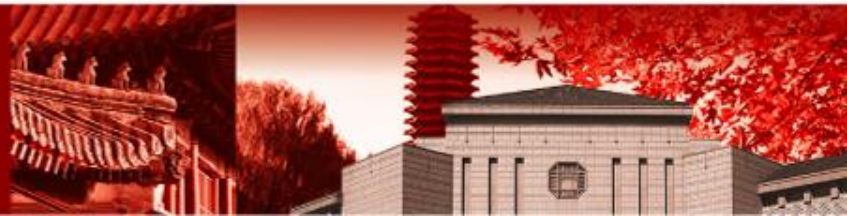
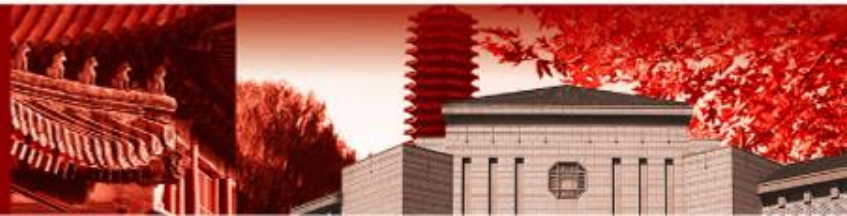
数据结构与算法B

15-拓扑排序与关键路径

目录

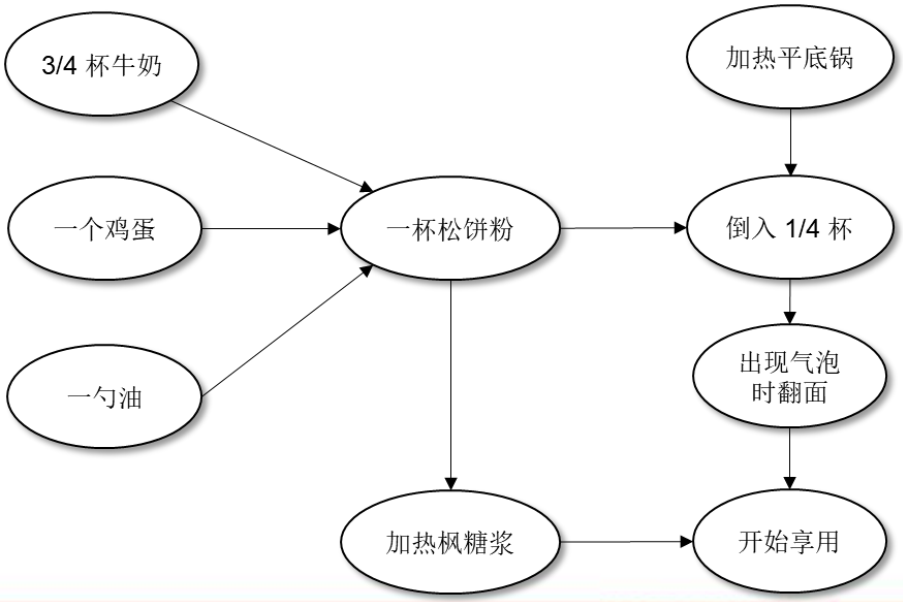


15.1 拓扑排序算法



问题引入

- 具有先后次序的流程的问题，通常可以用图来建模
- 下图展示了制作一批早餐松饼的流程
  - 关键在于，确定一个活动的执行顺序
  - 活动顺序不唯一，例如可以先加热平底锅，也可以先准备原材料



问题引入

- 大学的课程培养方案也涉及拓扑排序问题
  - 部分课程具有先修课要求
  - 学生必须修读要求的全部课程，因而需要自行决定修读的顺序
- 其他的拓扑排序例子包括软件项目调度、优化数据库查询的优先级表等等

