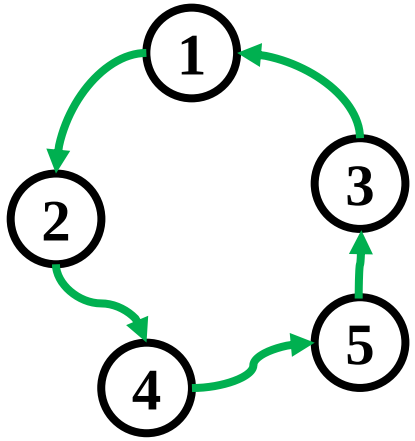
图算法篇：拓扑排序

童咏昕

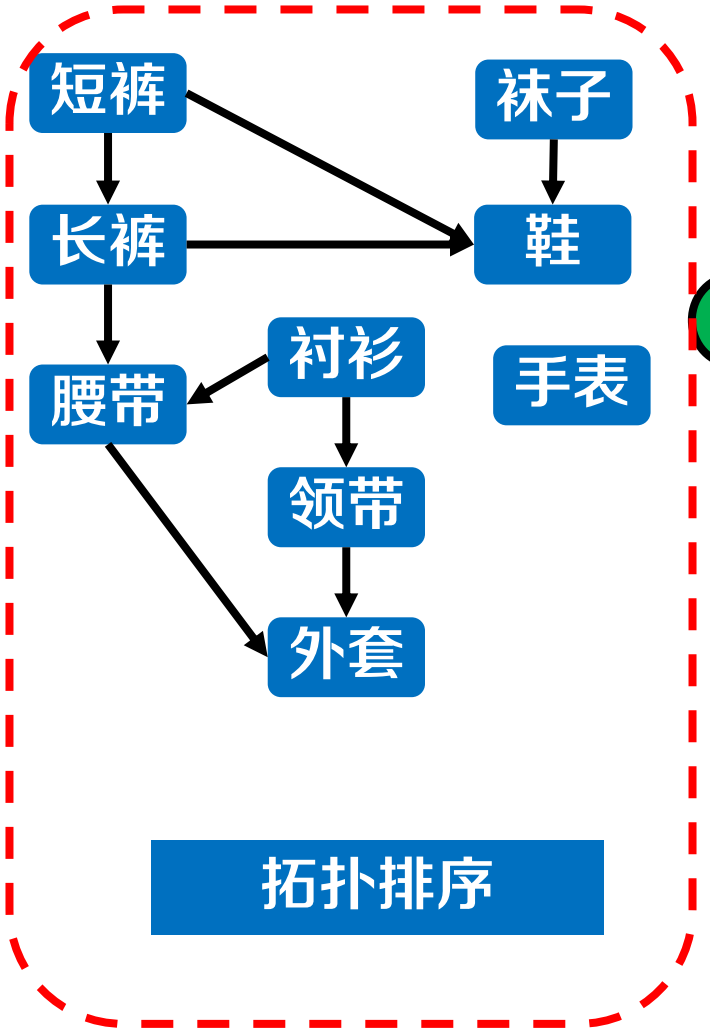
北京航空航天大学
计算机学院

中国大学MOOC北航《算法设计与分析》

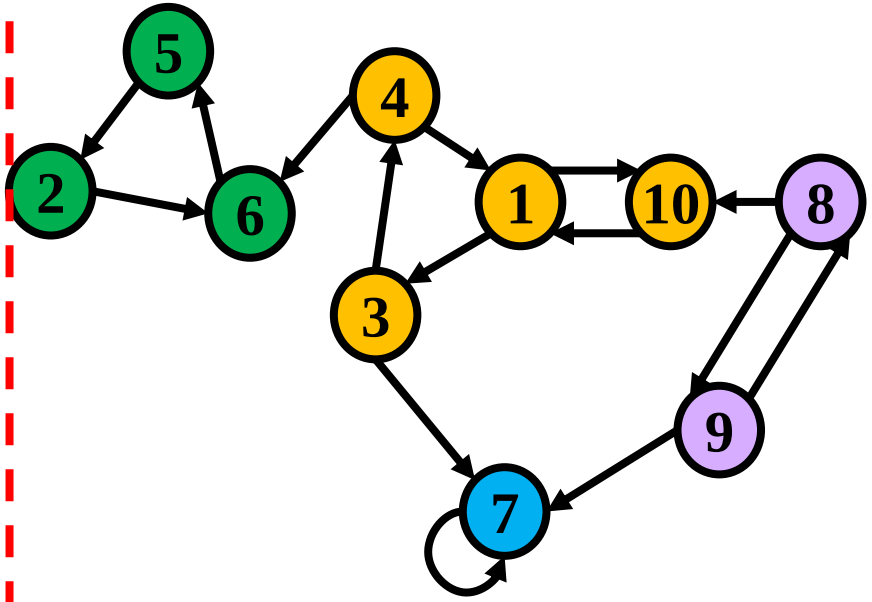
深度优先搜索应用



环路的存在性判断



拓扑排序



强连通分量

提纲

问题定义

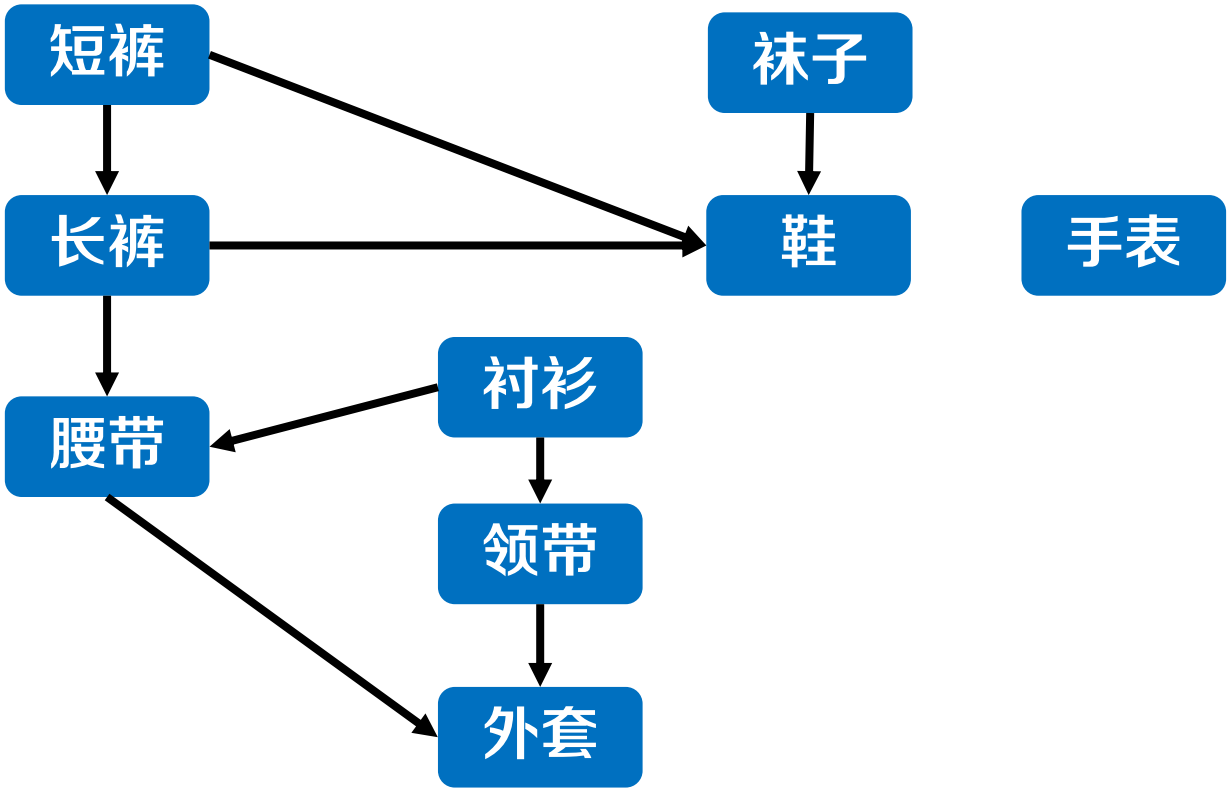
广度优先策略

深度优先策略

算法分析

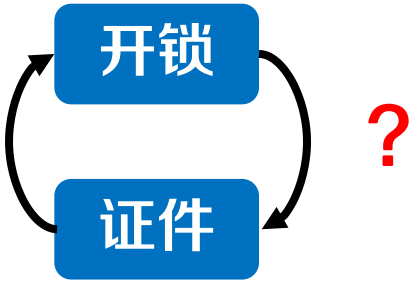
问题背景

- 穿衣步骤
  - 有向无环图（Directed Acyclic Graph，DAG）：表示事件发生的先后顺序



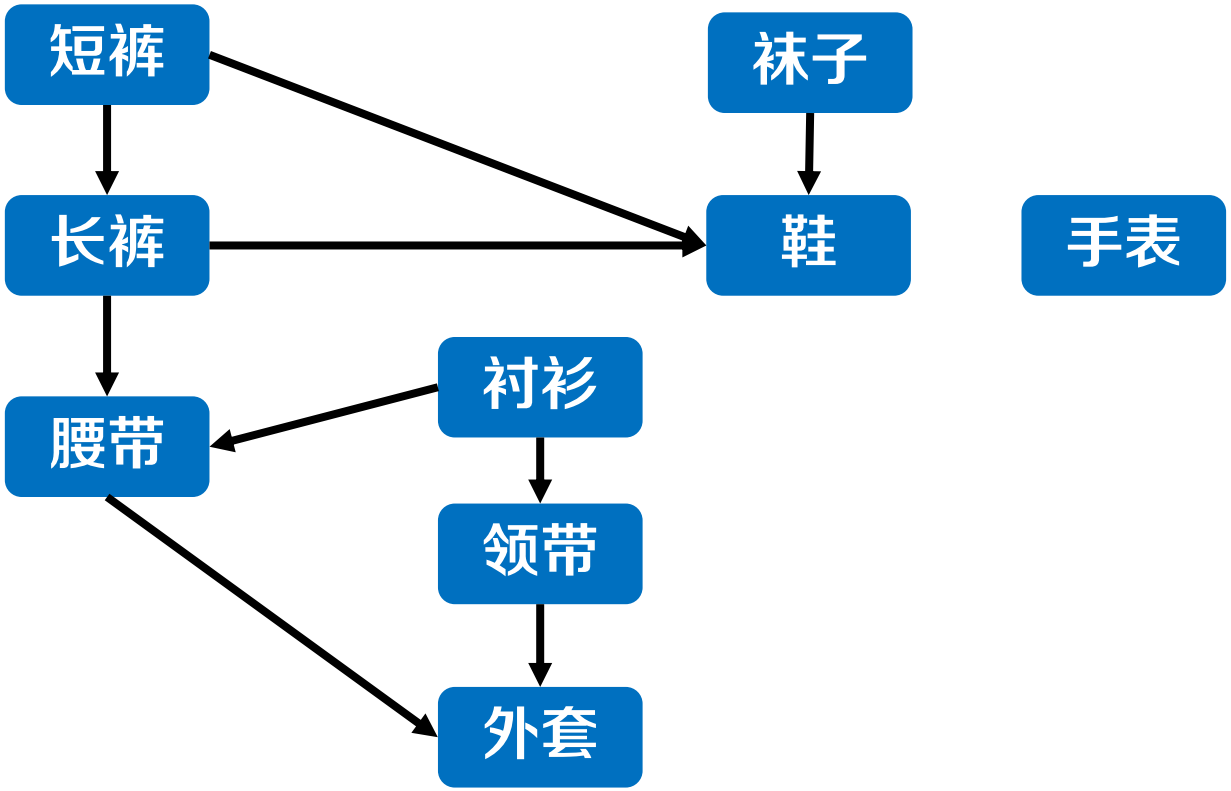
有向：规定先后顺序

无环：避免相互依赖

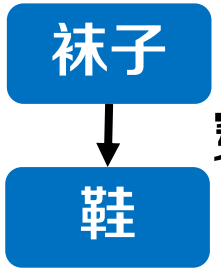


问题背景

- 穿衣步骤
  - 有向无环图（Directed Acyclic Graph，DAG）：表示事件发生的先后顺序

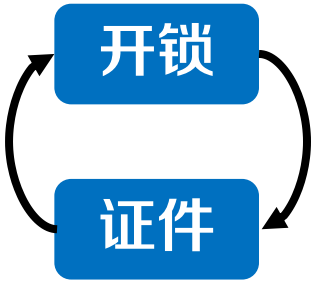


有向：规定先后顺序



穿袜子先于穿鞋

无环：避免相互依赖

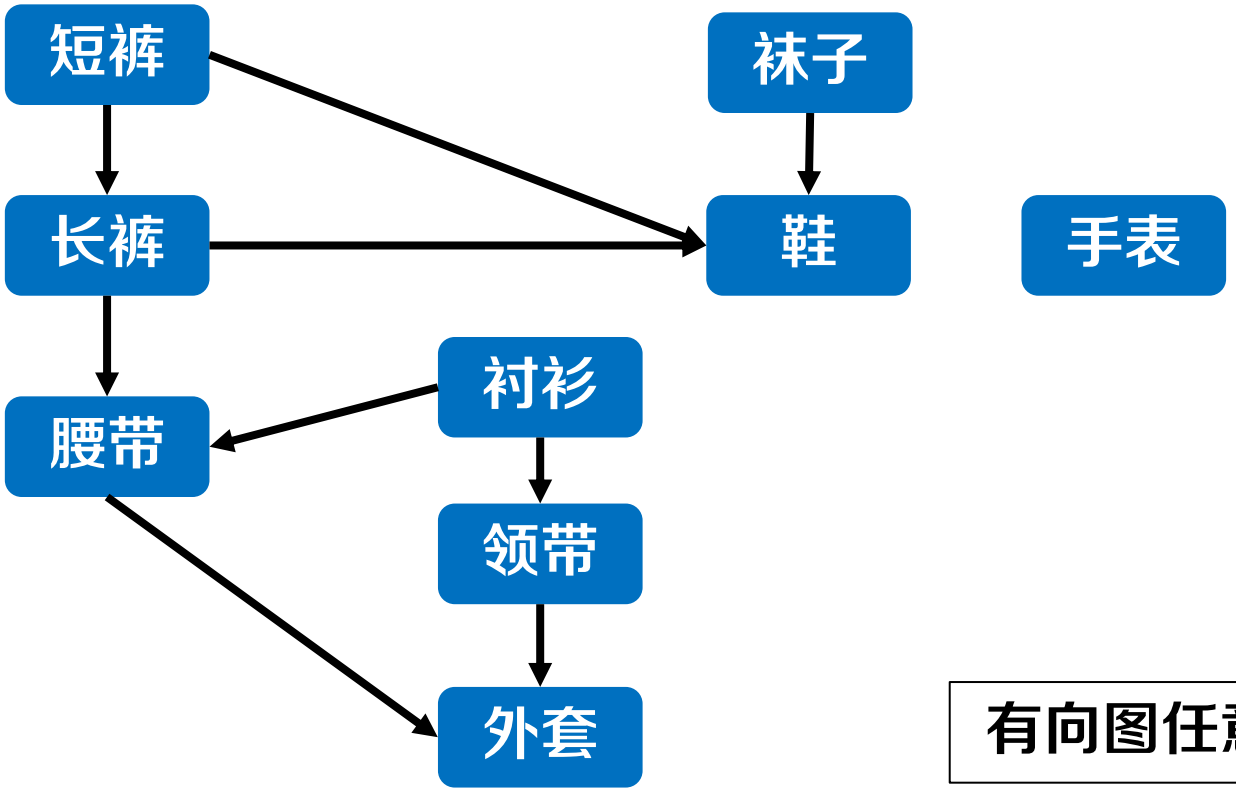


?

问题：如何确定一个可行的穿衣顺序?

