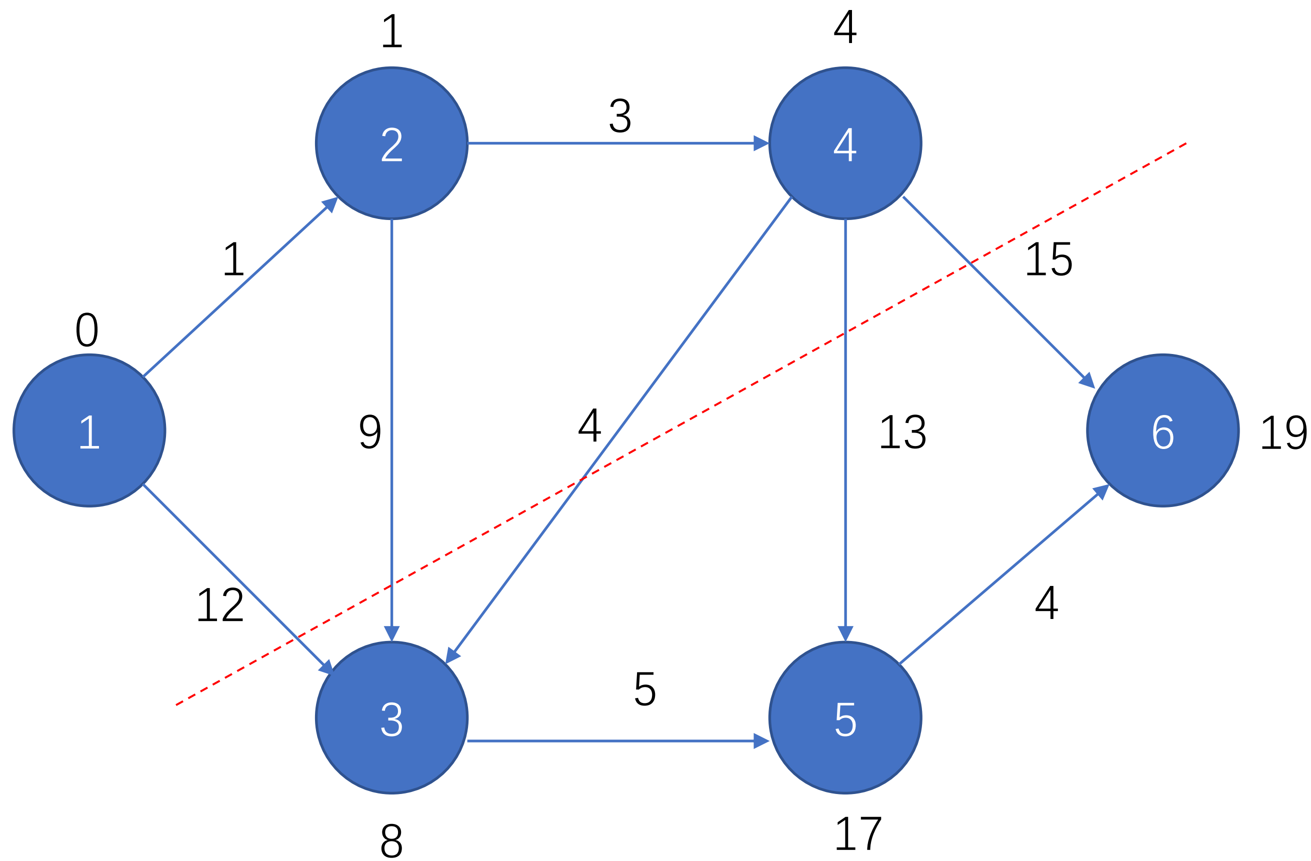
3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

加入顶点4:

λ[6]= ∞, λ[4]+length(4,6)=19, 19< ∞,更新顶点6到1的最短路径。

λ[5]= ∞, λ[4]+length(4,5)=17, 17< ∞,更新顶点5到1的最短路径。

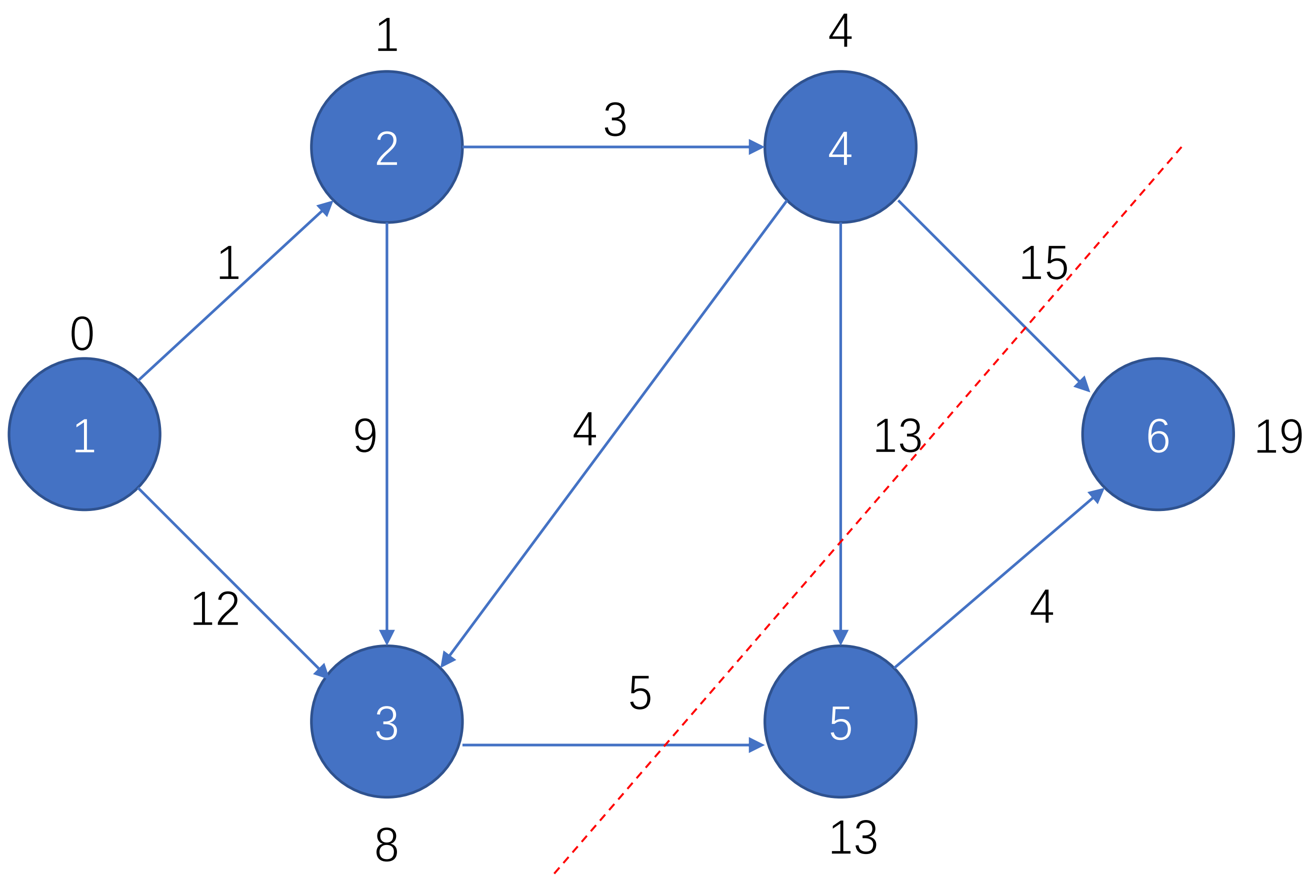
λ[3]=10, λ[4]+length(4,3)=8, 8<10, 更新顶点3到1的最短路径。



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:

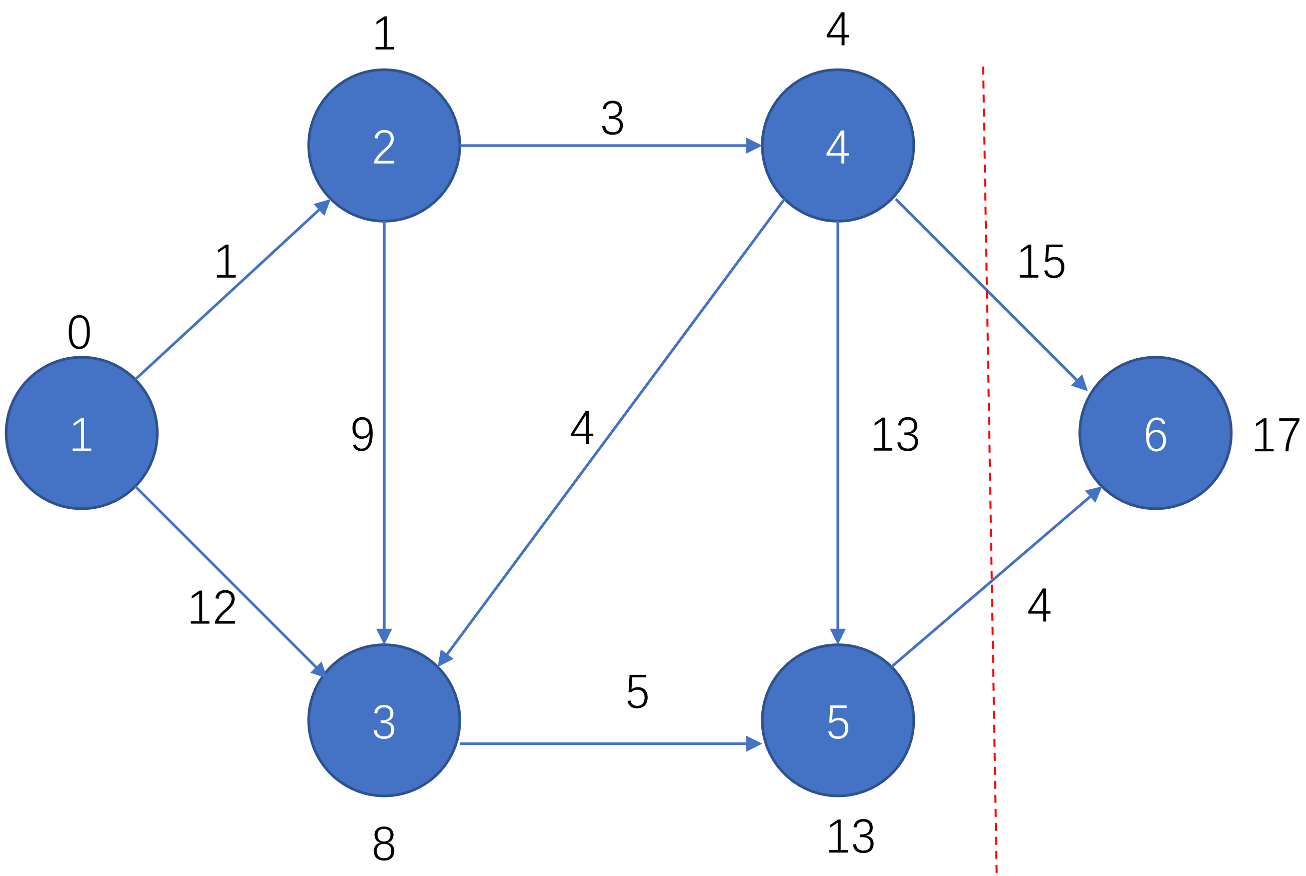
λ[5]= 17, λ[3]+length(3,5)=13, 13< 17,更新顶点5到1的最短路径。