拓扑排序问题

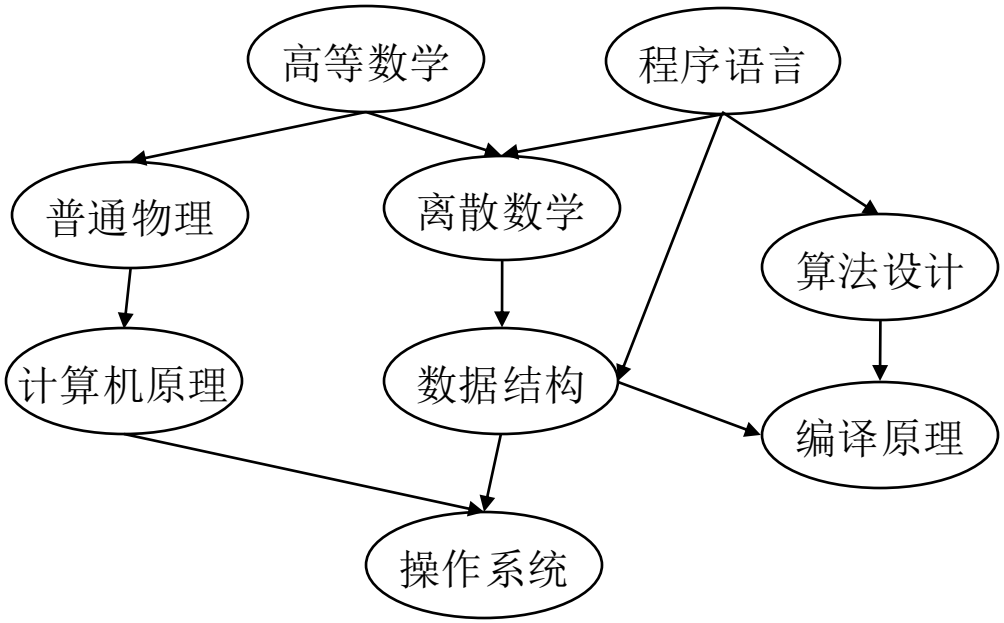
- 在有向图中，用顶点代表活动，有向边代表活动间的先后关系，则该有向图称为顶点活动图（AOV网，Activity on Vertex Network）
- 对于一个AOV网 G = (V, E)，其拓扑排序是 G 中所有结点的一种线性次序，满足如下条件
  - 如果图 G 包含边 <u, v>，则顶点 u 在拓扑排序中处于顶点 v 的前面
- 对于有向无环图（Directed Acyclic Graph, DAG），一定存在拓扑排序；如果图中存在环，则不存在拓扑排序，算法应该识别环的存在。

拓扑排序算法

- 直观的思路：不断查看图中是否存在可以立即完成的活动，如果存在就进行该活动
- 对应到图中：
  - 从 AOV 网中选择一个入度为 0 的顶点，将其输出到拓扑排序序列中
  - 从 AOV 网中删除此顶点，以及该顶点的所有出边
- 反复执行以上两步，直到：
  - 所有顶点都已经输出，此时拓扑排序已经完成
  - 或者，剩下的顶点入度都不为 0，此时说明 AOV 网中存在回路，拓扑排序无法进行

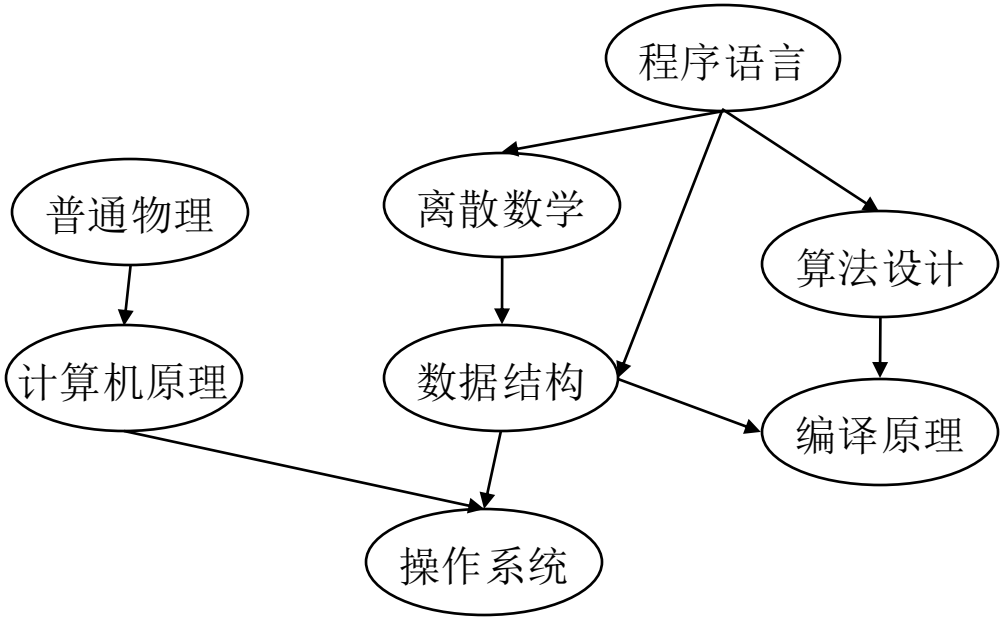
拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[高等数学，程序语言]
  - 拓扑排序序列：[ ]