问题背景

- 穿衣步骤
  - 有向无环图（Directed Acyclic Graph，DAG）：表示事件发生的先后顺序



有向图任意边的起点排在终点前面

问题定义

拓扑排序

Topological Sort

输入

- 有向无环图$G =< V, E >$

输出

- 图顶点V的拓扑序S，满足：

  对任意有向边(u, v)，排序后u在v之前

问题定义

拓扑排序

Topological Sort

输入

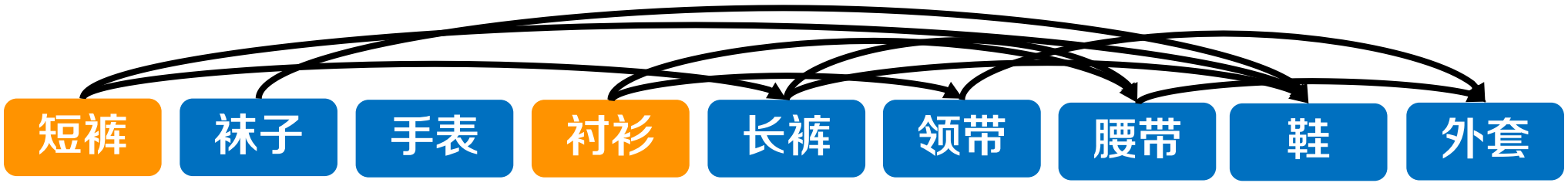
- 有向无环图$G =< V, E >$

输出

- 图顶点V的拓扑序S，满足：
  对任意有向边(u, v)，排序后u在v之前

- 拓扑序不唯一

有向图任意边的起点排在终点前面



问题定义

拓扑排序

Topological Sort

输入

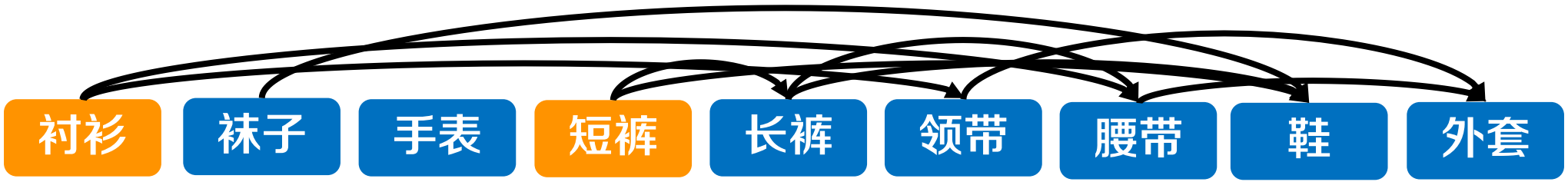
- 有向无环图$G =< V, E >$

输出

- 图顶点V的拓扑序S，满足：
  对任意有向边(u, v)，排序后u在v之前

- 拓扑序不唯一

有向图任意边的起点排在终点前面



提纲

问题定义

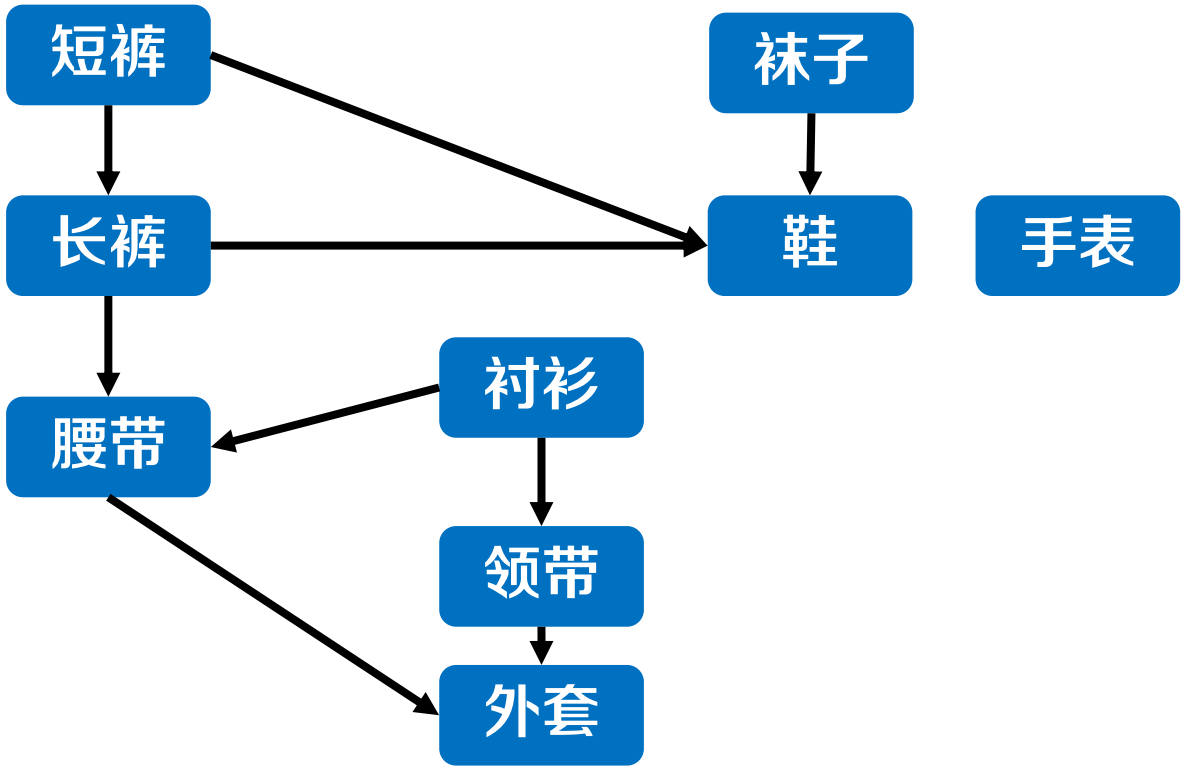
广度优先策略

深度优先策略

算法分析

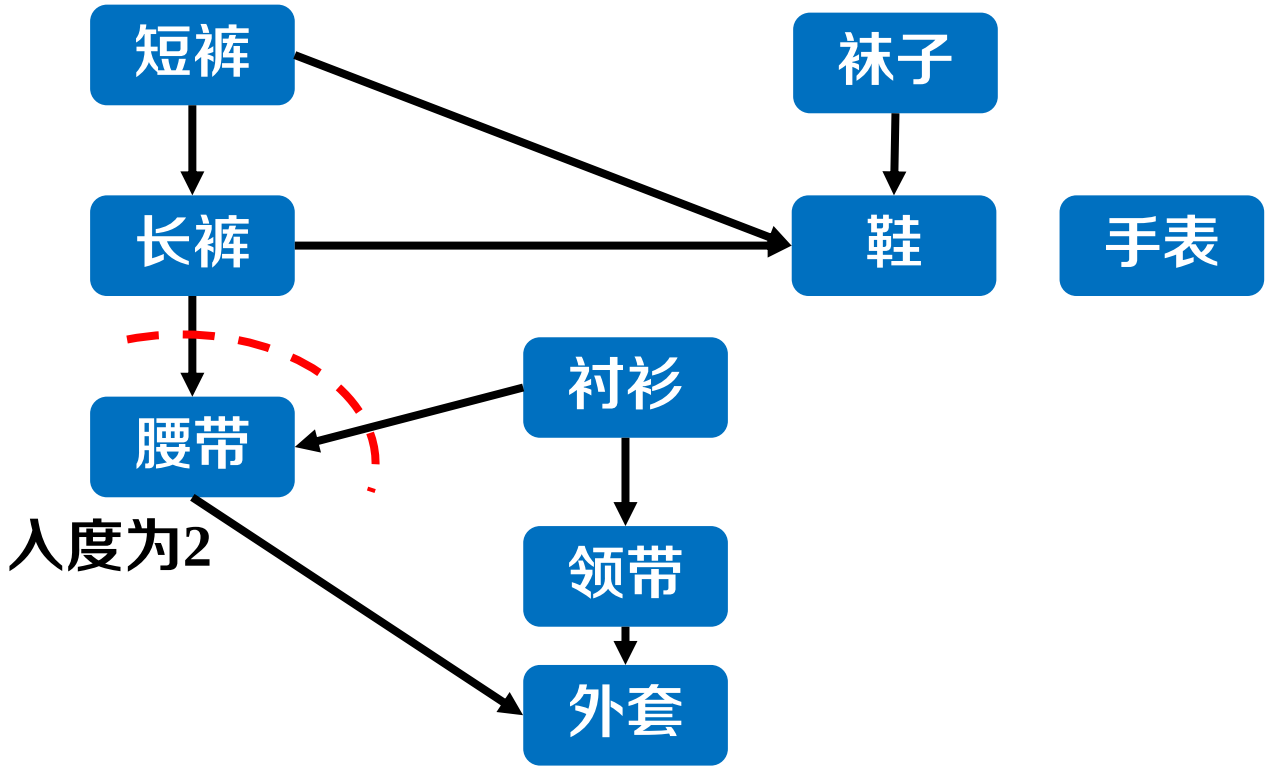
算法思想

- 有向图顶点的度分为入度和出度



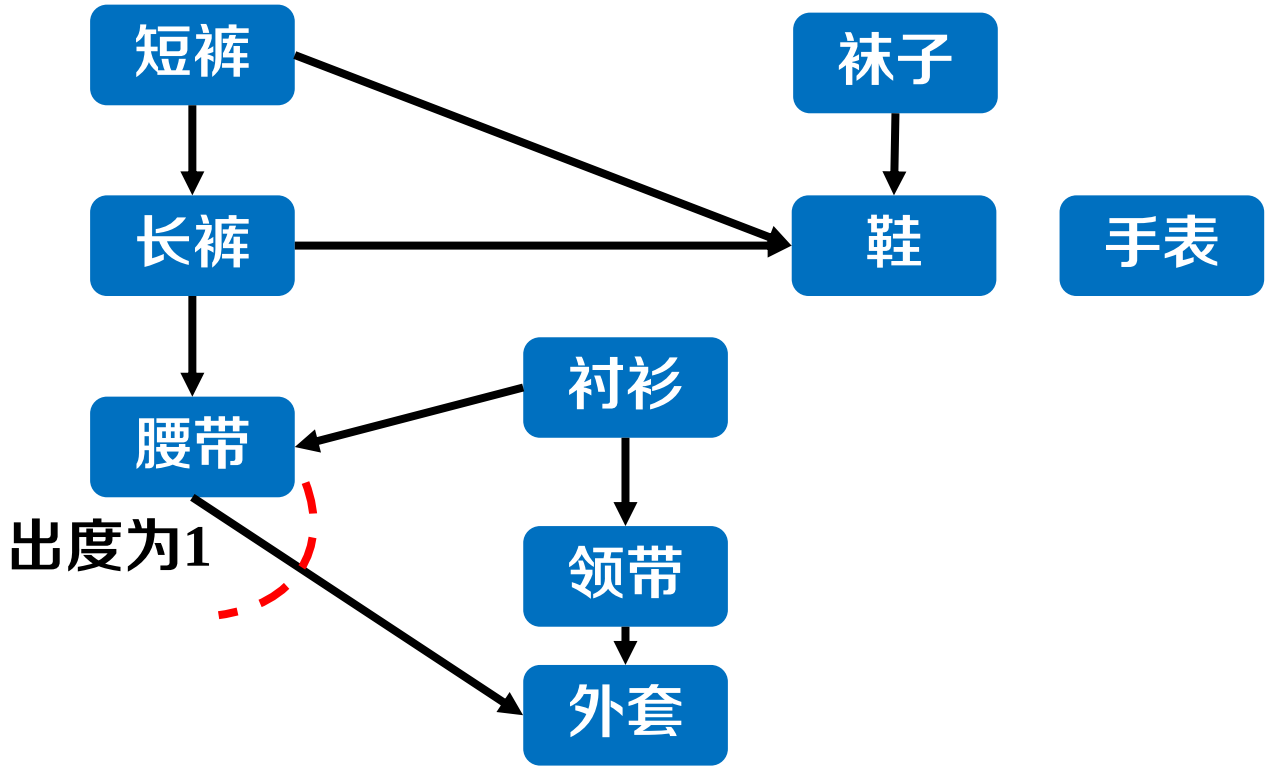
算法思想

- 有向图顶点的度分为入度和出度
  - 顶点u的入度：起点为u的边数



算法思想

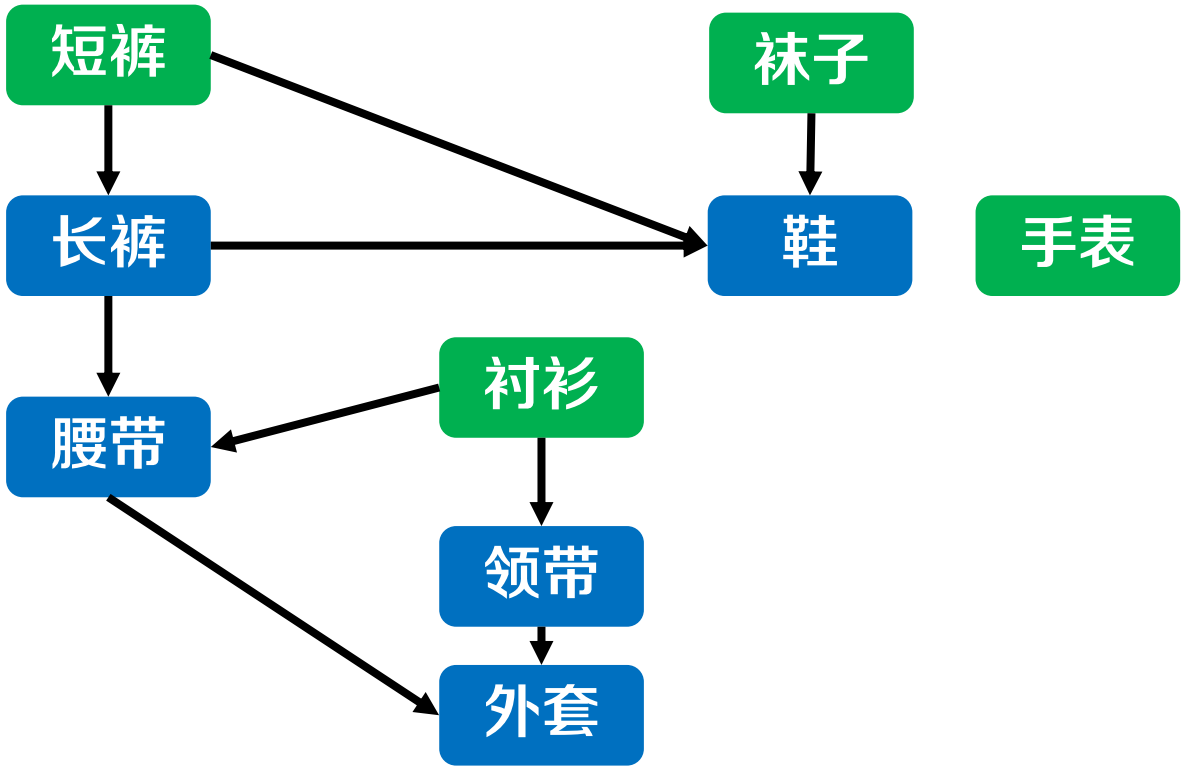
- 有向图顶点的度分为入度和出度
  - 顶点u的入度：起点为u的边数
  - 顶点u的出度：终点为u的边数



算法思想

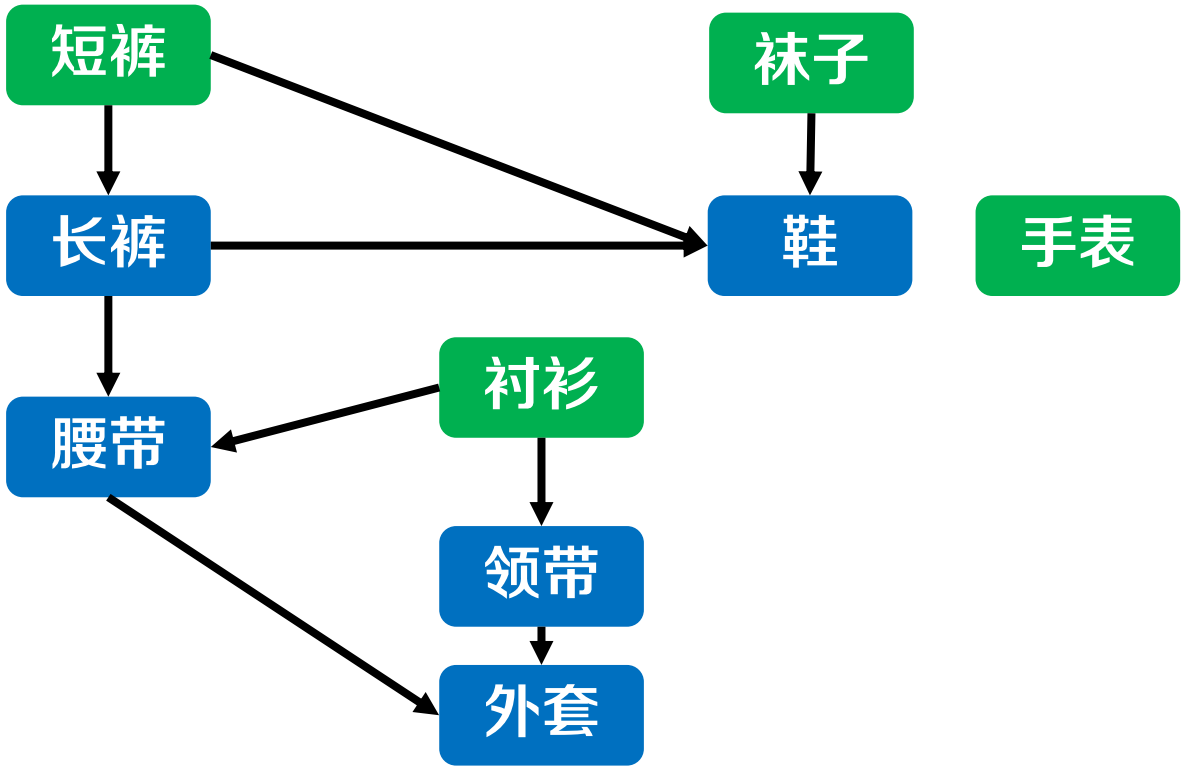
- 有向图顶点的度分为入度和出度
  - 顶点u的入度：起点为u的边数
  - 顶点u的出度：终点为u的边数