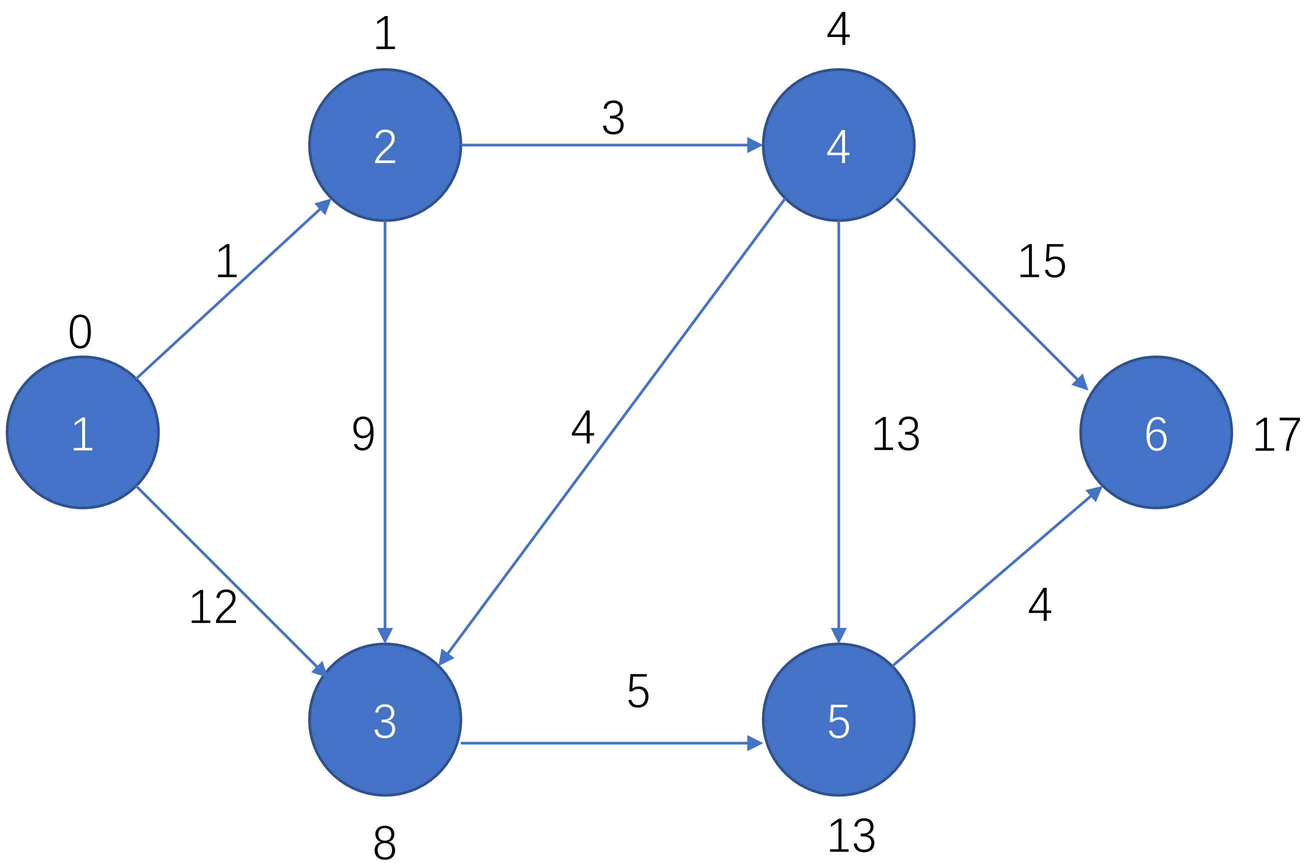
3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。