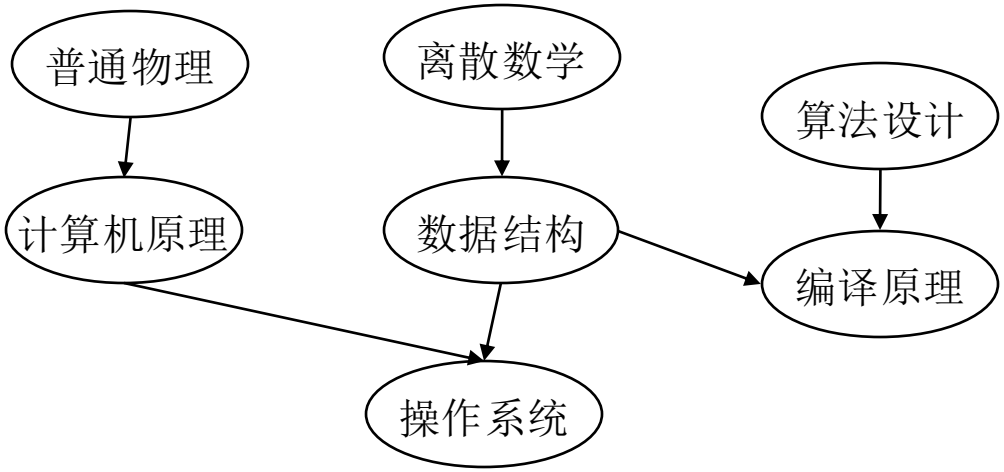
拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[程序语言，普通物理]
  - 拓扑排序序列：[高等数学]



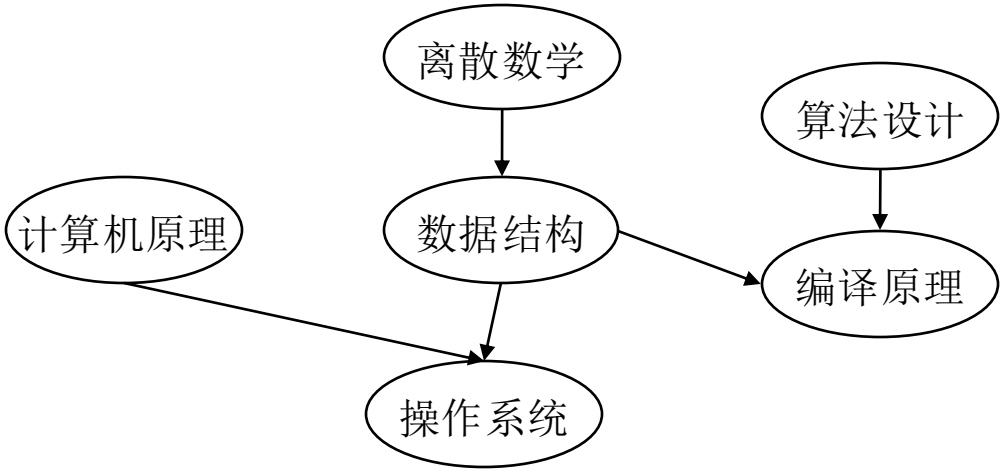
拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[普通物理，离散数学，算法设计]
  - 拓扑排序序列：[高等数学，程序语言]