- 若顶点入度为0
  - 所对应事件无制约，可直接完成



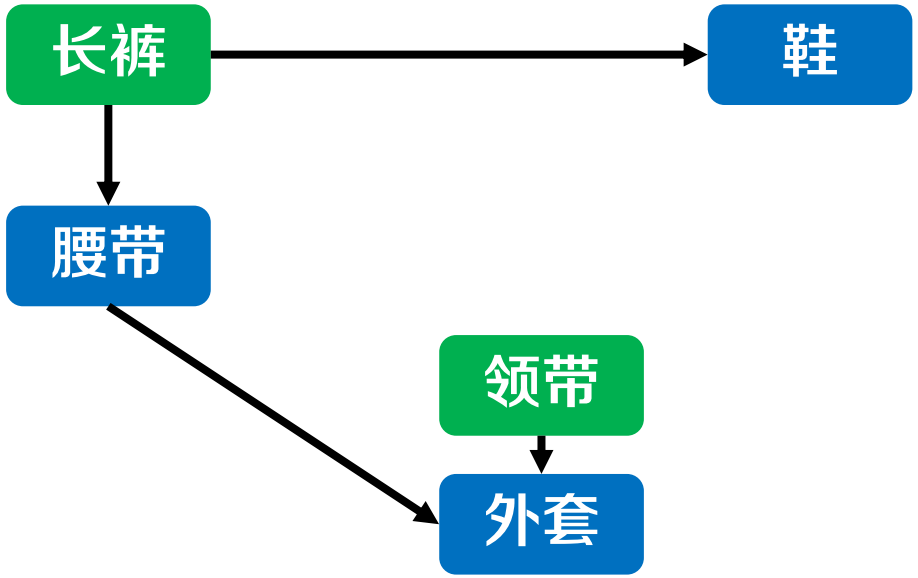
算法思想

- 完成入度为0点对应的事件



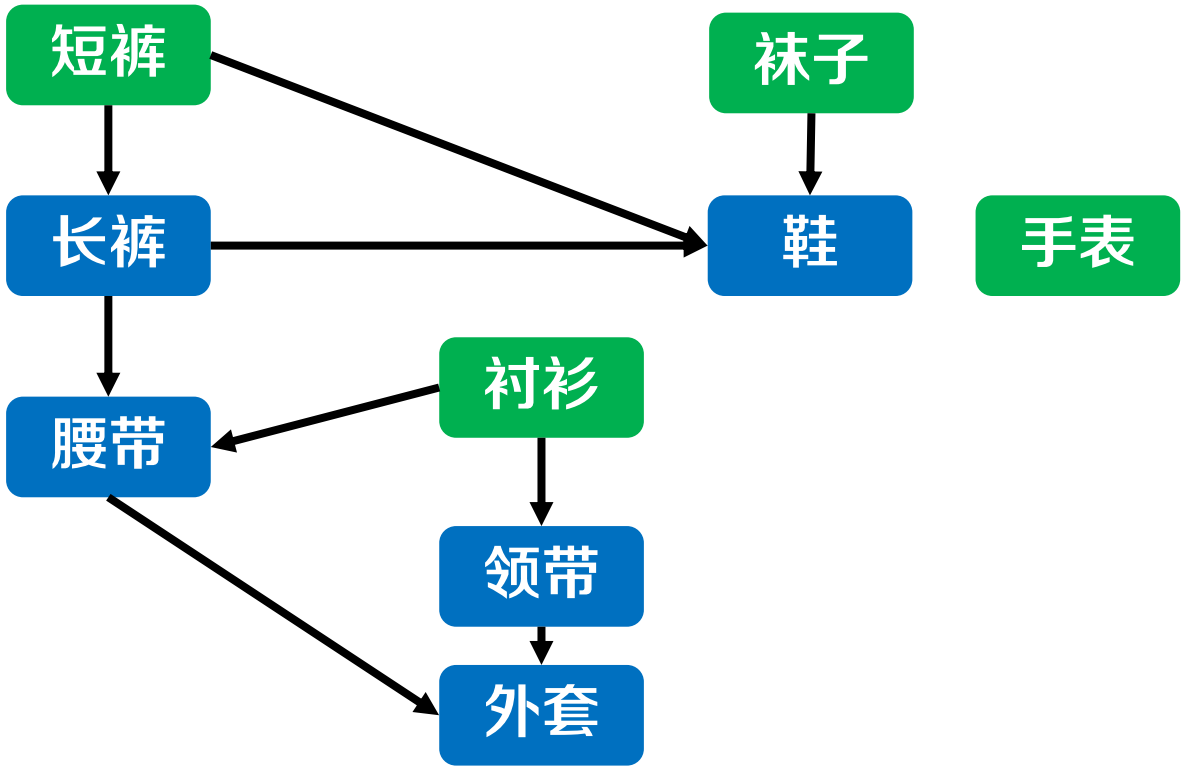
算法思想

- 完成入度为0点对应的事件
- 删除完成事件，产生新的入度为0点，继续完成



算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成



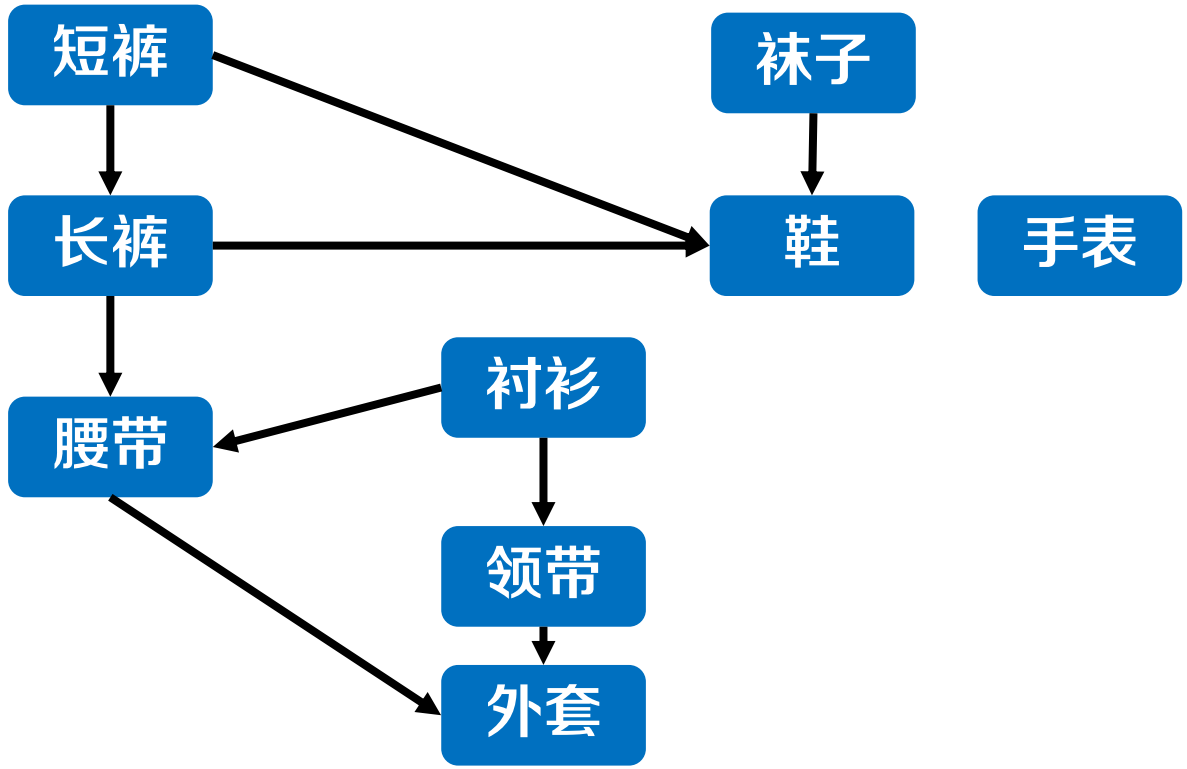
队列 记录入度为0度点

短裤 | 袜子 | 手表 | 衬衫

拓扑序 记录已完成事件

算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成



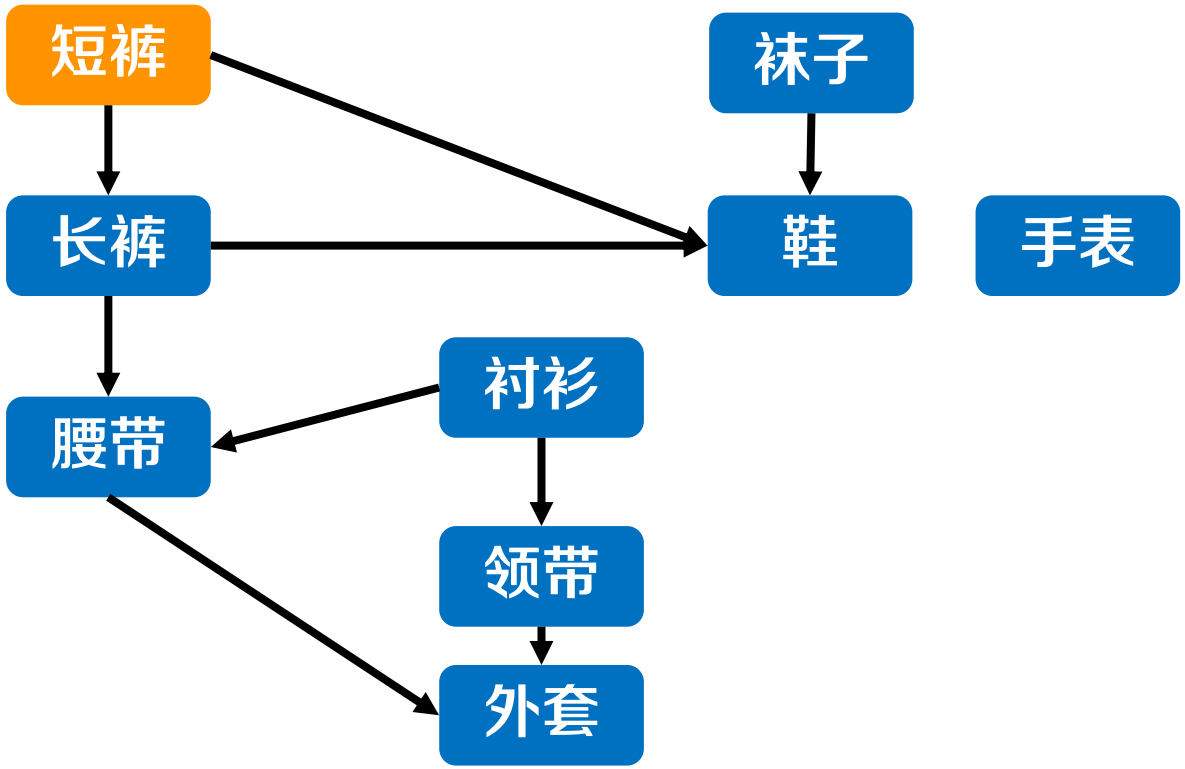
队列 记录入度为0度点

短裤 | 袜子 | 手表 | 衬衫

拓扑序 记录已完成事件

算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

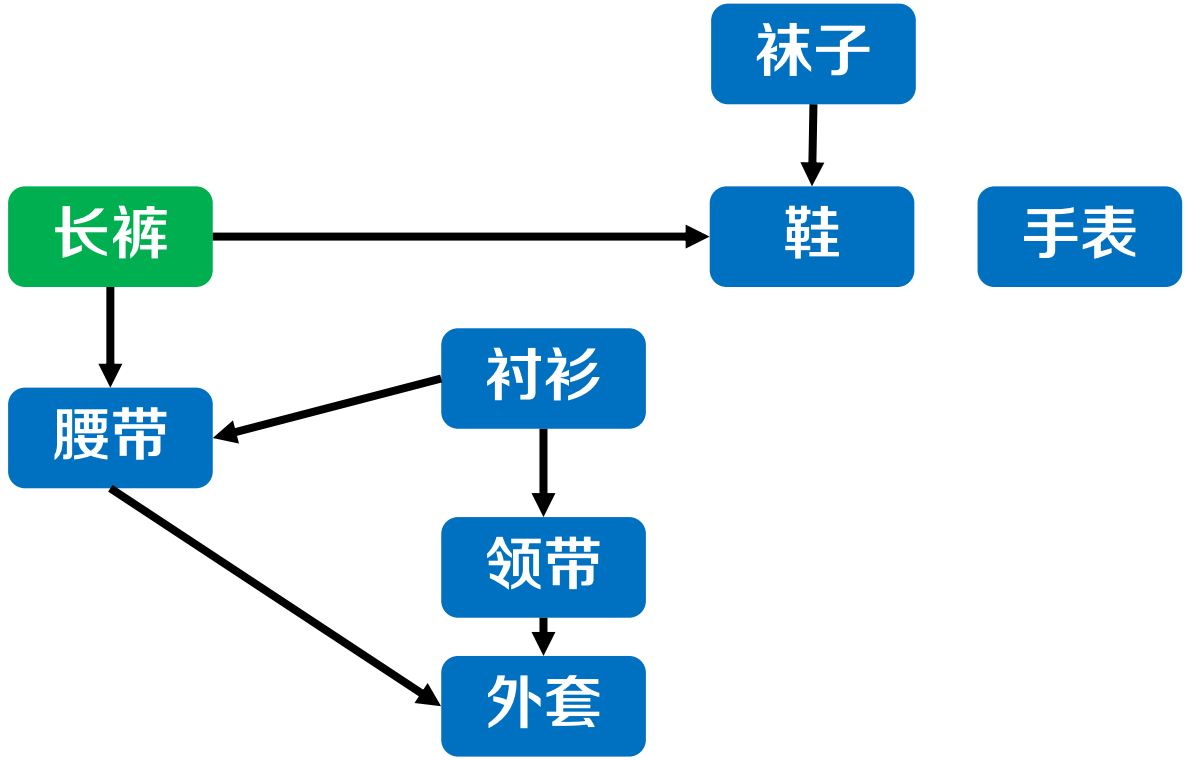


队列 记录入度为0度点

拓扑序 记录已完成事件

算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成



队列 记录入度为0度点

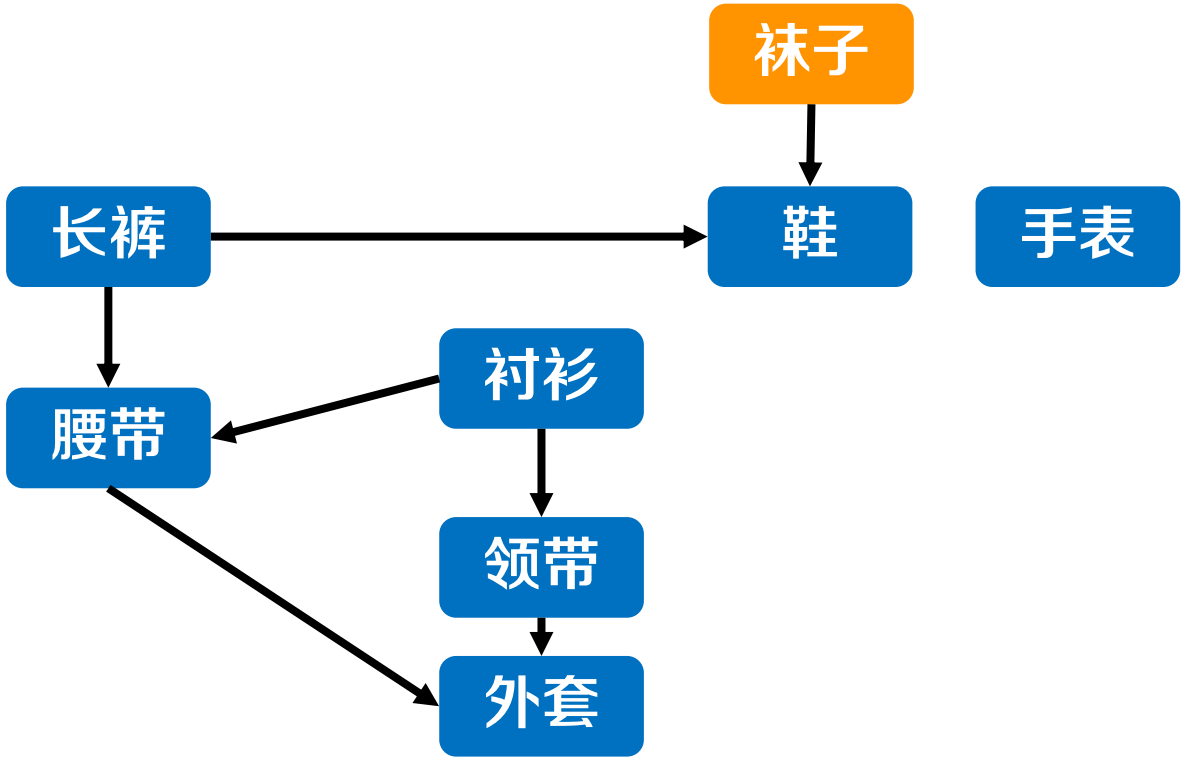
袜子 手表 衬衫 长裤

拓扑序 记录已完成事件

短裤

算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成



队列 记录入度为0度点

袜子 手表 衬衫 长裤

拓扑序 记录已完成事件

短裤

算法实例

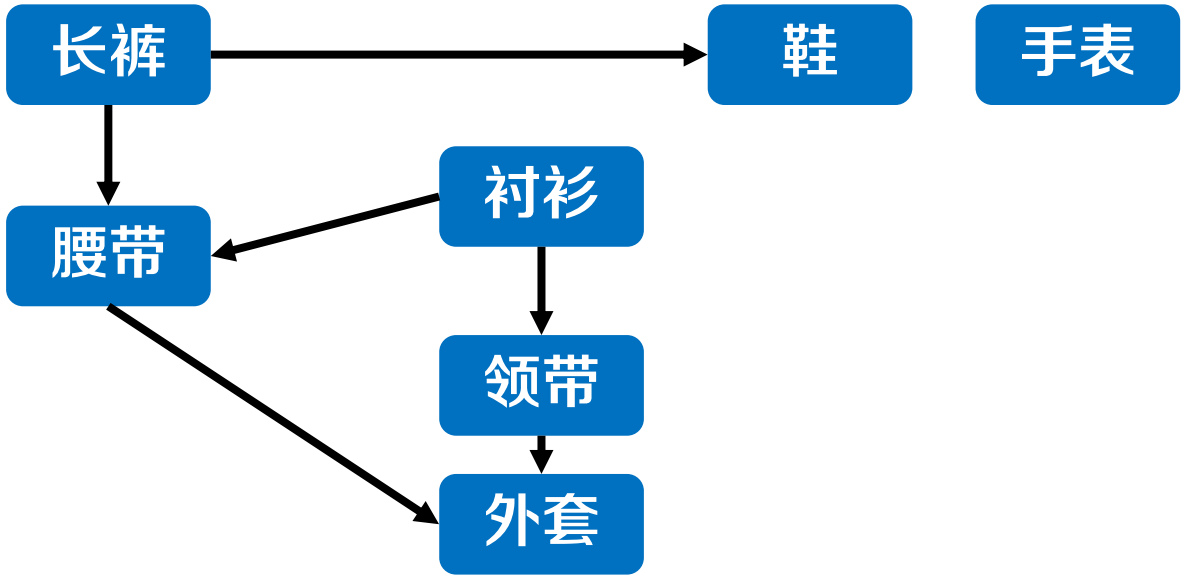
- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

队列 记录入度为0度点

手表 | 衬衫 | 长裤

拓扑序 记录已完成事件

短裤 | 袜子



算法实例

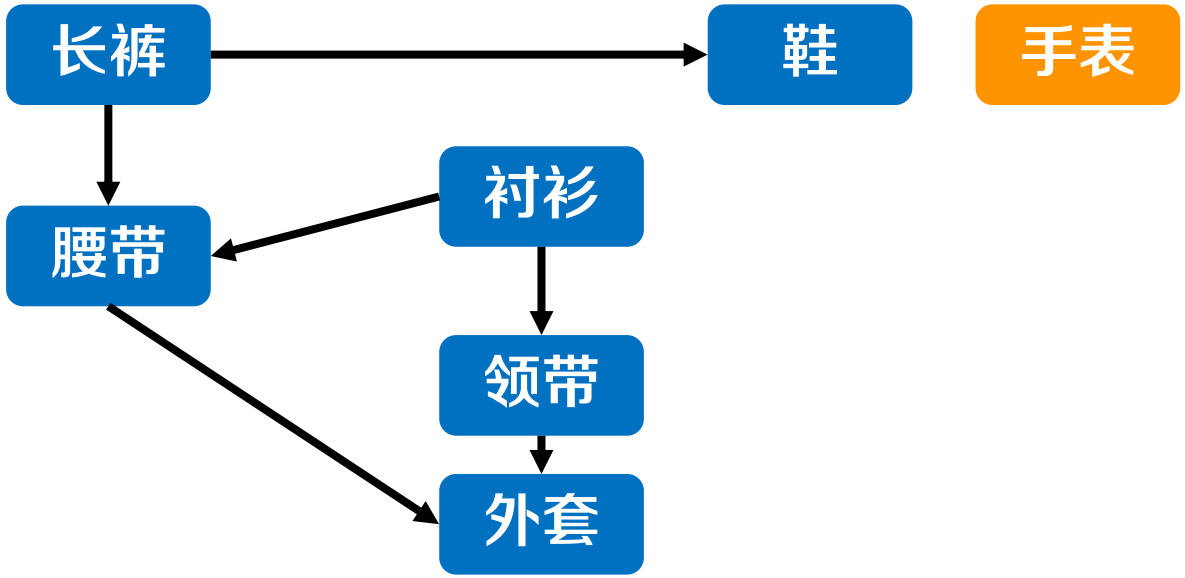
- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

队列 记录入度为0度点

手表 | 衬衫 | 长裤

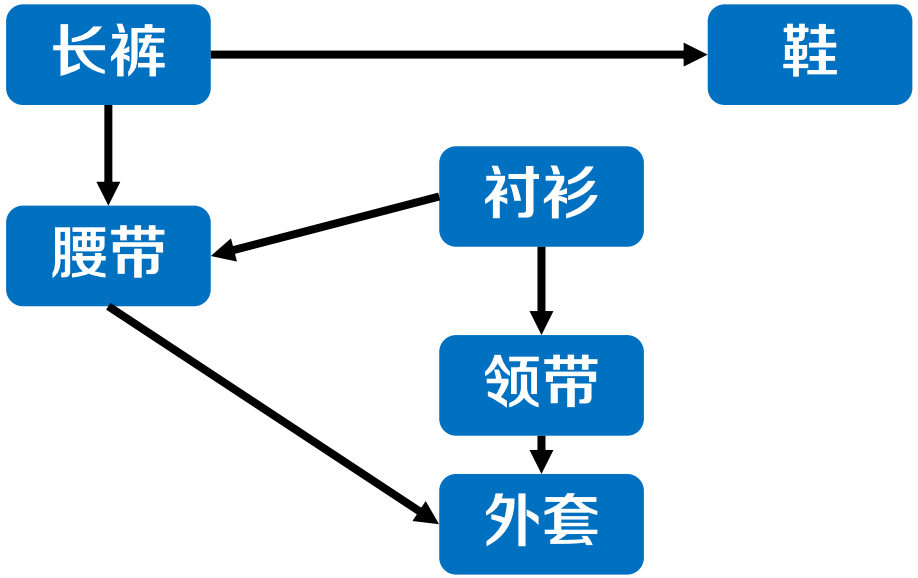
拓扑序 记录已完成事件

短裤 | 袜子



算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成



队列 记录入度为0度点

衬衫 长裤

拓扑序 记录已完成事件

短裤 袜子 手表

算法实例

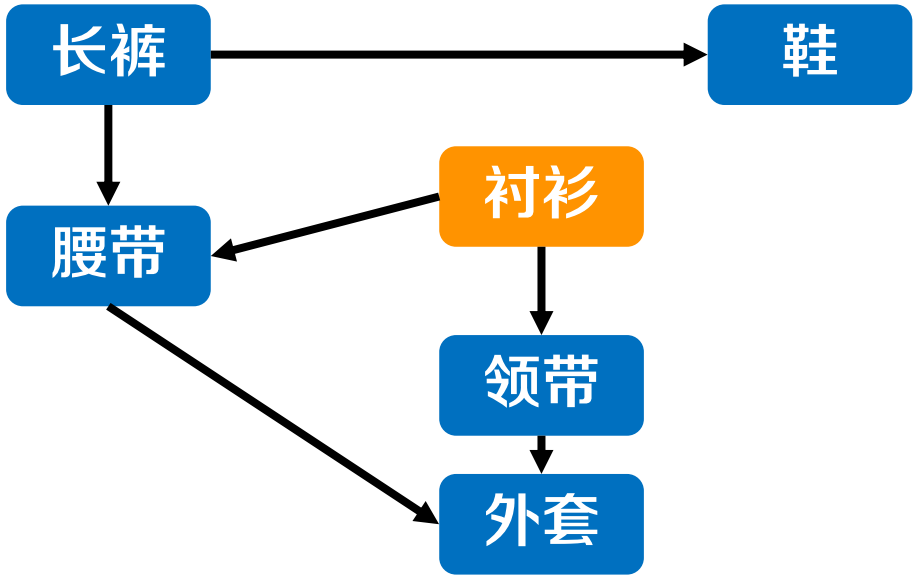
- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