加入顶点5：

λ[6]= 19, λ[5]+length(5,6)=17, 17< 19,更新顶点6到1的最短路径。



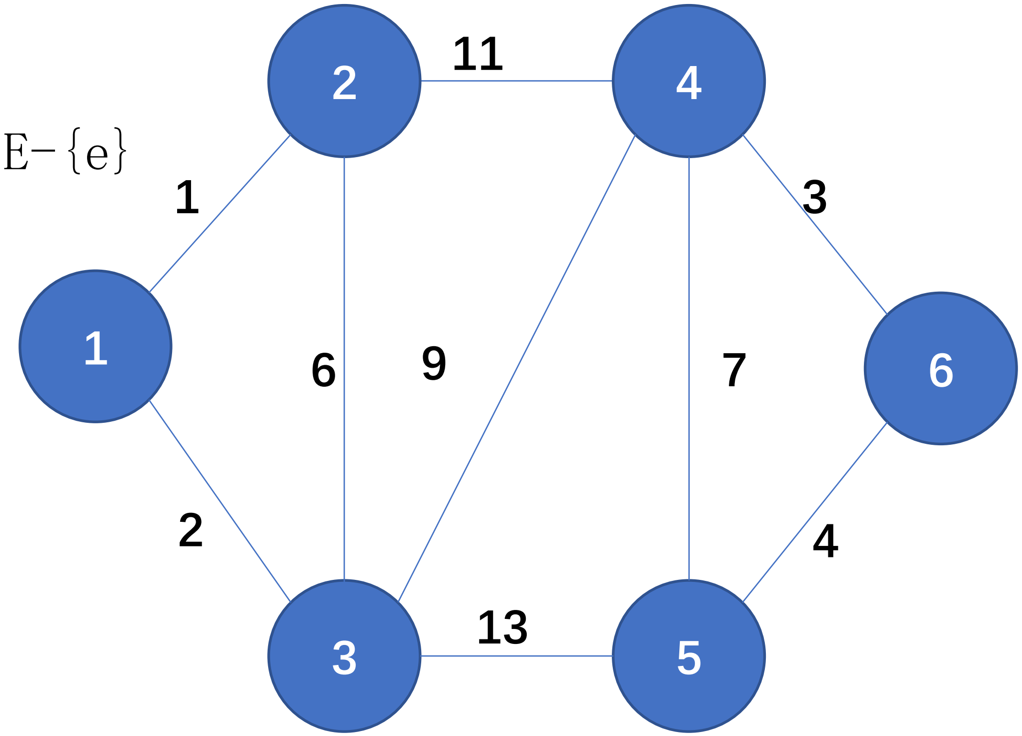
3. 已知含权有向图如下，采用Dijkstra算法求顶点1的单源最短路径。



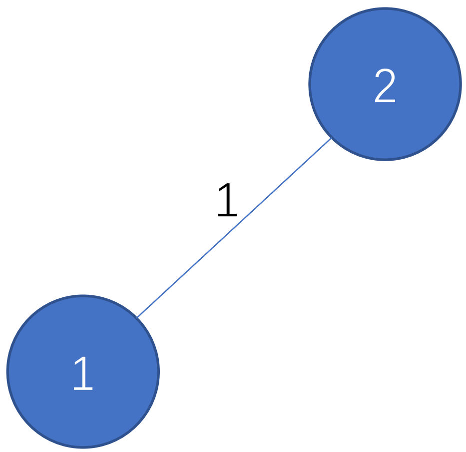
4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。

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

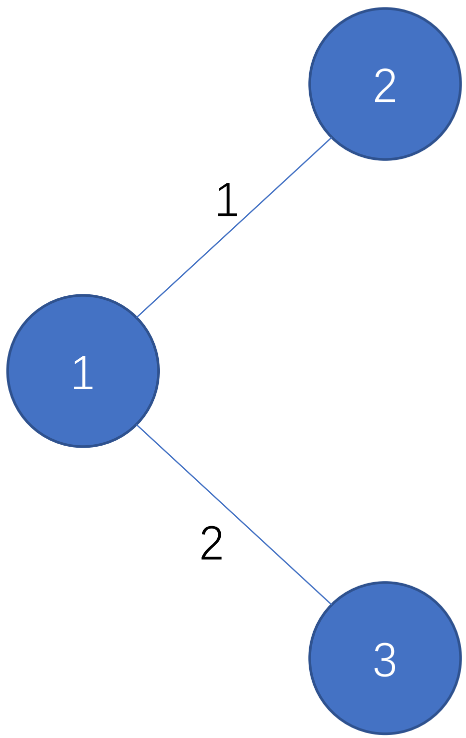
```
1.按非降序权重将E中的边排序
2.  T={ }
3.while |T| < n-1
4.      e ←E中第一个元素
5.       if e不会造成回路
6.           then T←T+{e}, E←E-{e}
6.      end if
7.      if E ={} then 算法结束
8.      end if
9.end while
```



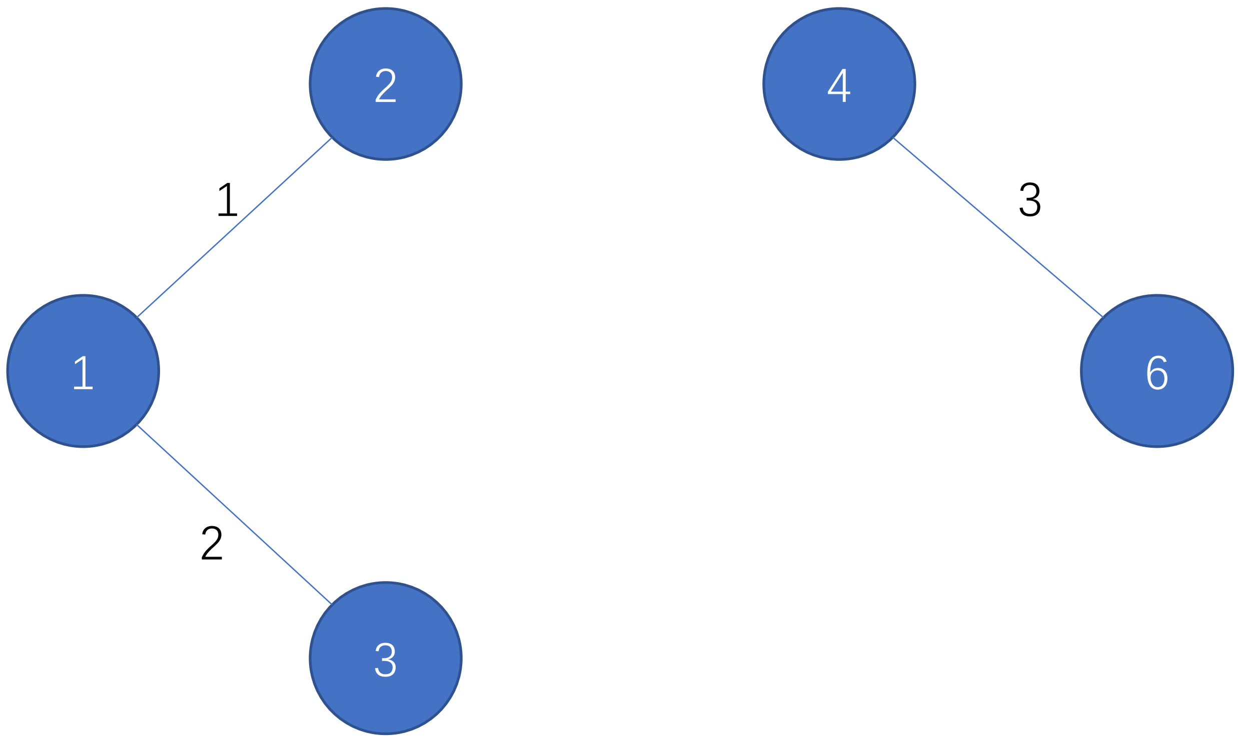
4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。