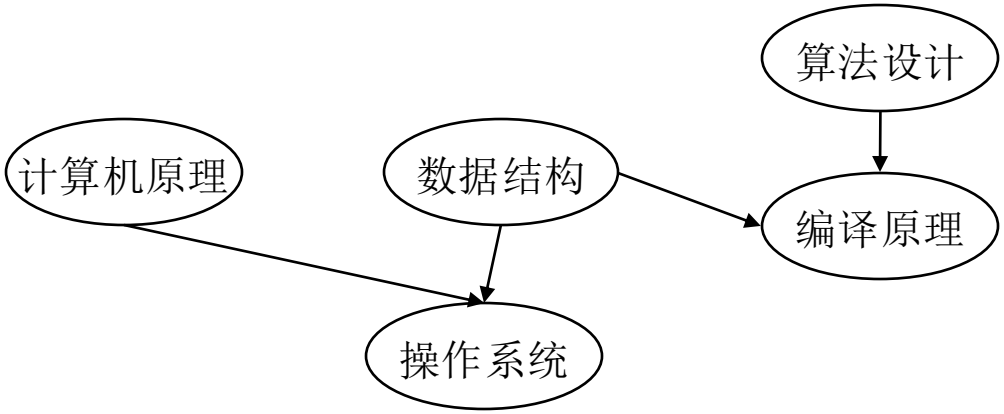
拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[离散数学，算法设计，计算机原理]
  - 拓扑排序序列：[高等数学，程序语言，普通物理]



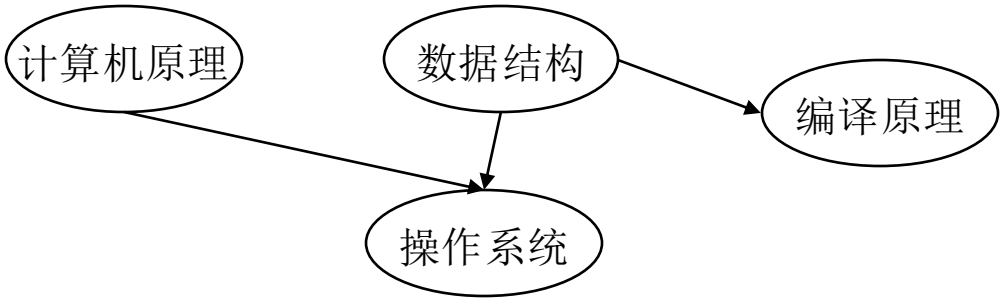
拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[算法设计，计算机原理，数据结构]
  - 拓扑排序序列：[高等数学，程序语言，普通物理，离散数学]



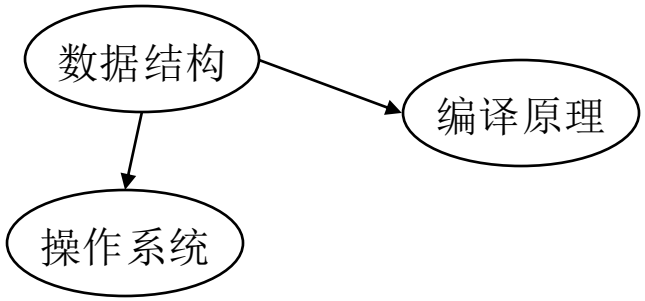
拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[计算机原理，数据结构]
  - 拓扑排序序列：[高等数学，程序语言，普通物理，离散数学，算法设计]



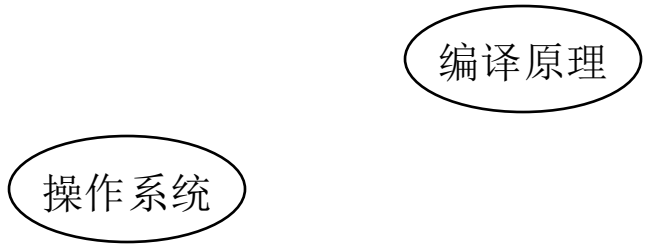
拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[数据结构]
  - 拓扑排序序列：[高等数学，程序语言，普通物理，离散数学，算法设计，计算机原理]



拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[编译原理，操作系统]
  - 拓扑排序序列：[高等数学，程序语言，普通物理，离散数学，算法设计，计算机原理，数据结构]



拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[操作系统]
  - 拓扑排序序列：[高等数学，程序语言，普通物理，离散数学，算法设计，计算机原理，数据结构，编译原理]

拓扑排序算法：示例

- 对如下图进行拓扑排序
  - 当前入度为 0 的结点：[]
  - 拓扑排序序列：[高等数学，程序语言，普通物理，离散数学，算法设计，计算机原理，数据结构，编译原理，操作系统]