队列 记录入度为0度点

衬衫 长裤

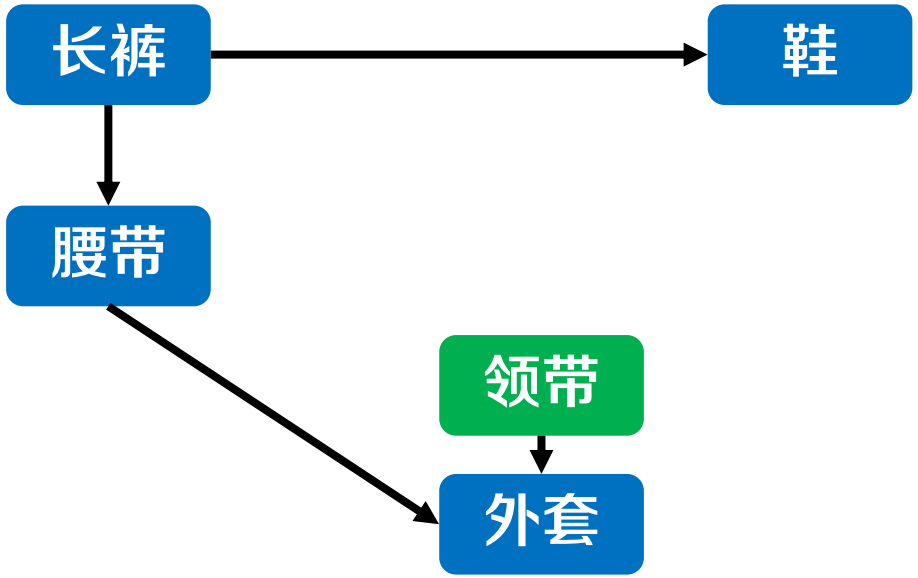
拓扑序 记录已完成事件

短裤 袜子 手表



算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

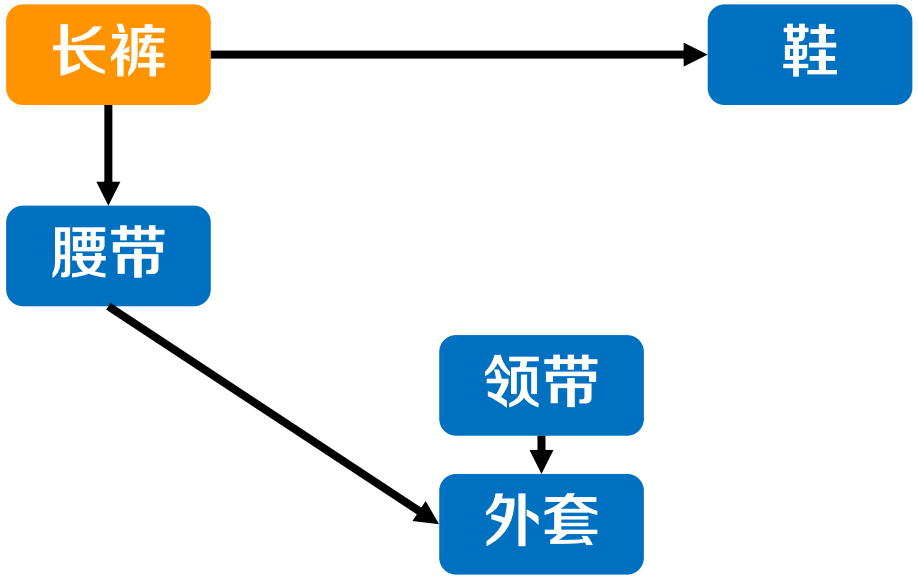


队列 记录入度为0度点

拓扑序 记录已完成事件

算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成



队列 记录入度为0度点

长裤 领带

拓扑序 记录已完成事件

短裤 袜子 手表 衬衫

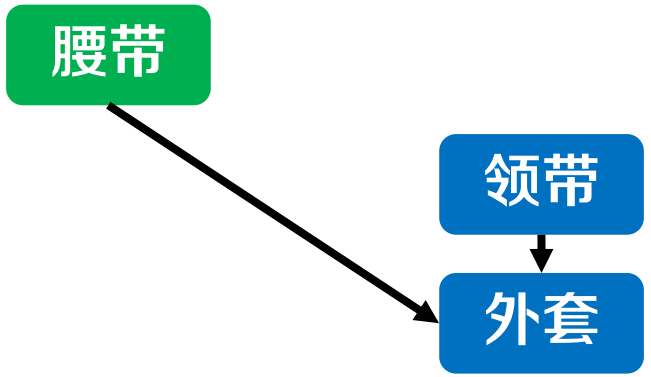
算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

鞋

队列 记录入度为0度点

领带 鞋 腰带

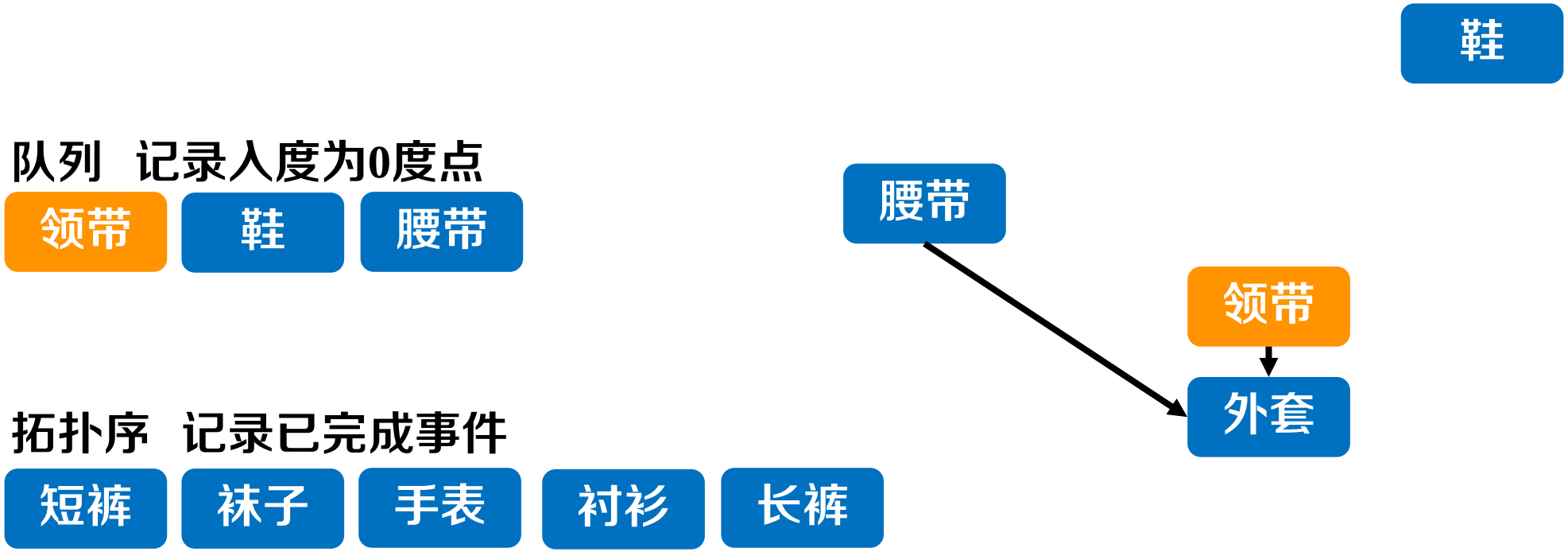


拓扑序 记录已完成事件

短裤 袜子 手表 衬衫 长裤

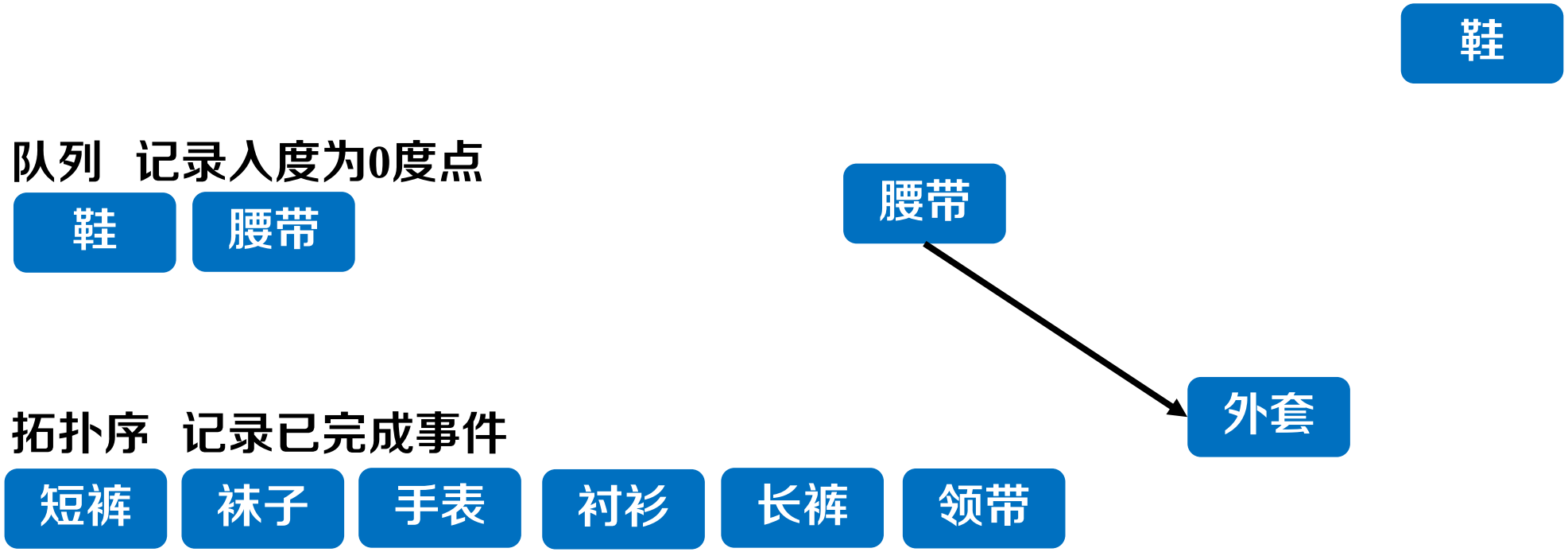
算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成



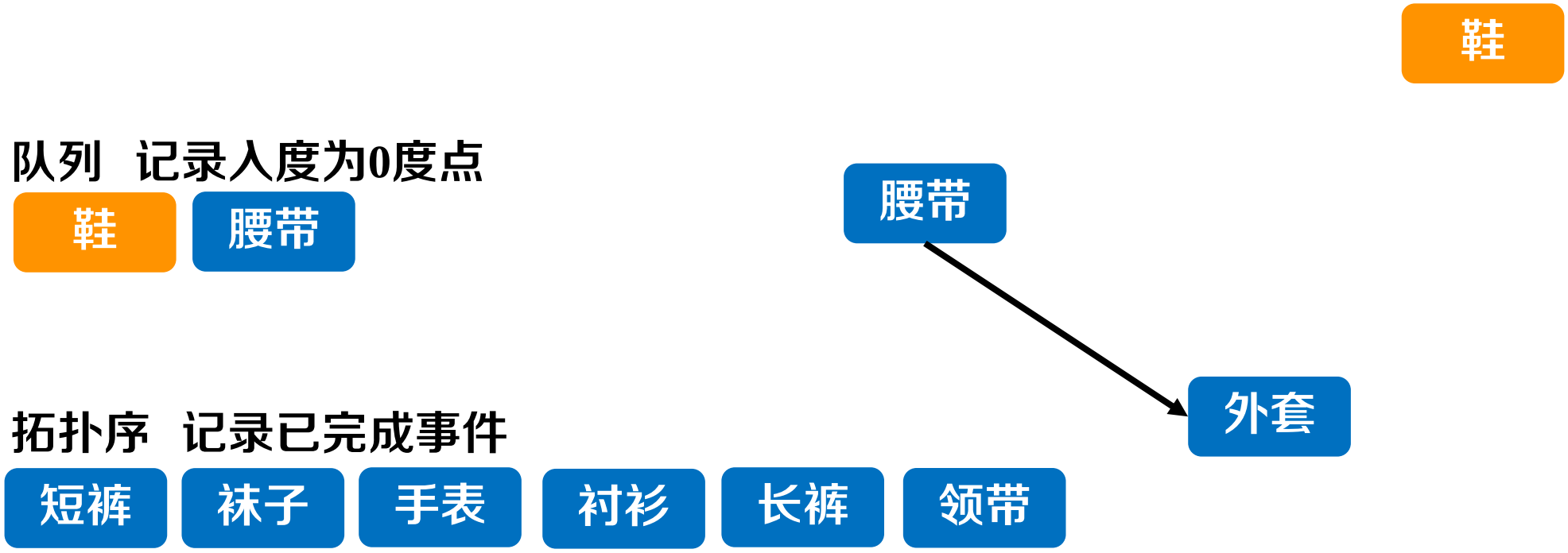
算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成



算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成



算法实例

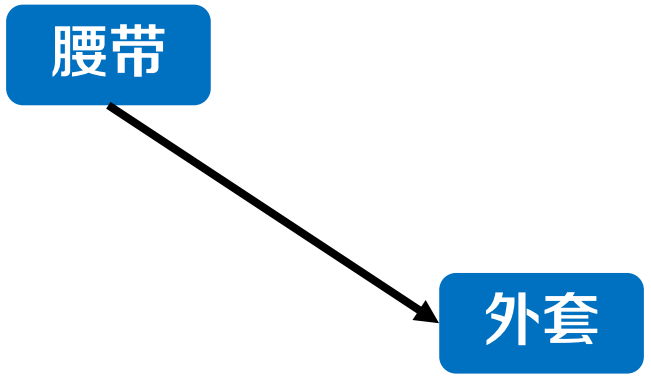
- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

队列 记录入度为0度点

腰带

拓扑序 记录已完成事件

短裤 袜子 手表 衬衫 长裤 领带 鞋



算法实例

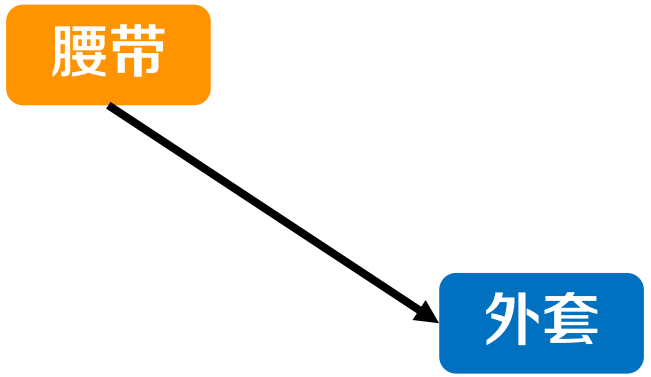
- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

队列 记录入度为0度点

腰带

拓扑序 记录已完成事件

短裤 袜子 手表 衬衫 长裤 领带 鞋



算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

队列 记录入度为0度点

外套

拓扑序 记录已完成事件

外套

短裤 袜子 手表 衬衫 长裤 领带 鞋 腰带

算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

队列 记录入度为0度点

外套

拓扑序 记录已完成事件

外套

短裤 袜子 手表 衬衫 长裤 领带 鞋 腰带

算法实例

- 算法思想
  - 完成入度为0点对应的事件
  - 删除完成事件，产生新的入度为0点，继续完成

队列 记录入度为0度点

拓扑序 记录已完成事件

短裤 袜子 手表 衬衫 长裤 领带 鞋 腰带 外套

伪代码

- **Topological-Sort-BFS(G)**

```
输入: 图G
输出: 顶点拓扑序
初始化空队列Q
for v ∈ V do
    if v.in_degree = 0 then
        Q.Enqueue(v)
    end
end
while not Q.is_empty() do
    u ← Q.Dequeue()
    print u
    for v ∈ G.Adj(u) do
        v.in_degree ← v.in_degree − 1
        if v.in_degree = 0 then
            Q.Enqueue(v)
        end
    end
end
```

初始化

伪代码

- **Topological-Sort-BFS(G)**

```
输入: 图G
输出: 顶点拓扑序
初始化空队列Q
for v ∈ V do
    if v.in_degree = 0 then
        Q.Enqueue(v)
    end
end
while not Q.is_empty() do
    u ← Q.Dequeue()
    print u
    for v ∈ G.Adj(u) do
        v.in_degree ← v.in_degree − 1
        if v.in_degree = 0 then
            Q.Enqueue(v)
        end
    end
end
```

入度为0，加入队列

伪代码

- **Topological-Sort-BFS(G)**

```
输入: 图G
输出: 顶点拓扑序
初始化空队列Q
for v ∈ V do
    if v.in_degree = 0 then
        Q.Enqueue(v)
    end
end
while not Q.is_empty() do
    u ← Q.Dequeue()
    print u
    for v ∈ G.Adj(u) do
        v.in_degree ← v.in_degree − 1
        if v.in_degree = 0 then
            Q.Enqueue(v)
        end
    end
end
```

完成事件

伪代码

- **Topological-Sort-BFS(G)**

```
输入: 图G
输出: 顶点拓扑序
初始化空队列Q
for v ∈ V do
    if v.in_degree = 0 then
        Q.Enqueue(v)
    end
end
while not Q.is_empty() do
    u ← Q.Dequeue()
    print u
    for v ∈ G.Adj(u) do
        v.in_degree ← v.in_degree − 1
        if v.in_degree = 0 then
            Q.Enqueue(v)
        end
    end
end
```

删除事件，更新入度

伪代码

- **Topological-Sort-BFS(G)**

```
输入: 图G
输出: 顶点拓扑序
初始化空队列Q
for v ∈ V do
    if v.in_degree = 0 then
        Q.Enqueue(v)
    end
end
while not Q.is_empty() do
    u ← Q.Dequeue()
    print u
    for v ∈ G.Adj(u) do
        v.in_degree ← v.in_degree − 1
        if v.in_degree = 0 then
            Q.Enqueue(v)
        end
    end
end
```

入度为0，加入队列

复杂度分析

- **Topological-Sort-BFS(G)**

```
输入: 图G
输出: 顶点拓扑序
初始化空队列Q
for v ∈ V do
    if v.in_degree = 0 then
        Q.Enqueue(v)
    end
end
while not Q.is_empty() do
    u ← Q.Dequeue()
    print u
    for v ∈ G.Adj(u) do
        v.in_degree ← v.in_degree − 1
        if v.in_degree = 0 then
            Q.Enqueue(v)
        end
    end
end
```