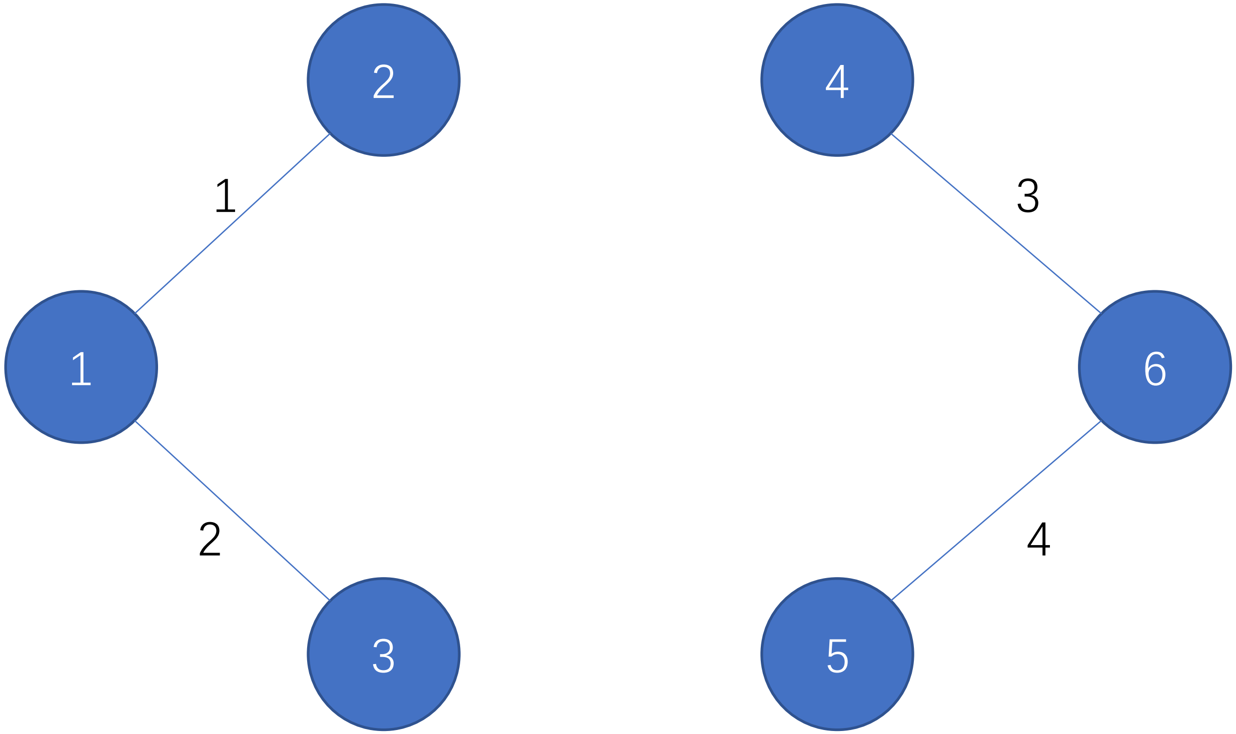
4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。