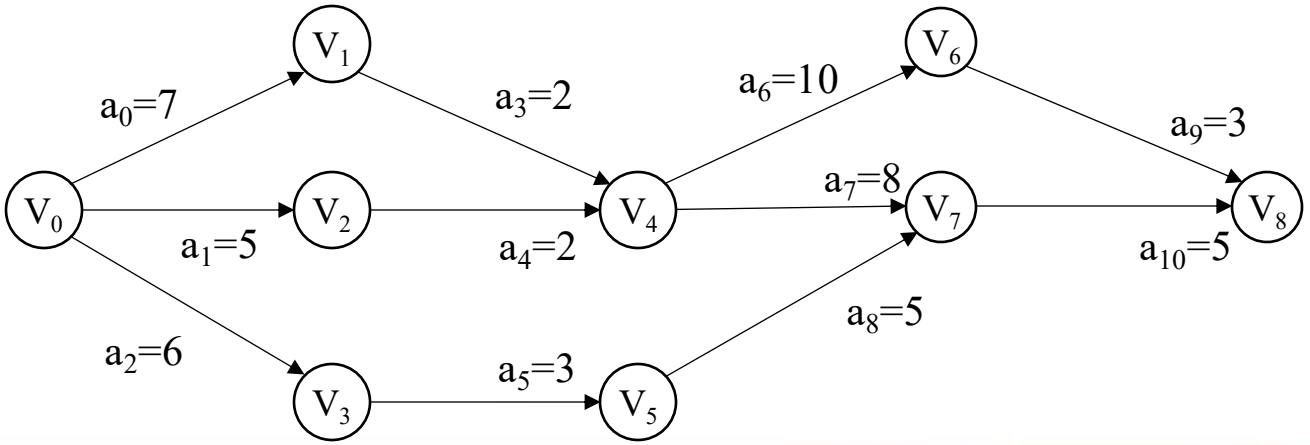
拓扑排序算法的时间复杂度

- 记 n = |V|，e = |E|，拓扑排序算法的主要代价包括：
  - 初始时维护所有结点的度数，代价为 O(n + e)
  - 每条边会被处理一次，代价为 O(e)
  - 每个顶点会被添加到队列中，并输出到结果序列中，代价为 O(n)
- 因此，拓扑排序算法的时间复杂度为 O(n+e)

15.2 关键路径

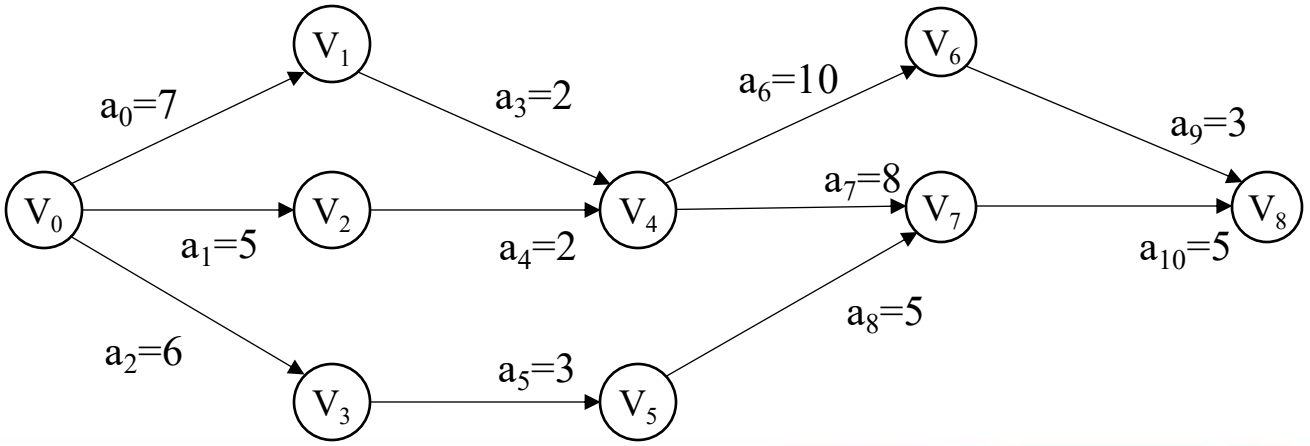
问题引入

- 拓扑排序将 AOV 网络转化为可以顺序执行的线性序列
- 而现实场景中，先后次序无关的多个活动往往是可以同时进行的，完成不同活动所需的时间也往往不同
  - 此时，适合使用边活动图（AOE 网，Activity on Edge Network）
  - 顶点表示事件，边代表活动，边权代表活动所需的时间
  - 例如，V_0 表示起始，V_1 表示活动 a_0 结束，V_8表示全部活动结束