$O(|V|)$

$\sum_{v \in V} deg(v) = O(|E|)$

$O(\sum_{v \in V}(1 + |Adj[v]|)) = O(|V| + |E|)$

时间复杂度：$O(|V| + |E|)$

提纲

问题定义

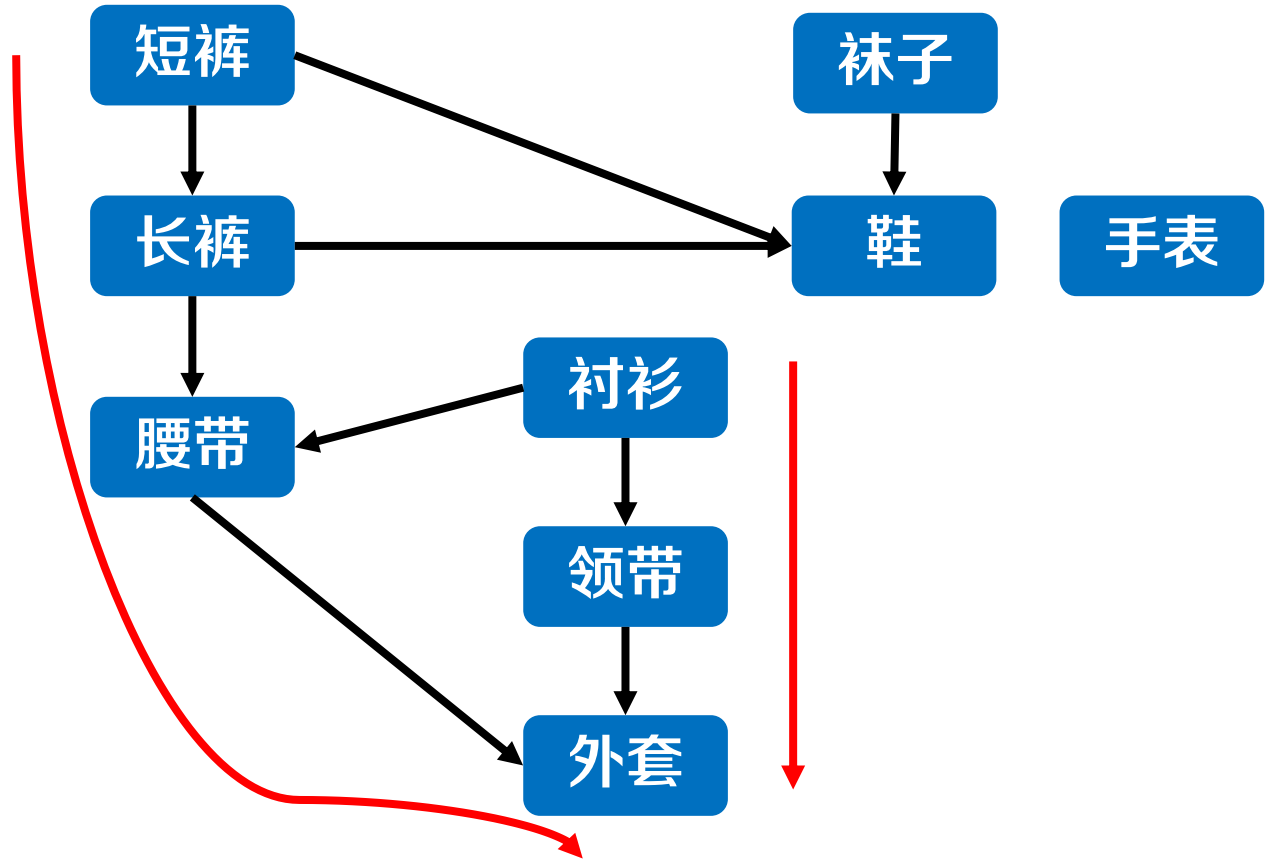
广度优先策略

深度优先策略

算法分析

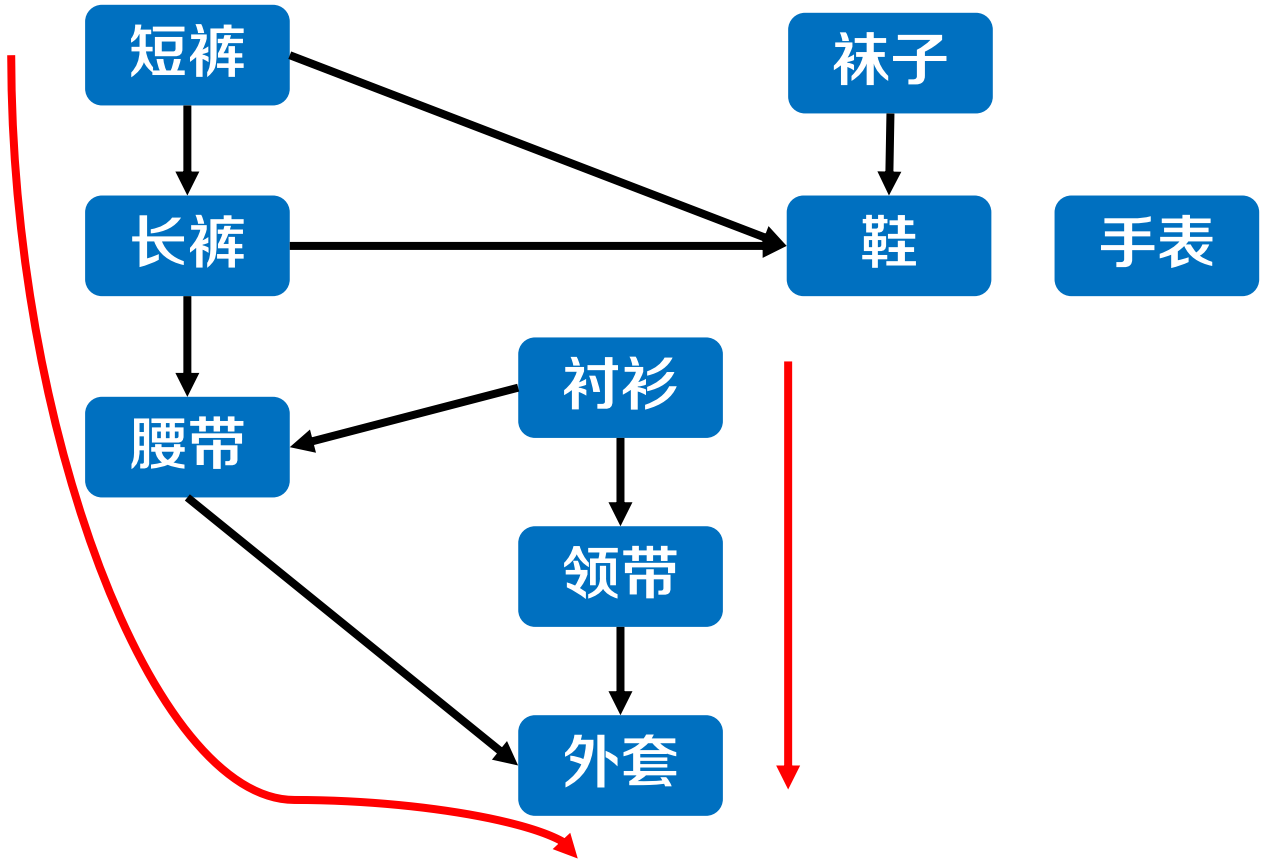
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后



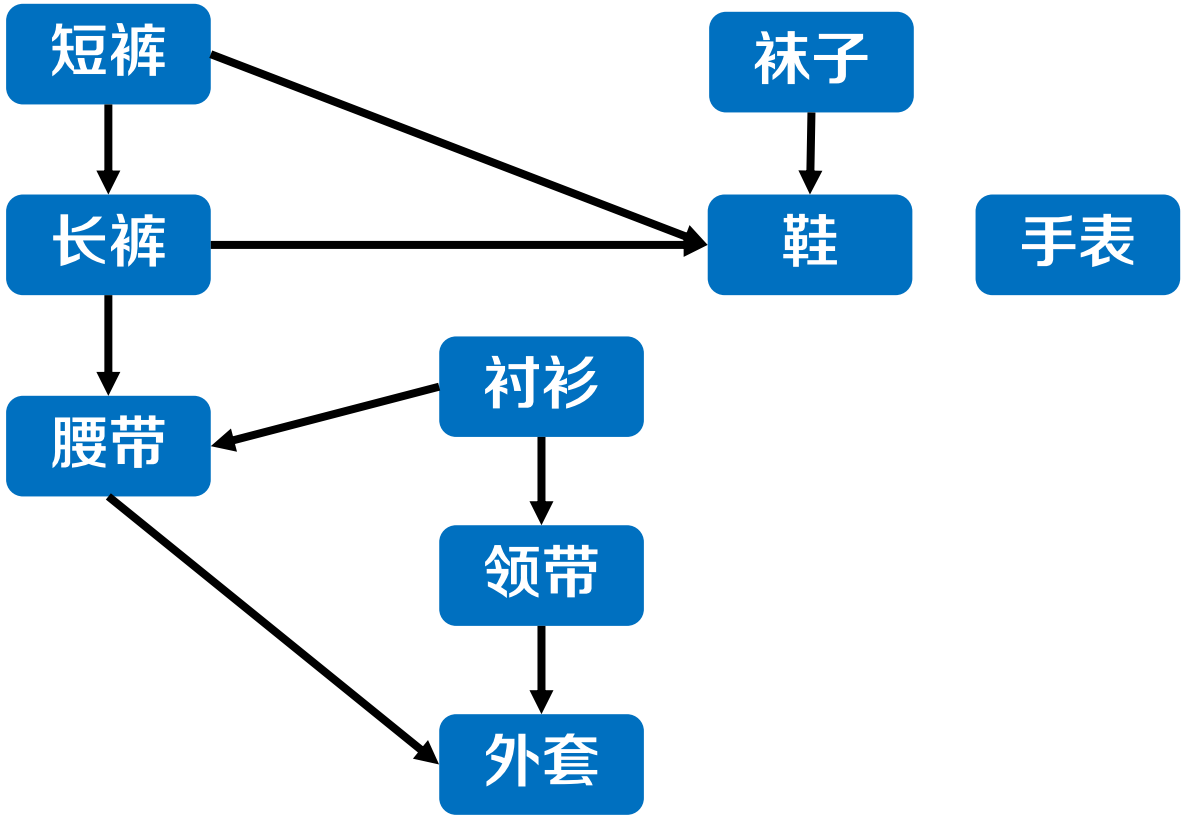
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚
    - 完成时刻越早



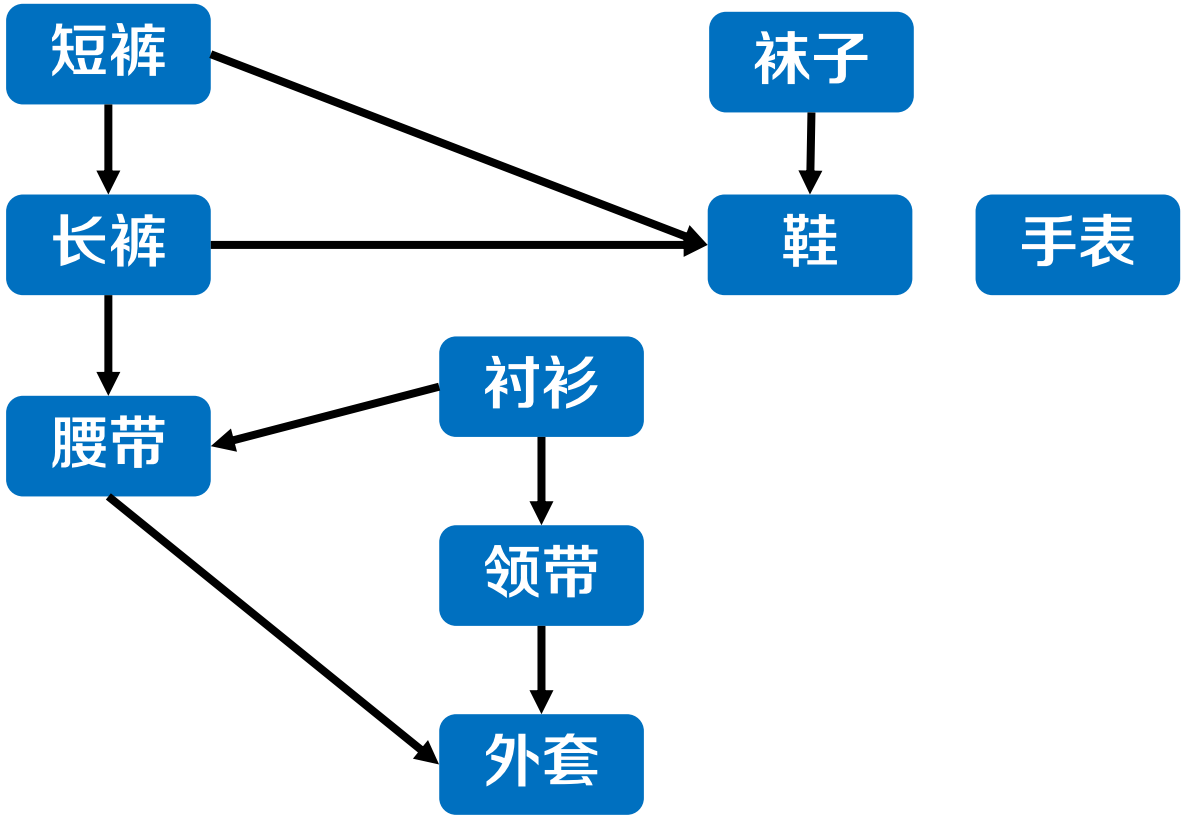
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序
    - 完成时刻越早：按完成时刻逆序



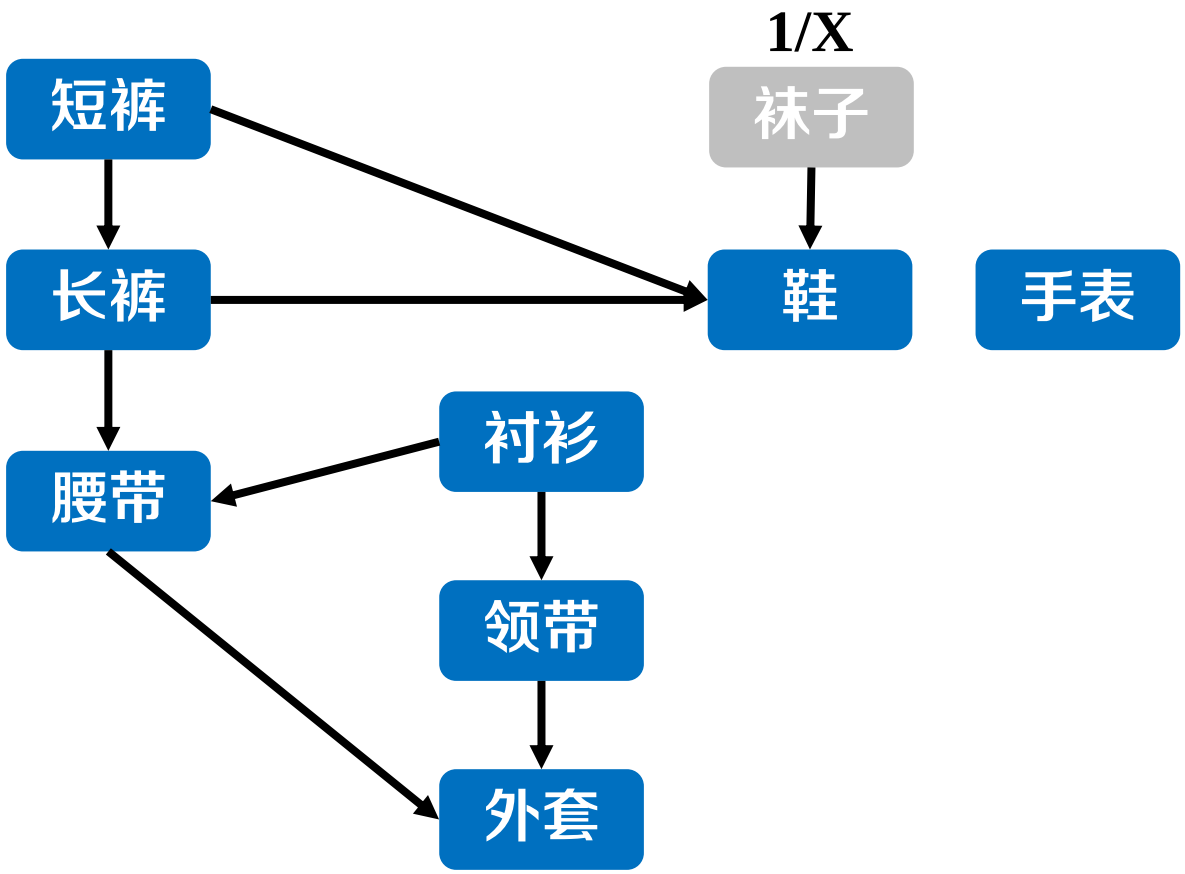
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序？
    - 完成时刻越早：按完成时刻逆序



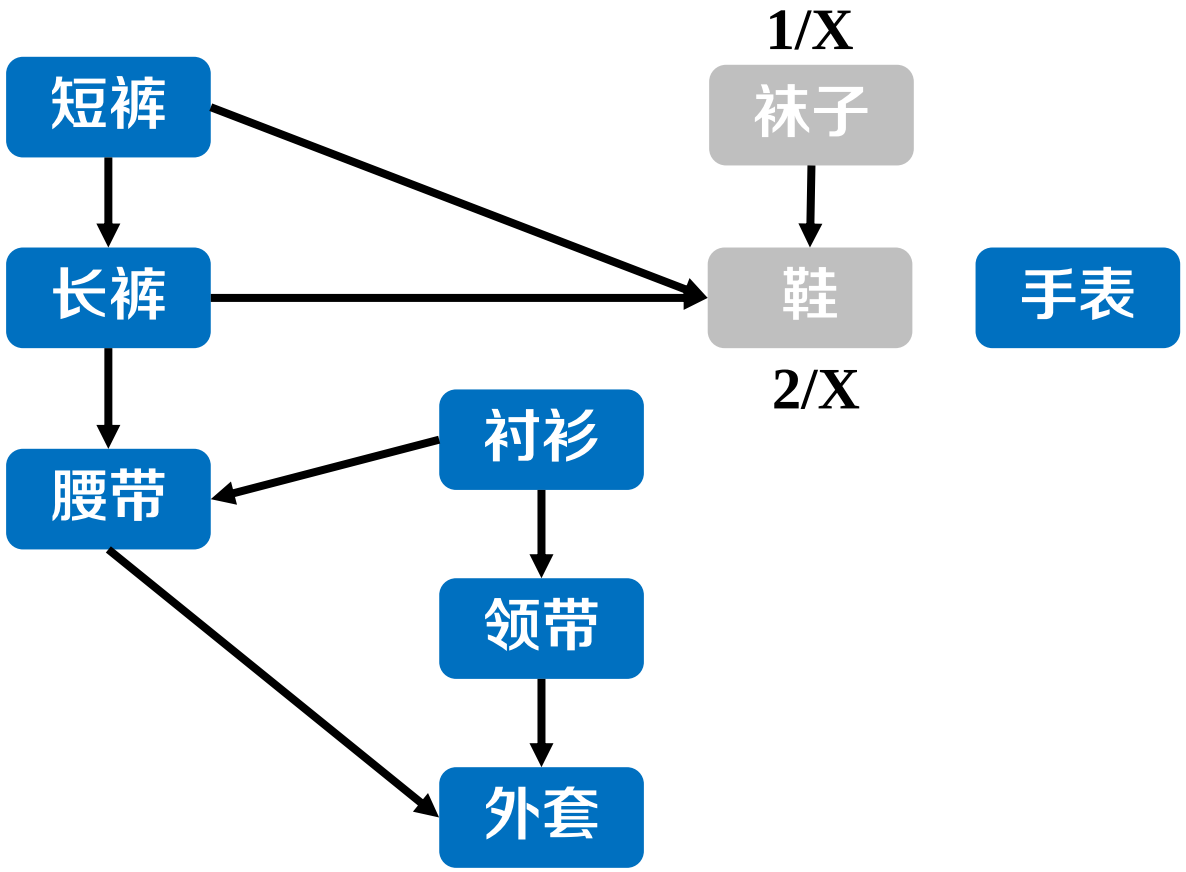
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序?
    - 完成时刻越早：按完成时刻逆序