4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。

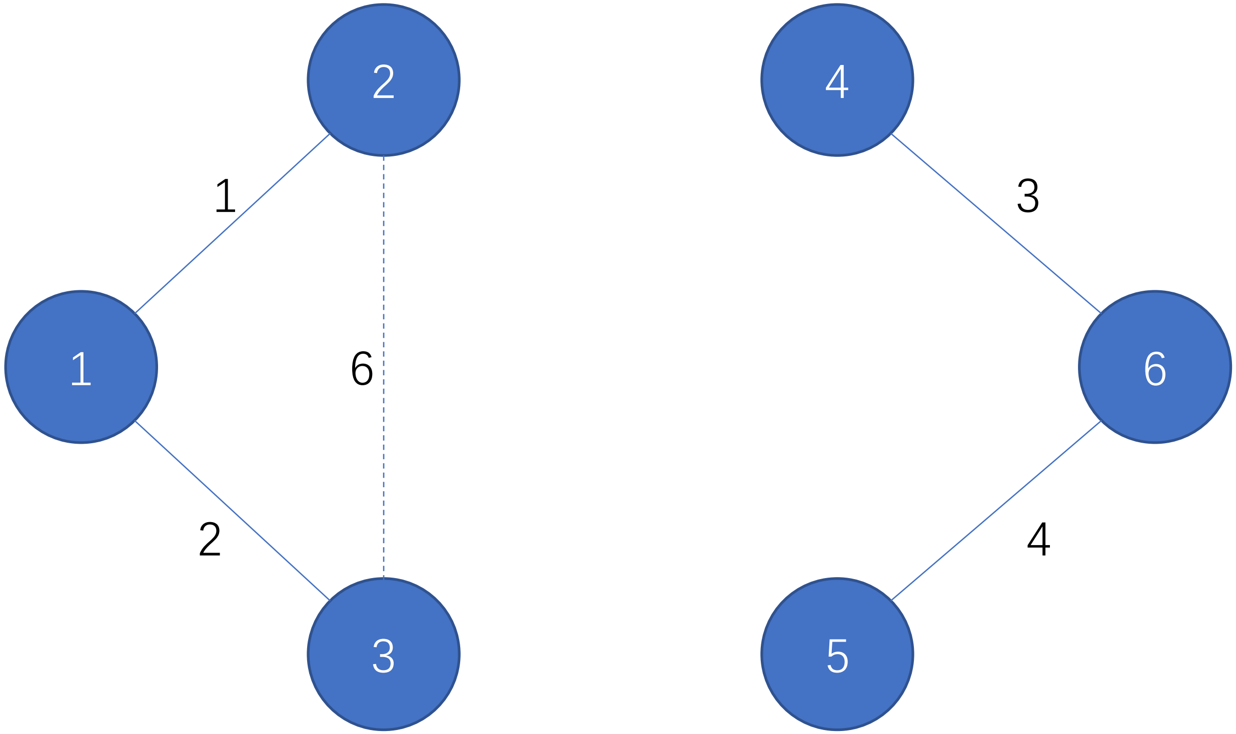


4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。



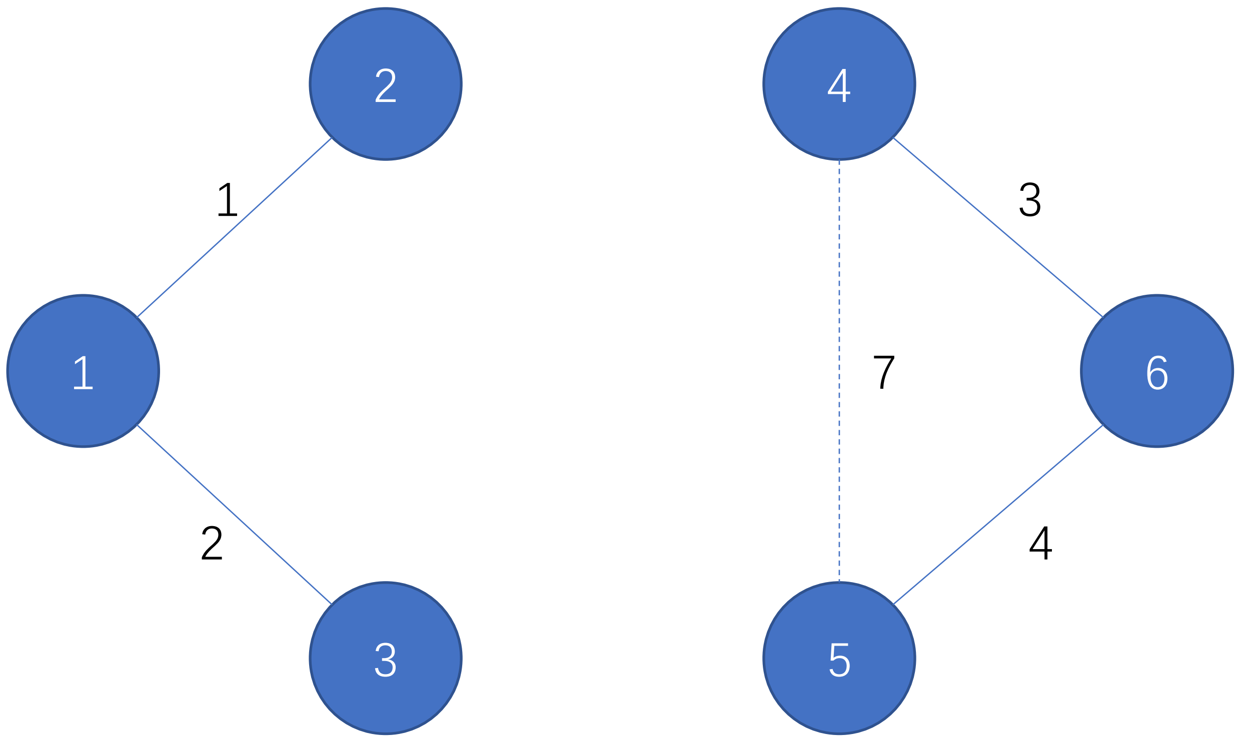
4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。