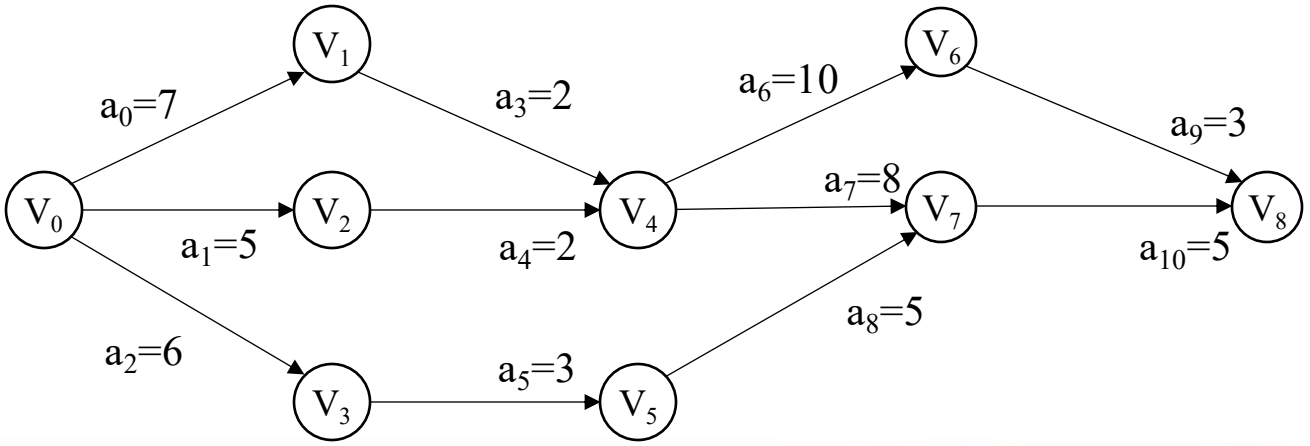
问题引入

- 我们的目标是使整个工程尽快完成，即回答以下问题
- 1. 每个事件最早可以在什么时间发生?
- 2. 整个工程最早可以在什么时间完成？
  - 所有的事件都发生了就代表着完成了整个工程。因此，所有事件最早发生时间的最大值即为工程的最早完成时间