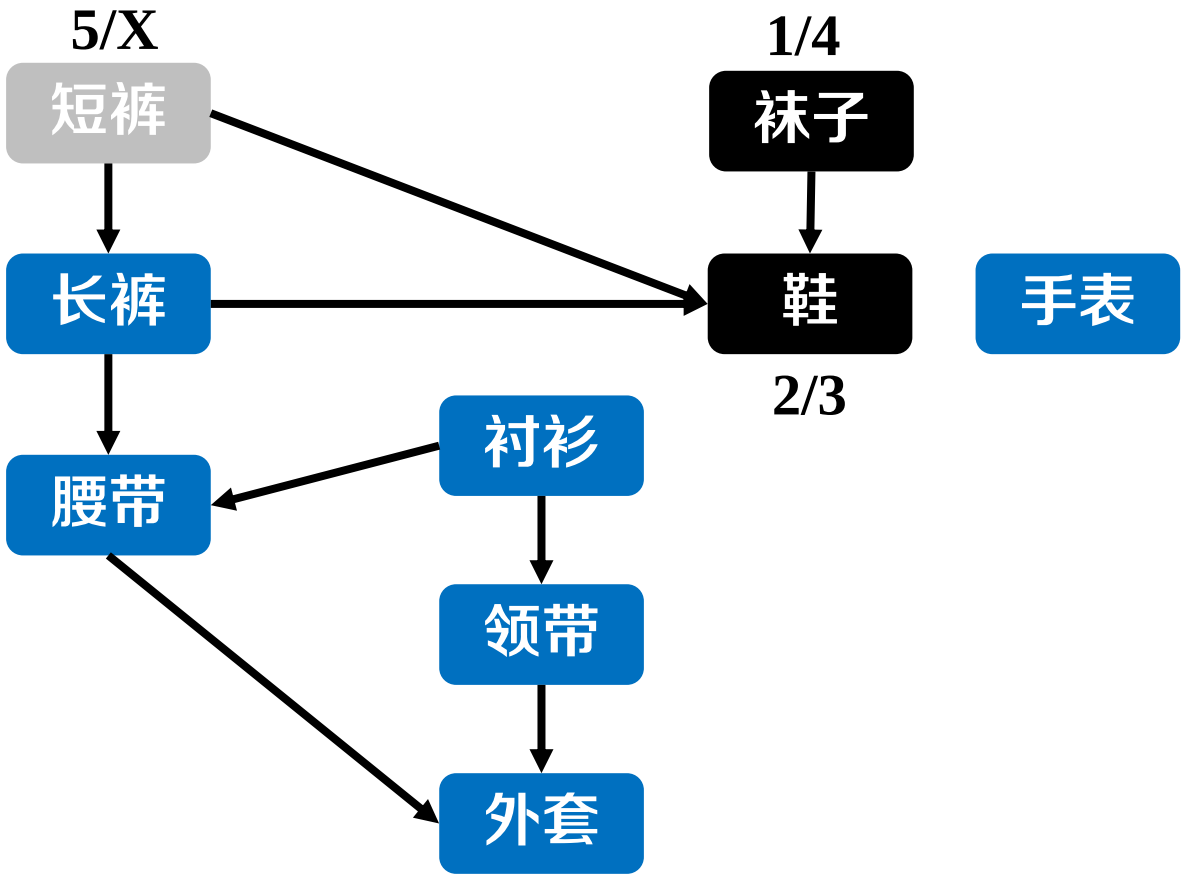
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序?
    - 完成时刻越早：按完成时刻逆序



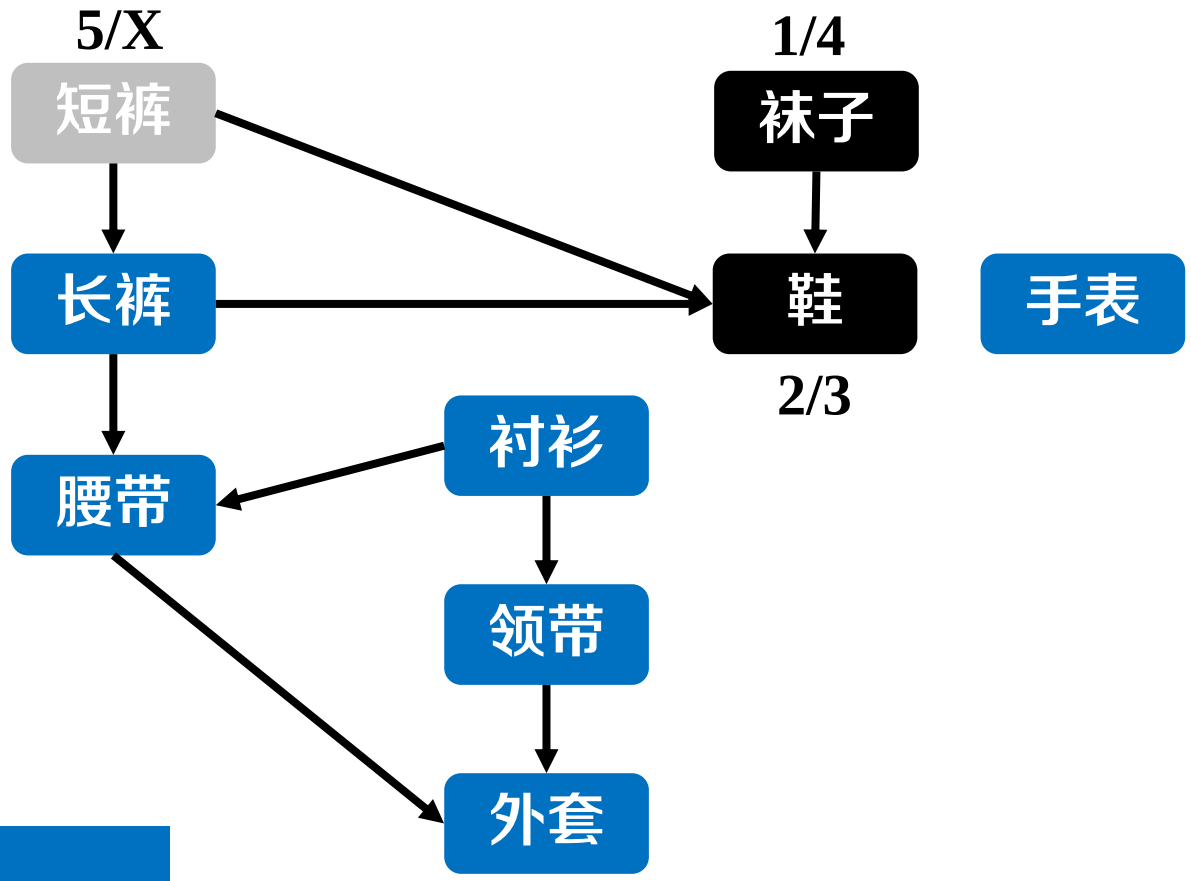
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序?
    - 完成时刻越早：按完成时刻逆序



算法思想

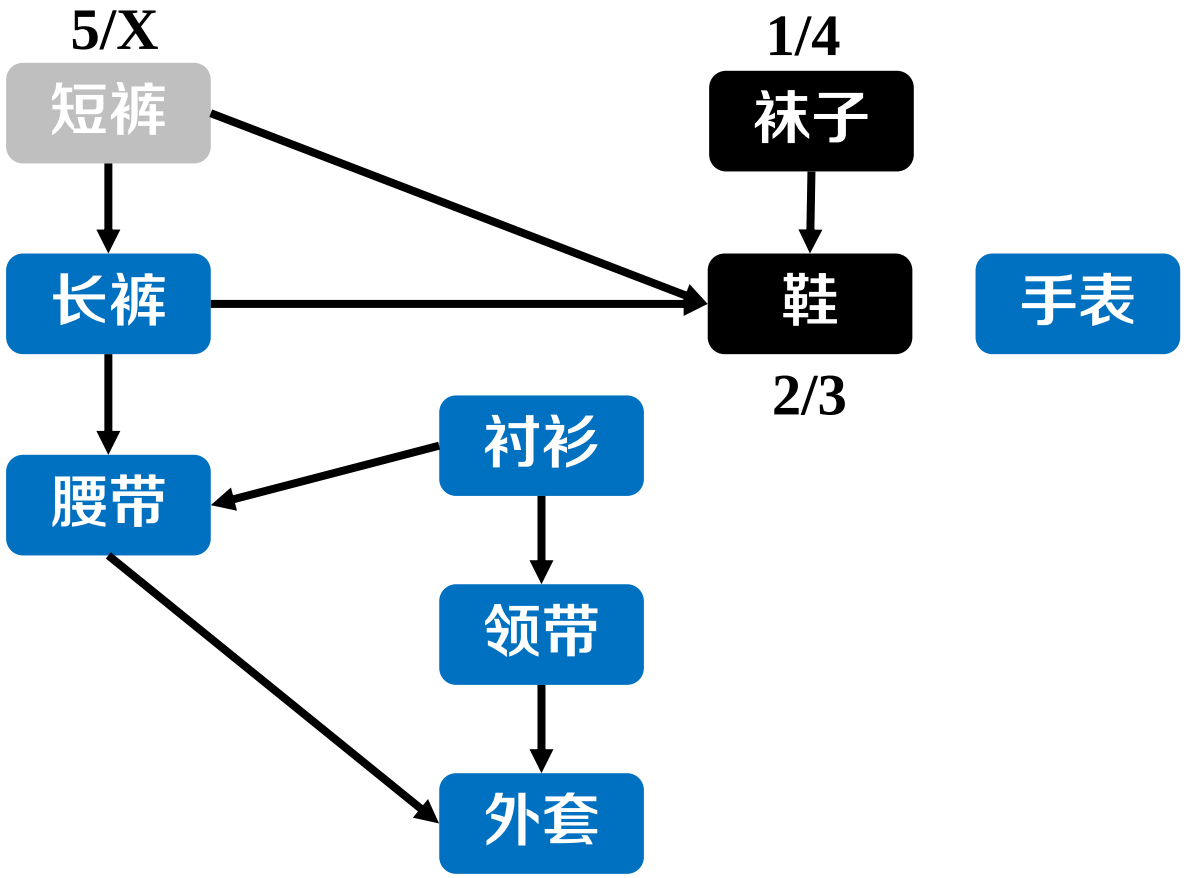
- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序
    - 完成时刻越早：按完成时刻逆序



若按发现时刻顺序执行，会先穿鞋后穿短裤

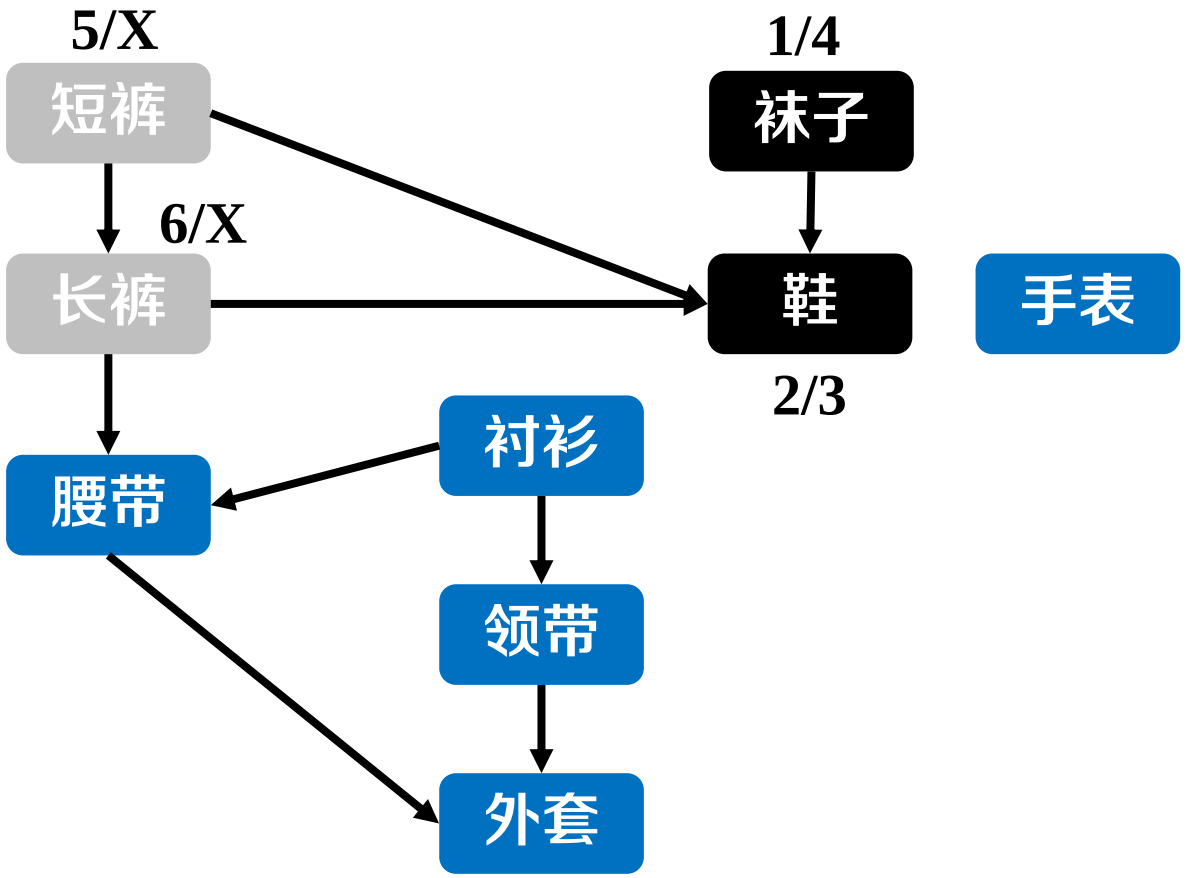
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序
    - 完成时刻越早：按完成时刻逆序?



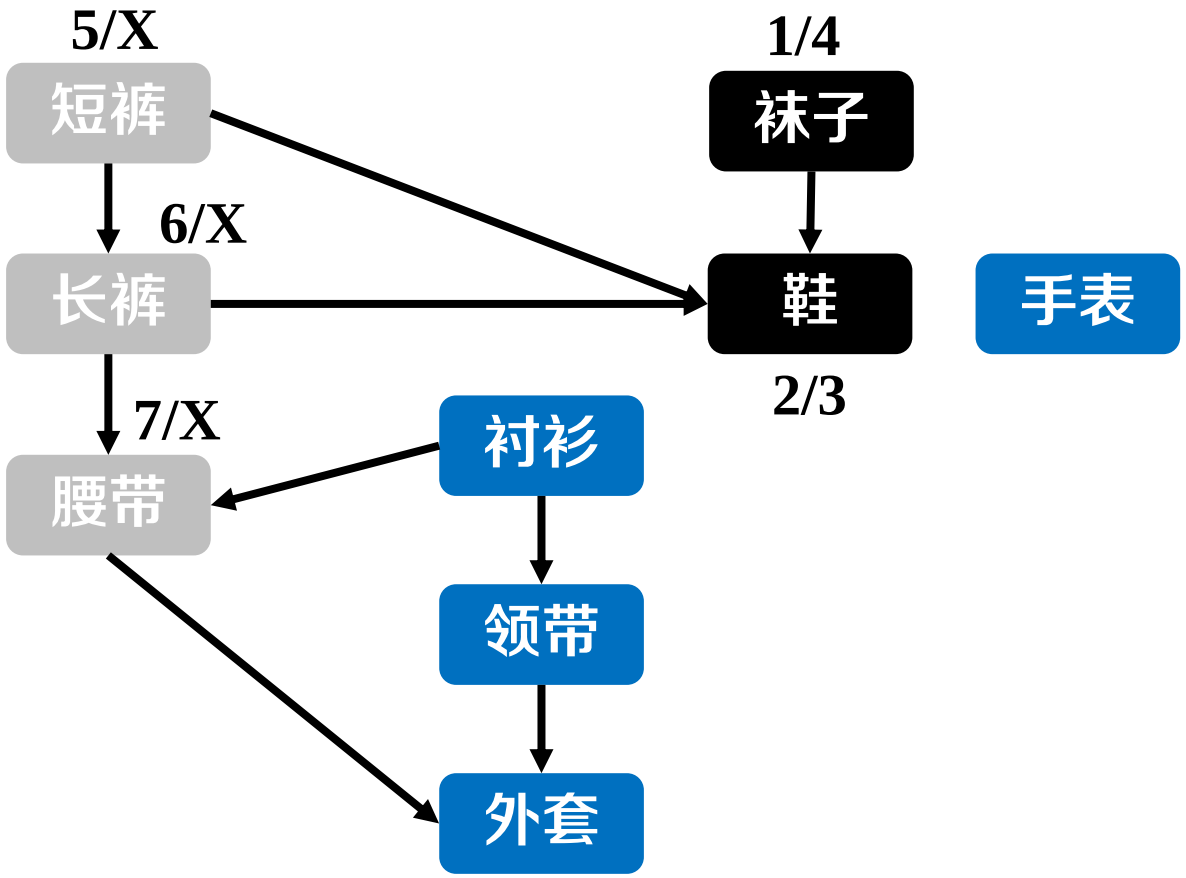
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序
    - 完成时刻越早：按完成时刻逆序？