由于边(2,3)构成回路，舍弃

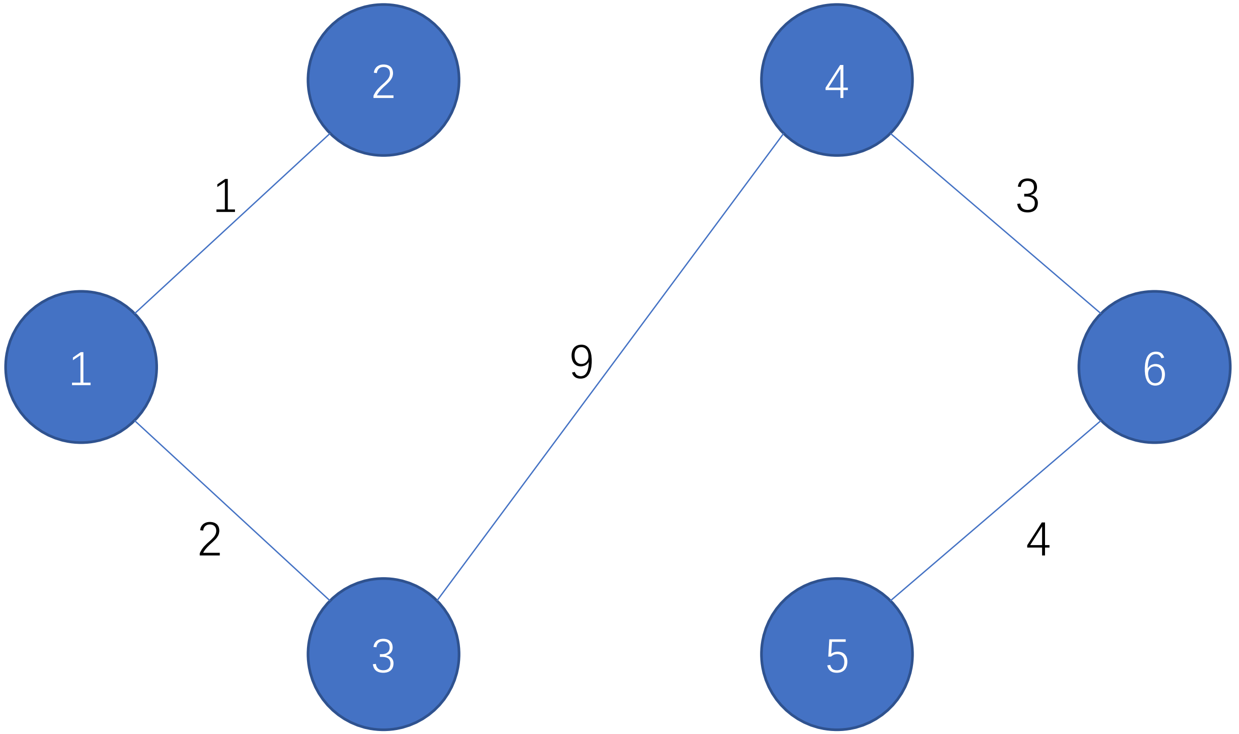


4. 已知有权无向图如下，根据Kruskal算法画出最小生成树。

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。



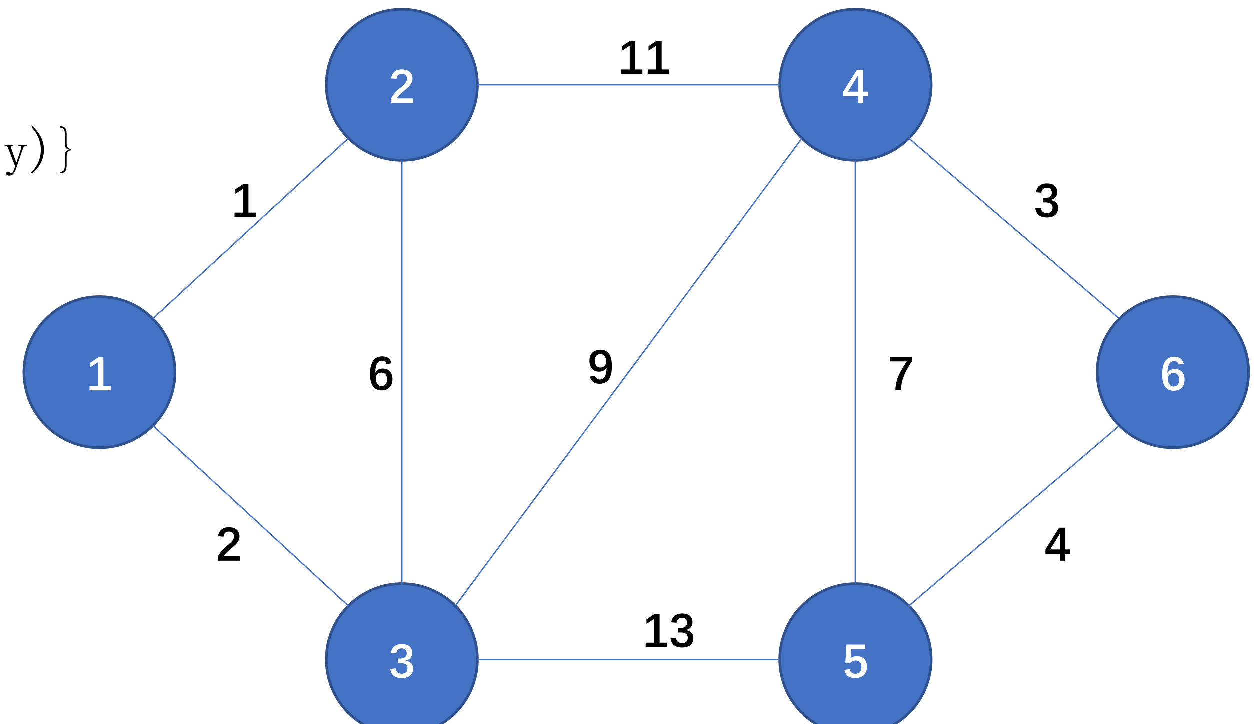
5. 已知有权无向图如下，根据Prim算法画出最小生成树。

Prim算法

输入：包含n个顶点的含权连通无向图G={V,E}

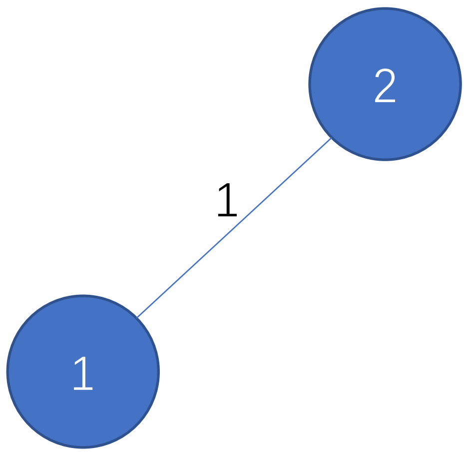
输出：由G生成的最小耗费生成树T组成的边的集合

1. T={ }，X←{1}，Y←V-{1}
2. while Y≠{ }
3. 设(x,y)是最小权重的边，其中x∈X，y∈Y
4. X←X∪{y}
5. Y←Y-{y}
6. T←T∪{(x,y)}
7. end while