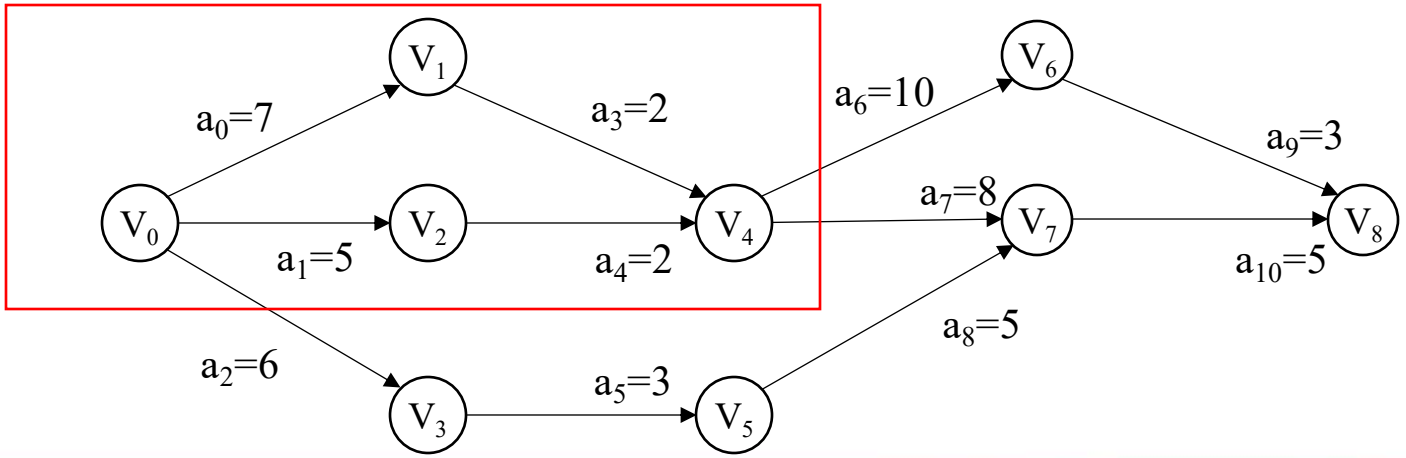
问题引入

- 3. 每个事件最晚可以在什么时间发生？
  - 有的事件，并不是可以发生了就必须立即发生；将其推迟一些发生，也可以不影响整体的工期



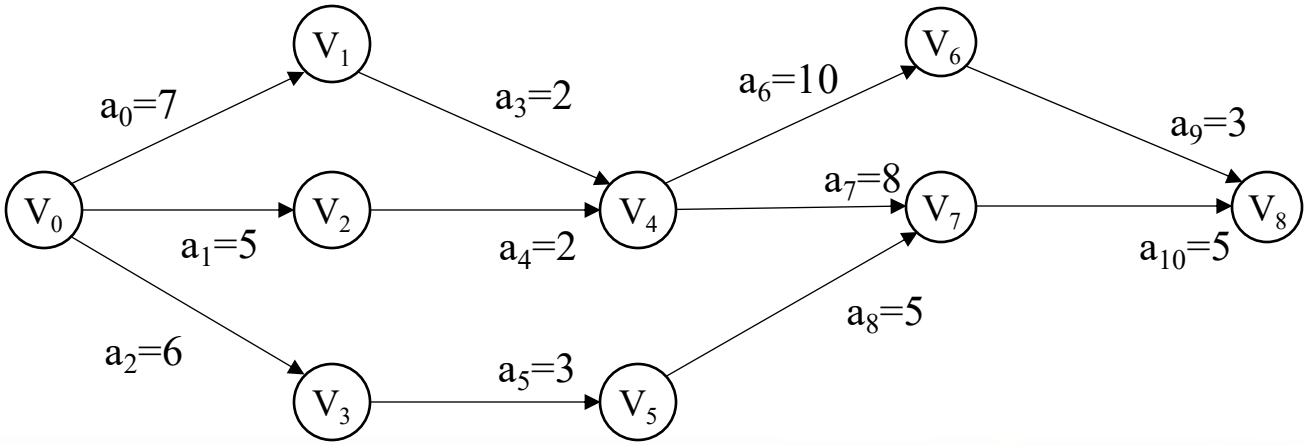
问题引入

- 3. 每个事件最晚可以在什么时间发生？
  - 有的事件，并不是可以发生了就必须立即发生；将其推迟一些发生，也可以不影响整体的工期
  - 例如事件 V_2，最早发生时间为 5，但实际上可以将活动 a_1 推迟，使 V_2 在 7 时刻发生，而不影响工程的最早完成时间



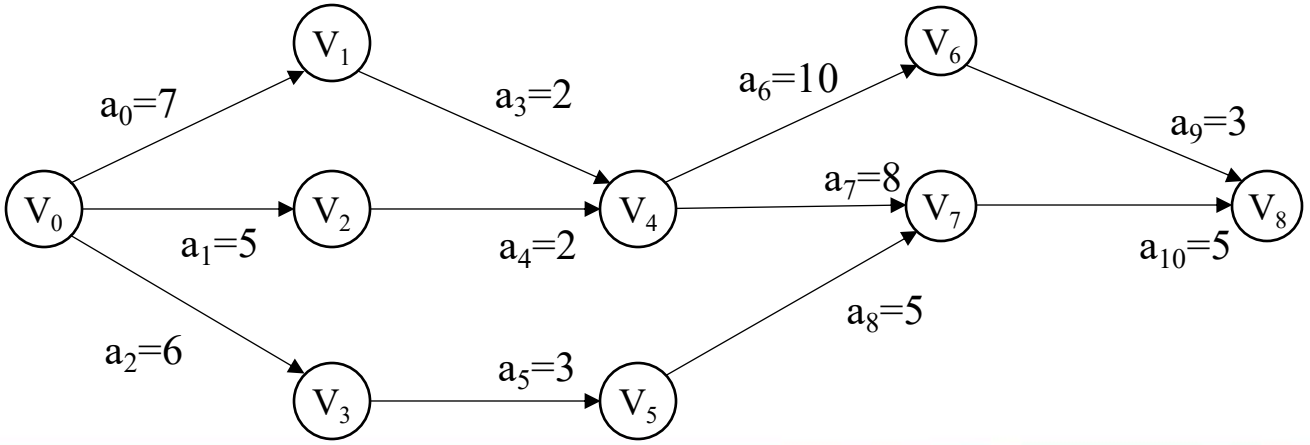
关键活动

- 4. 哪些活动是关键活动?
  - 活动的最早开始时间 = 起点事件的最早发生时间
  - 活动的最晚开始时间 = 终点事件的最晚发生时间 – 活动时长
  - 最早开始时间与最晚开始时间相等的活动，称为工程的关键活动
  - 关键活动不得推迟，否则将导致整个工期的延长