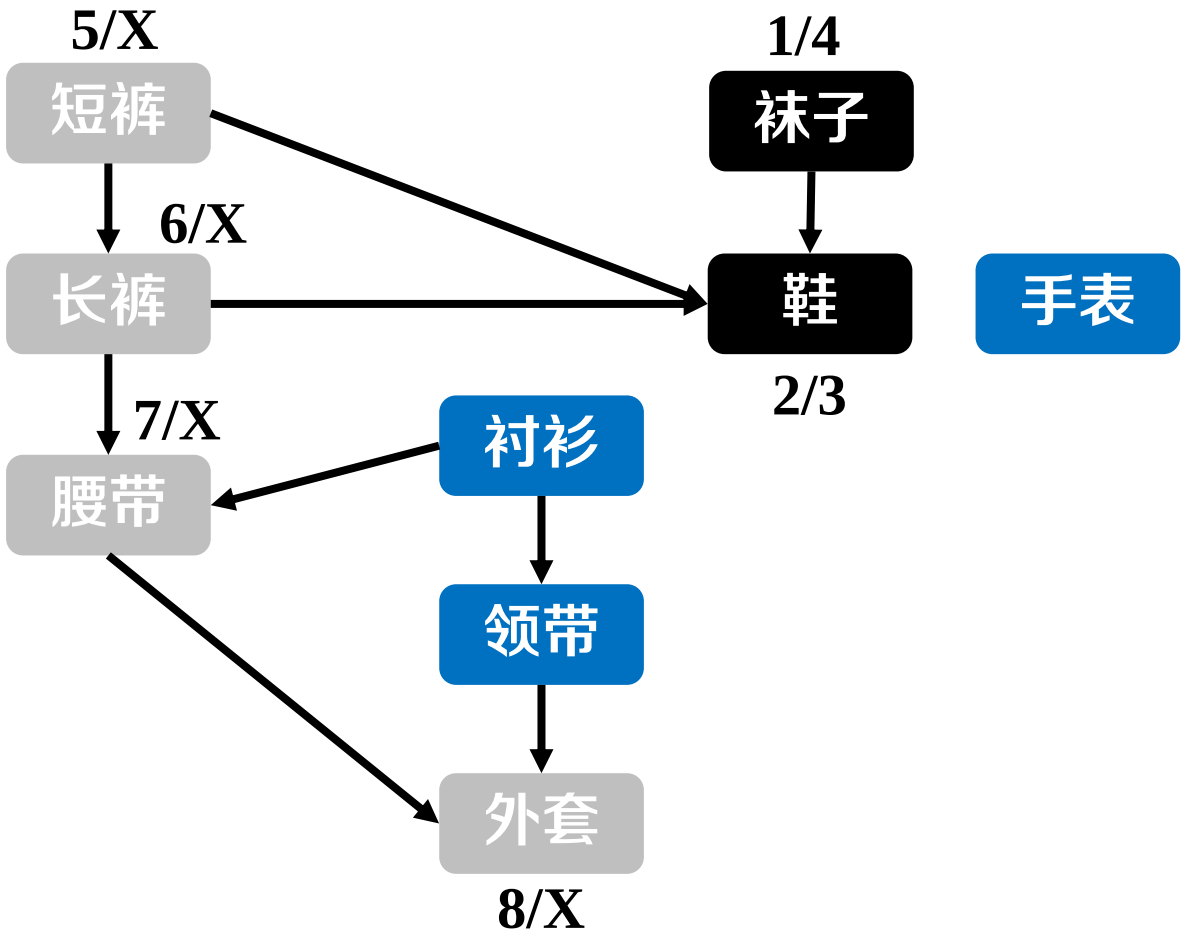
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序
    - 完成时刻越早：按完成时刻逆序？



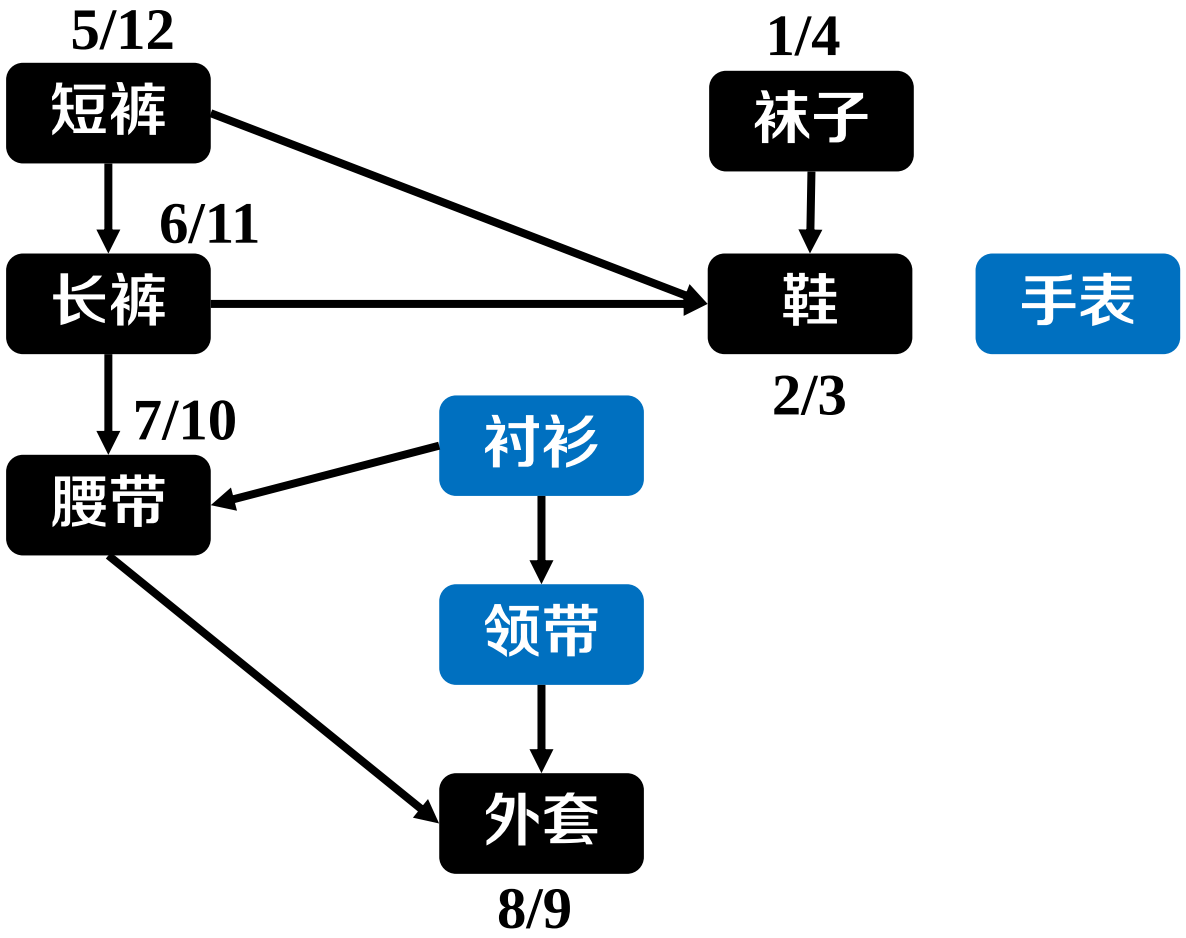
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序
    - 完成时刻越早：按完成时刻逆序



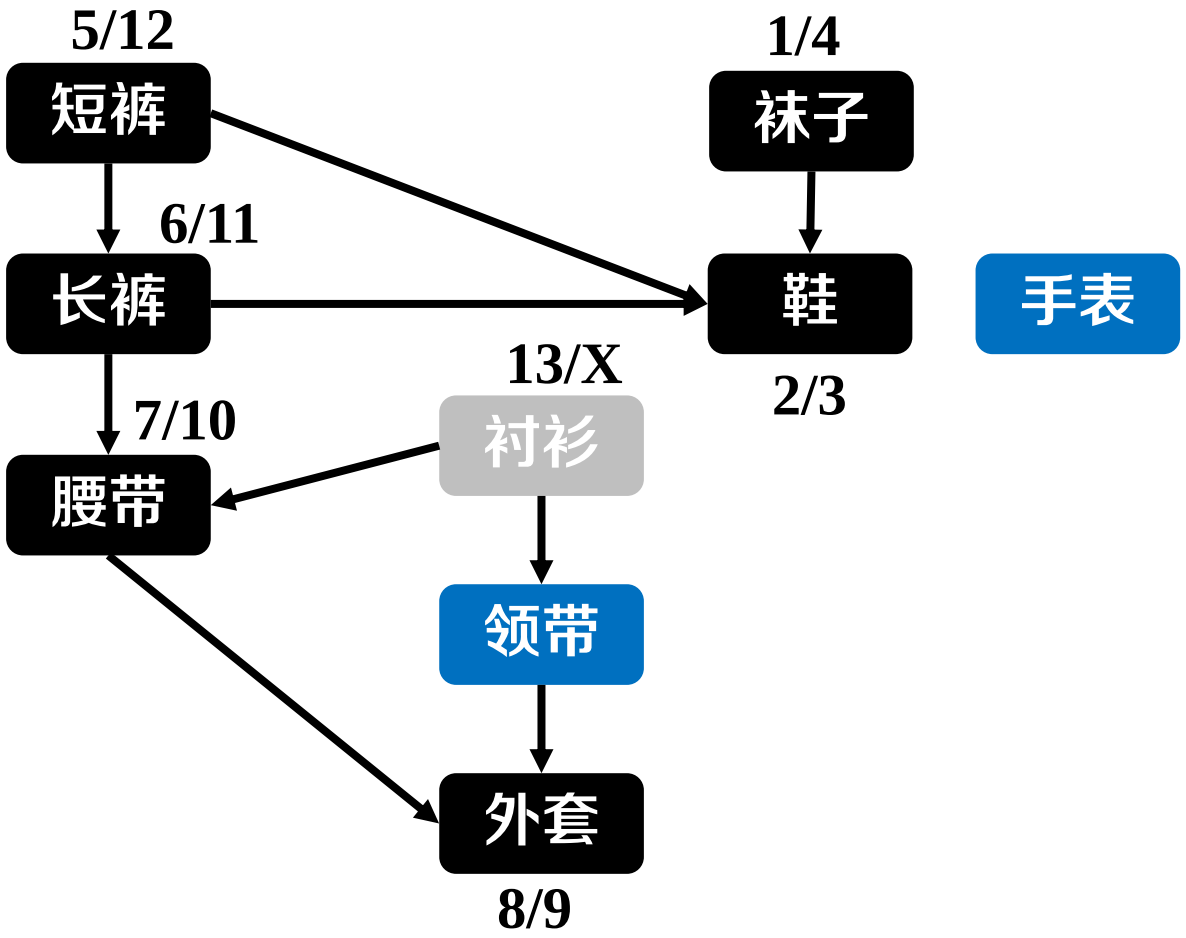
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序 ✗
    - 完成时刻越早：按完成时刻逆序？



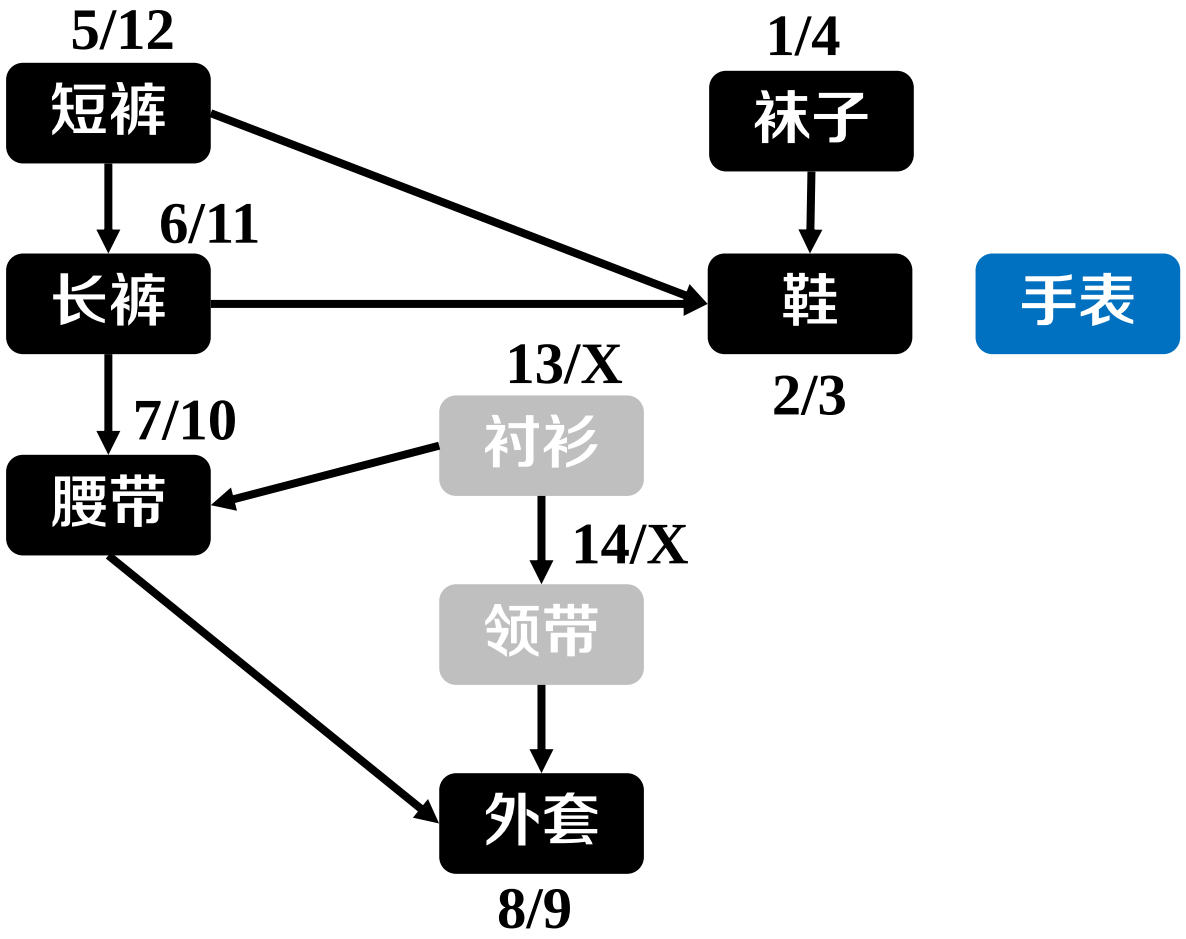
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序
    - 完成时刻越早：按完成时刻逆序？



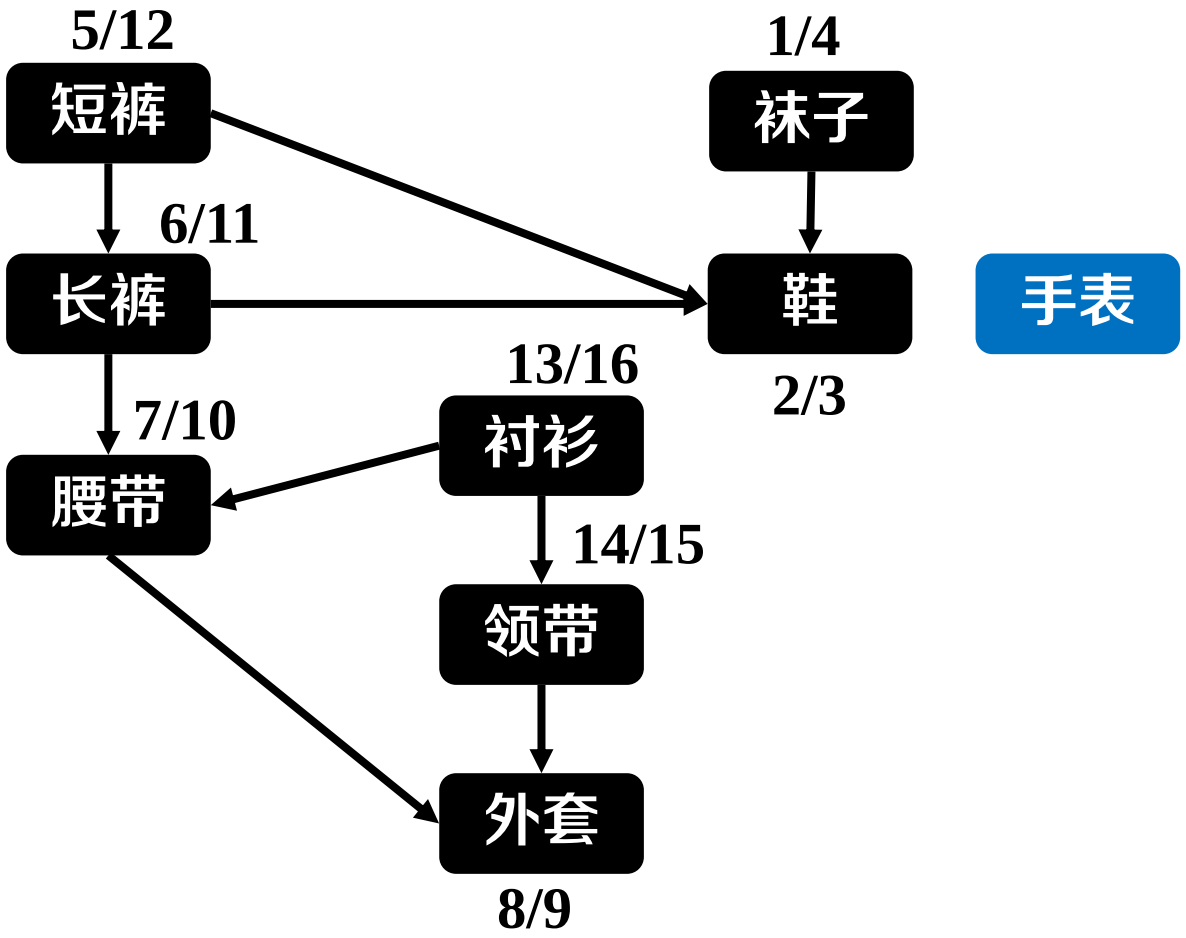
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序
    - 完成时刻越早：按完成时刻逆序



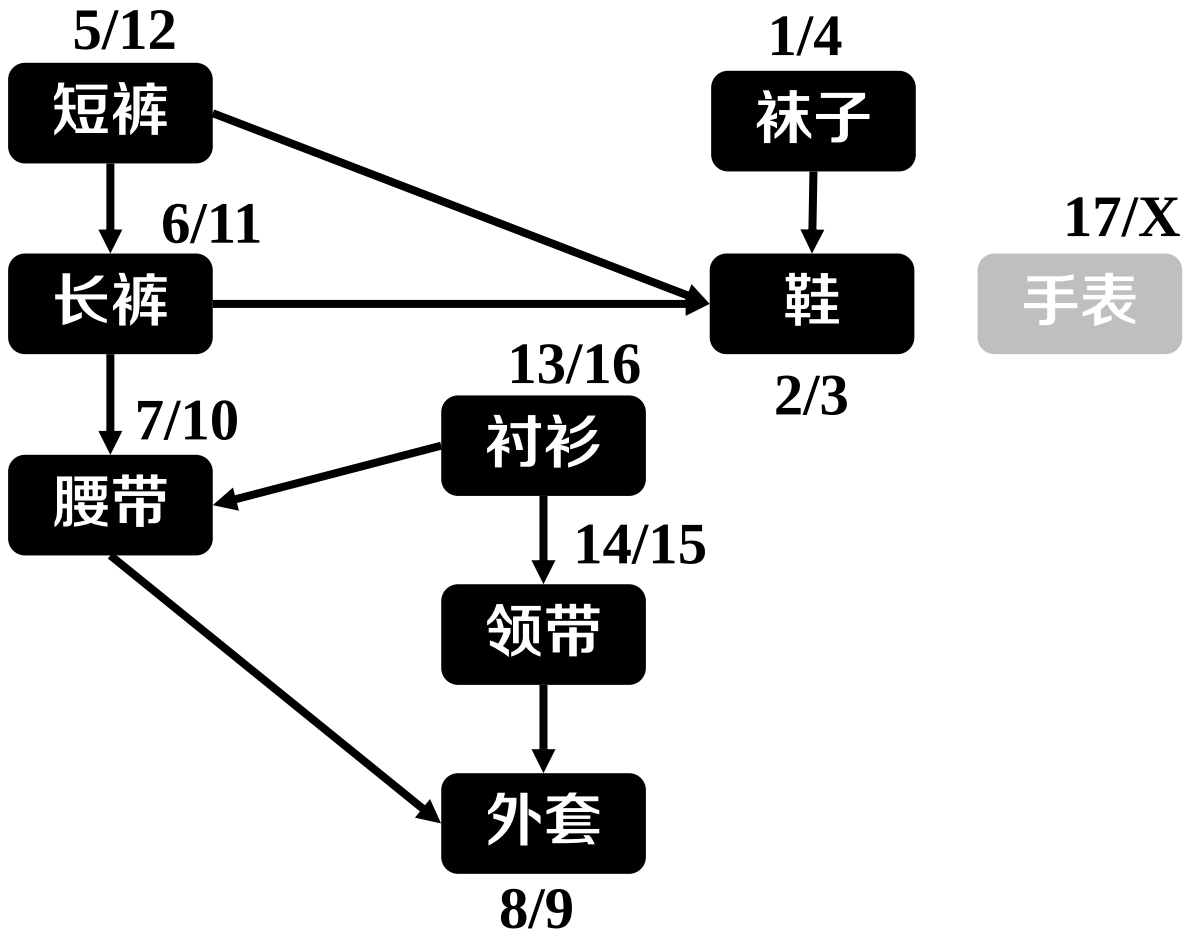
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序 ✗
    - 完成时刻越早：按完成时刻逆序？