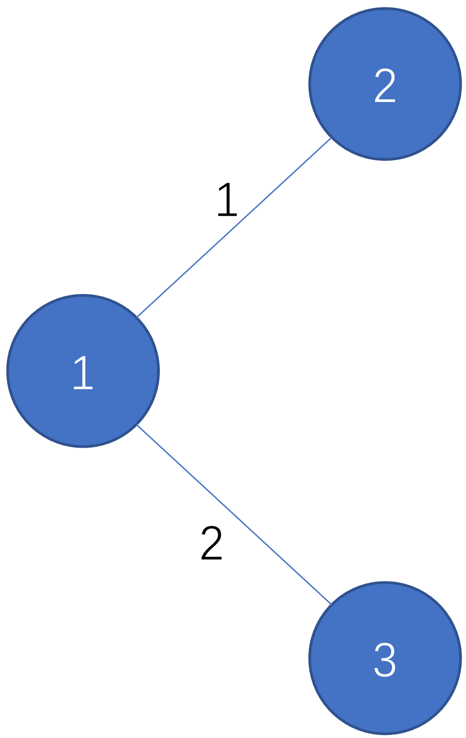
5. 已知有权无向图如下，根据Prim算法画出最小生成树。

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



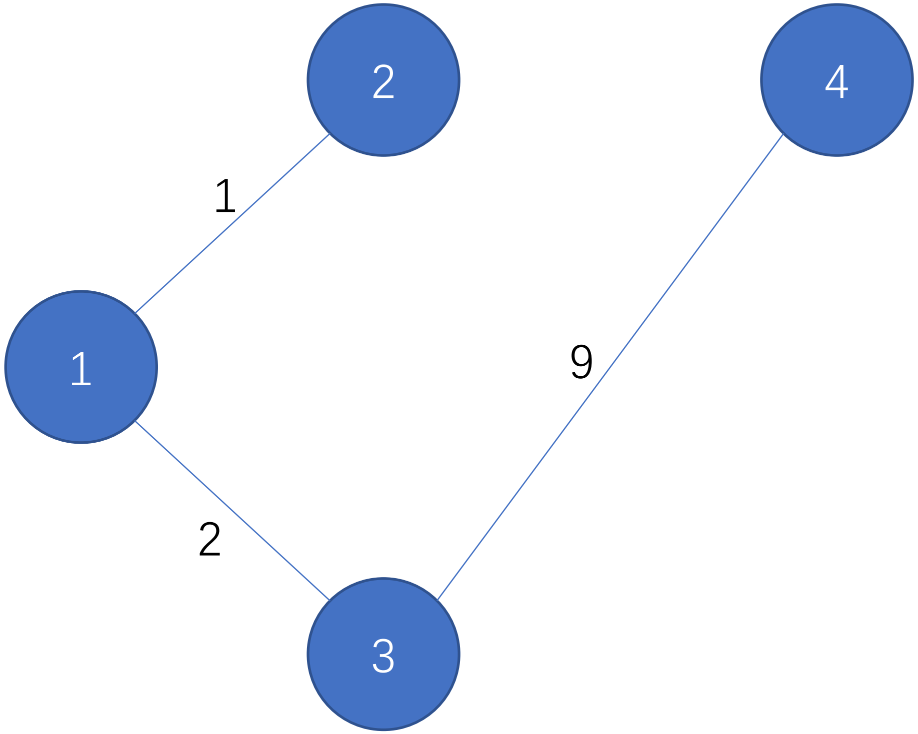
5. 已知有权无向图如下，根据Prim算法画出最小生成树。

$X=\{1,2,3\}$　$Y=\{4,5,6\}$



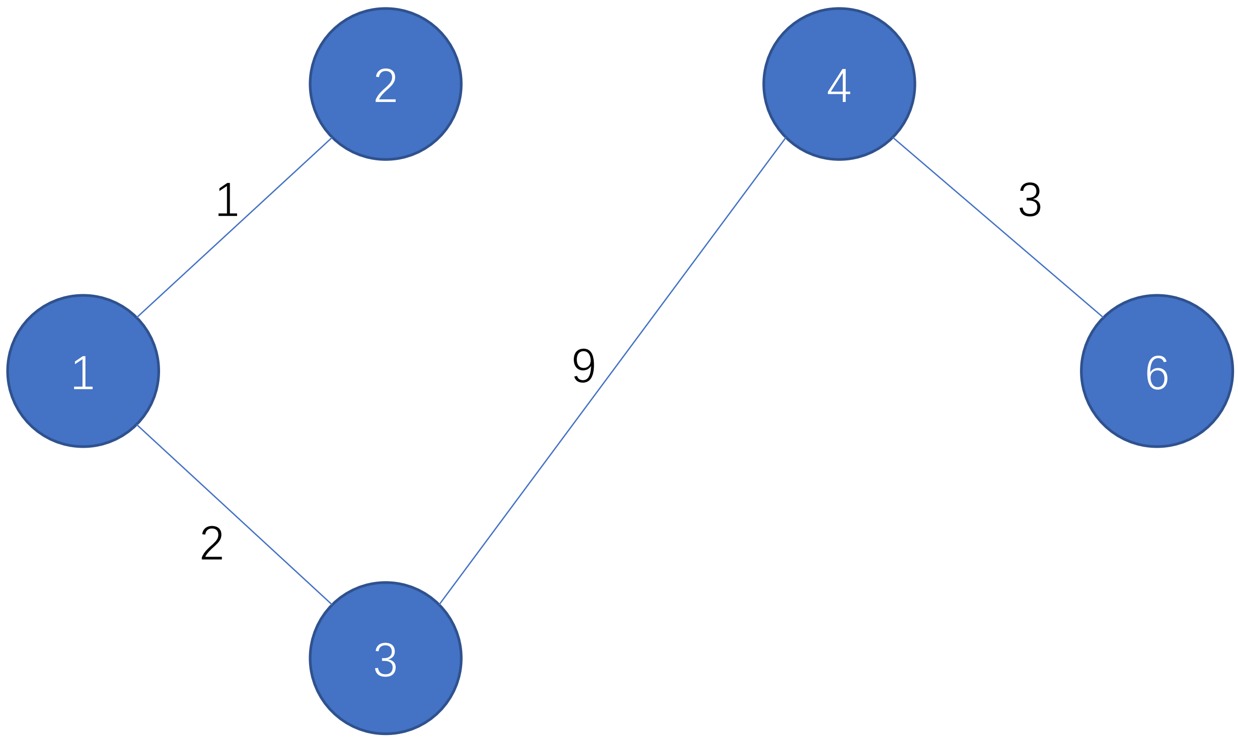
5. 已知有权无向图如下，根据Prim算法画出最小生成树。

X={1, 2, 3, 4}　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}　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}　Y={}