关键路径

- 5. 求出关键路径
  - 关键路径是 AOE 网络上权值之和最大的路径，即最长路径
  - 关键路径上的活动都是关键活动
  - 关键路径可能不唯一。因此，即便减少了某一条关键路径上活动的耗时，并不一定导致工程提前完成



关键路径算法

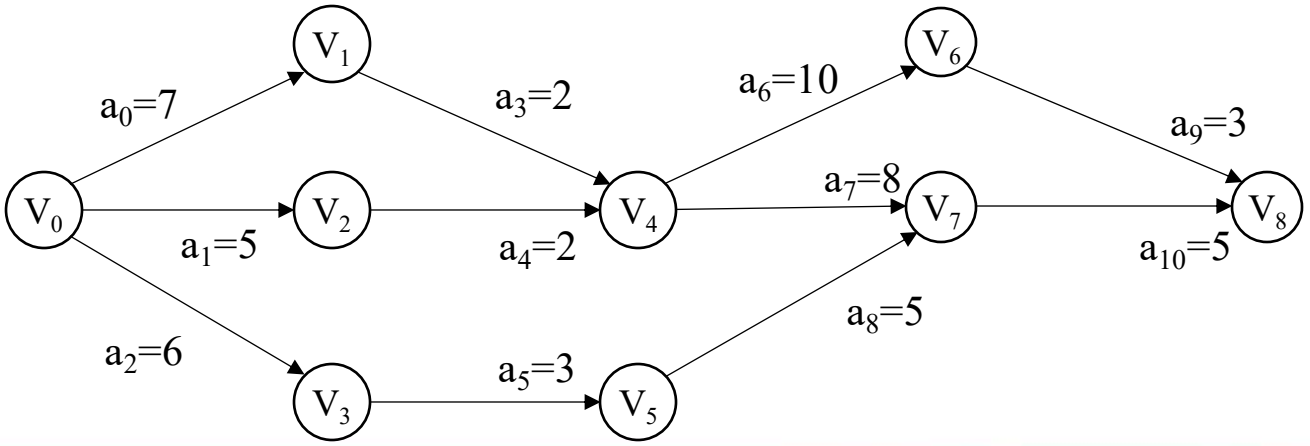
- 关键路径算法的主要思想为，借助拓扑排序，求出每个事件的最早与最晚发生时间，进而确定关键活动与关键路径。
- 为了求出所有事件 V_i 的最早发生时间 earliest[i]：
  - 首先将所有 earliest[i] 都初始为 0
  - 对 AOE 网络进行拓扑排序
  - 按拓扑排序序列的顺序，更新事件的 earliest 值
    - 对于顶点 V_i，考察所有出边 $< V_i, V_j >$ 即相应的权值 W_{ij}
    - 由于 V_i 最早在 earliest[i] 发生，因此 V_j 最早在 earliest[i] + W_{ij} 发生
    - 执行 earliest[j] = max(earliest[j], earliest[i] + W_{ij})

关键路径算法

- 为了求出所有事件 V_i 的最晚发生时间 latest[i]：
  - 首先将所有 latest[i] 都初始为工程的最早完成时间，即所有 earliest 值的最大值
  - 对 AOE 网络进行拓扑排序
  - 按拓扑排序序列的逆序，更新事件的 latest 值
    - 对于顶点 V_i，考察所有出边 $< V_i, V_j >$ 即相应的权值 W_{ij}
    - 由于 V_j 需要在 latest[j] 之前发生，因此 V_i 最晚需要在 latest[j] – W_{ij} 之前发生
    - 执行 earliest[i] = min(earliest[i], earliest[j] - W_{ij})
- 然后，基于 earliest 值与 latest 值求出每个活动的最早和最晚开始时间，确认关键活动与关键路径
- 算法的时间复杂度为 O(n+e)

关键路径算法

事件	V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
最早发生时间	0	7	5	6	9	9	19	17	22
最晚发生时间	0	7	7	9	9	12	19	17	22



关键路径算法

活动	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
最早开始时间	0	0	0	7	5	6	9	9	9	19	17
最晚开始时间	0	2	3	7	7	9	9	9	12	19	17