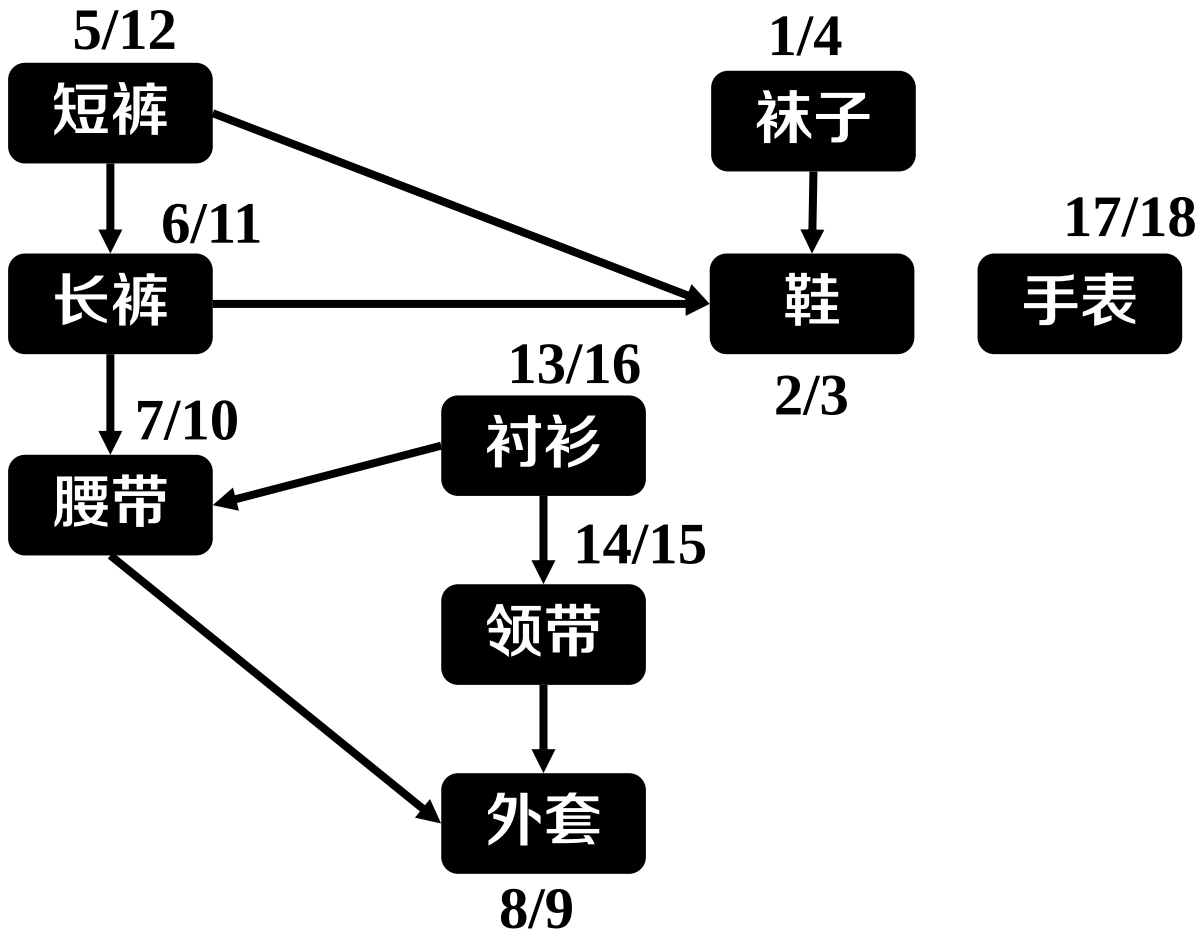
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序 ✗
    - 完成时刻越早：按完成时刻逆序？



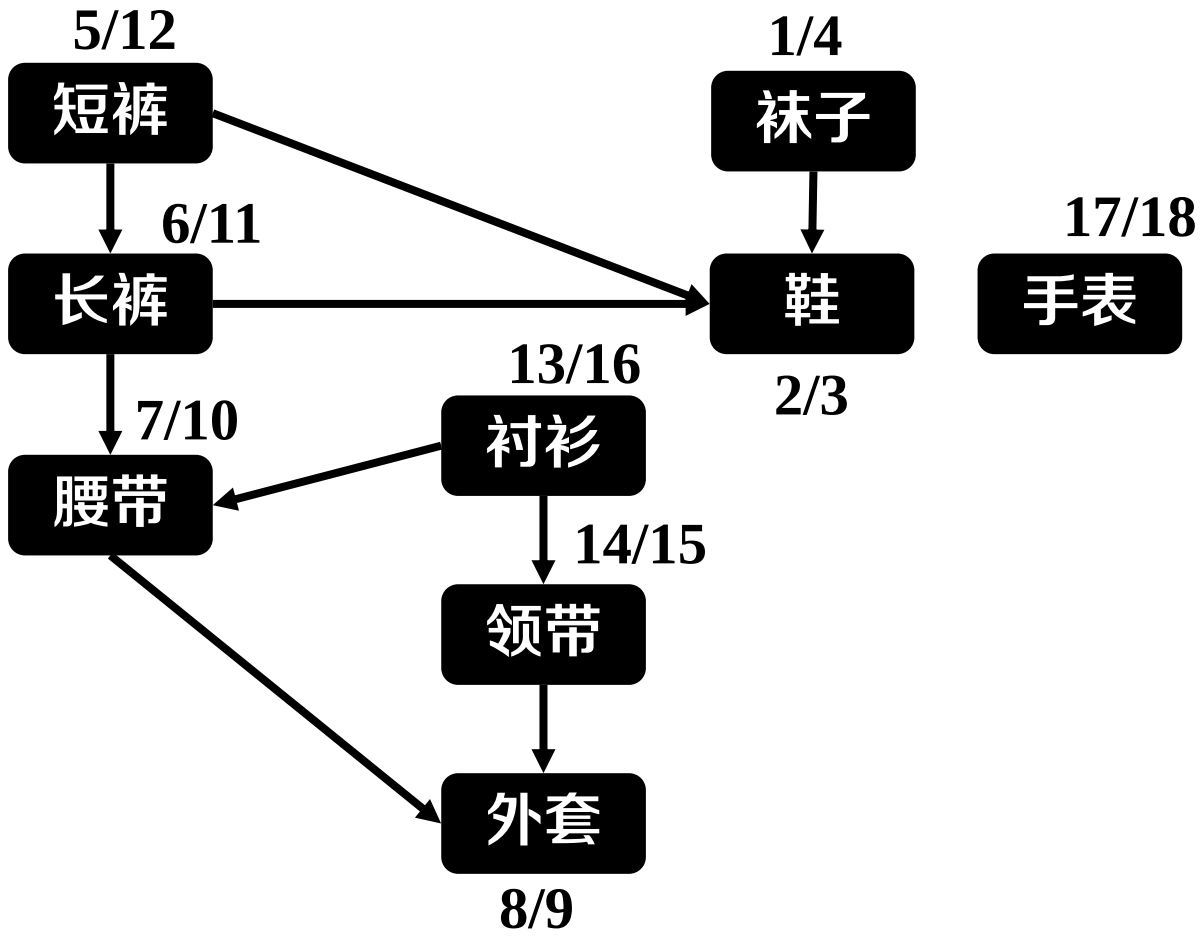
算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序 ✗
    - 完成时刻越早：按完成时刻逆序？



算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序 ✗
    - 完成时刻越早：按完成时刻逆序？

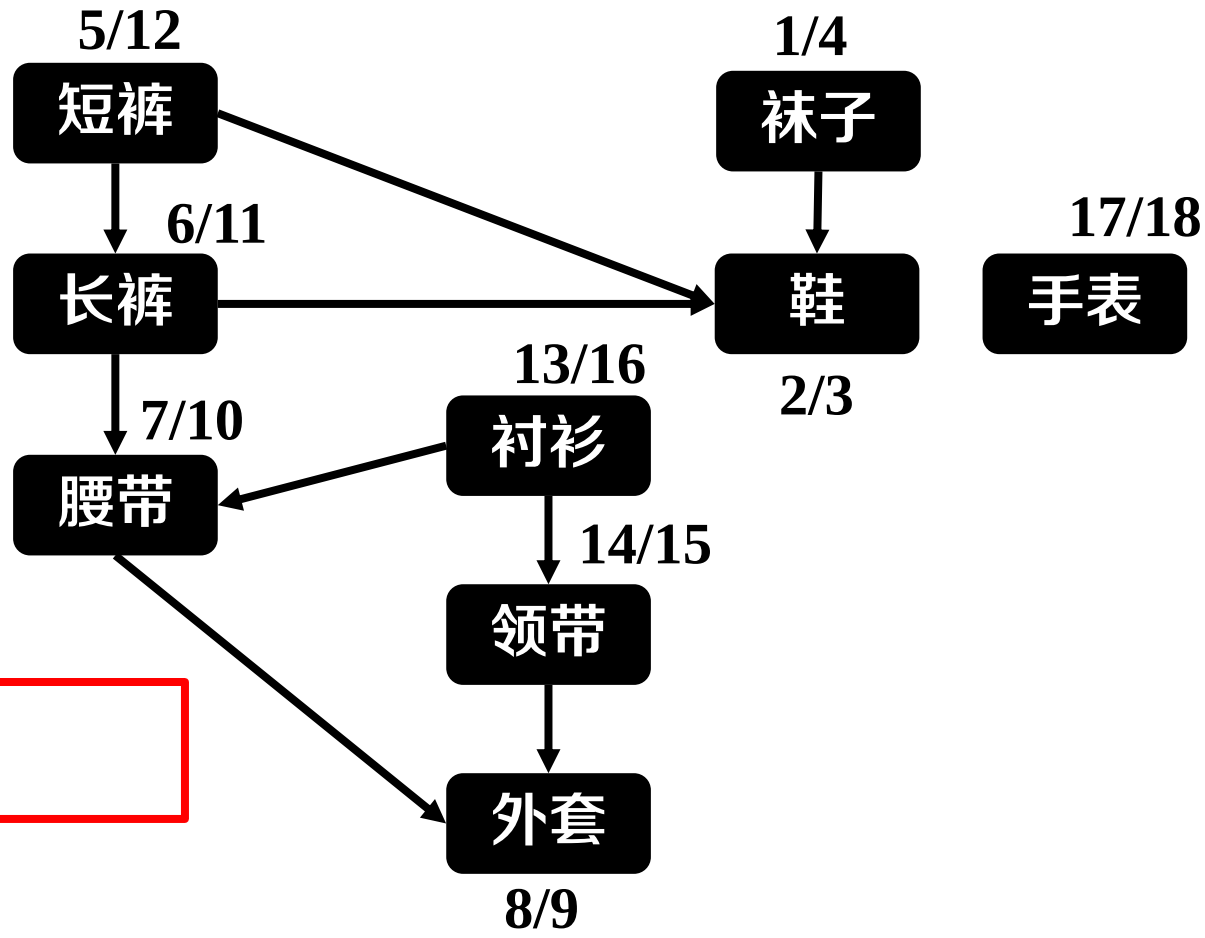


完成时刻逆序排列

手表 | 衬衫 | 领带 | 短裤 | 长裤 | 腰带 | 外套 | 袜子 | 鞋

算法思想

- 从DFS的视角观察
  - 穿衣顺序和搜索深度有关：深度越深，顺序越靠后
  - 深度越深
    - 发现时刻越晚：按发现时刻顺序 ✗
    - 完成时刻越早：按完成时刻逆序？



按完成时刻逆序是否正确？

完成时刻逆序排列

手表 | 衬衫 | 领带 | 短裤 | 长裤 | 腰带 | 外套 | 袜子 | 鞋

提纲

问题定义

广度优先策略

深度优先策略

算法分析

正确性证明

- 深度优先搜索确定的顺序：顶点完成时刻的逆序
- 拓扑序：对任意边(u,v)，u在v前面

正确性证明

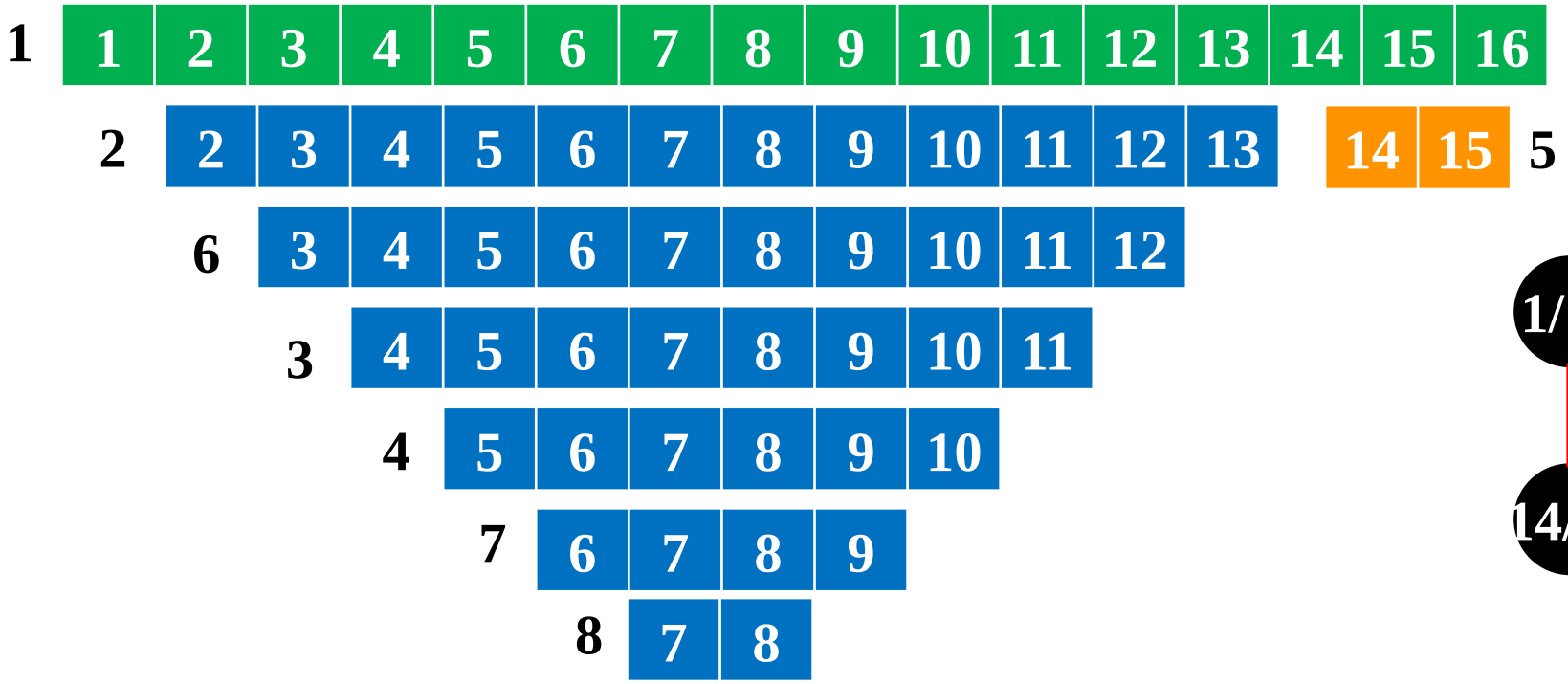
- 已知深度优先搜索确定的顺序：顶点完成时刻的逆序
- 已知拓扑序：对任意边(u,v)，u在v前面
- 对任意边(u,v)，完成时刻满足：$f(u) > f(v)$ ⟶ 算法正确

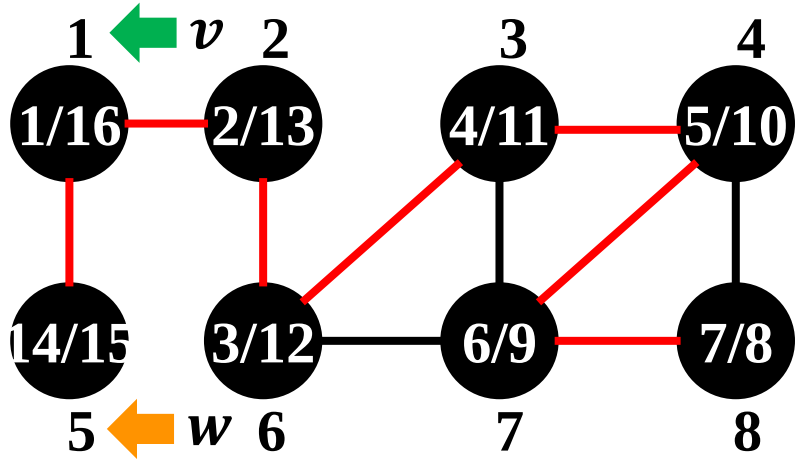
正确性证明

- 已知深度优先搜索确定的顺序：顶点完成时刻的逆序
- 已知拓扑序：对任意边(u,v)，u在v前面
- 对任意边(u,v)，完成时刻满足：$f(u)>f(v)$ ⟶ 算法正确
  - 证明：设当前顶点为u，搜索顶点v

正确性证明

- 已知深度优先搜索确定的顺序：顶点完成时刻的逆序
- 已知拓扑序：对任意边(u,v)，u在v前面
- 对任意边(u,v)，完成时刻满足：$f(u)>f(v)$ ⟶ 算法正确
  - 证明：设当前顶点为u，搜索顶点v
    - 若v为白色，v是u的后代，$f(u)>f(v)$ （括号化定理）



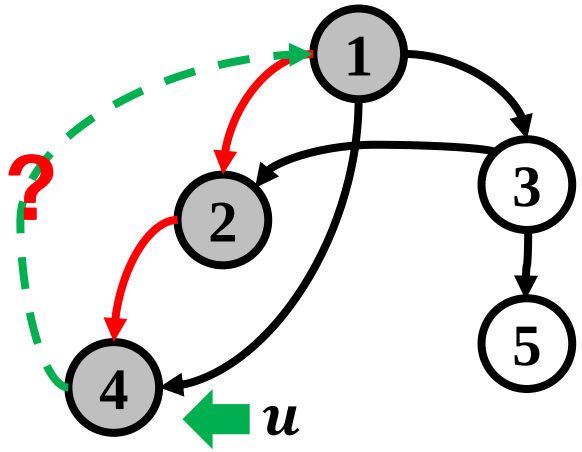


正确性证明

- 已知深度优先搜索确定的顺序：顶点完成时刻的逆序
- 已知拓扑序：对任意边(u,v)，u在v前面
- 对任意边(u,v)，完成时刻满足：$f(u)>f(v)$ ⟶ 算法正确
  - 证明：设当前顶点为u，搜索顶点v
    - 若v为白色，v是u的后代，$f(u)>f(v)$ （括号化定理）
    - 若v为黑色，v已经完成，u尚未完成，$f(u)>f(v)$

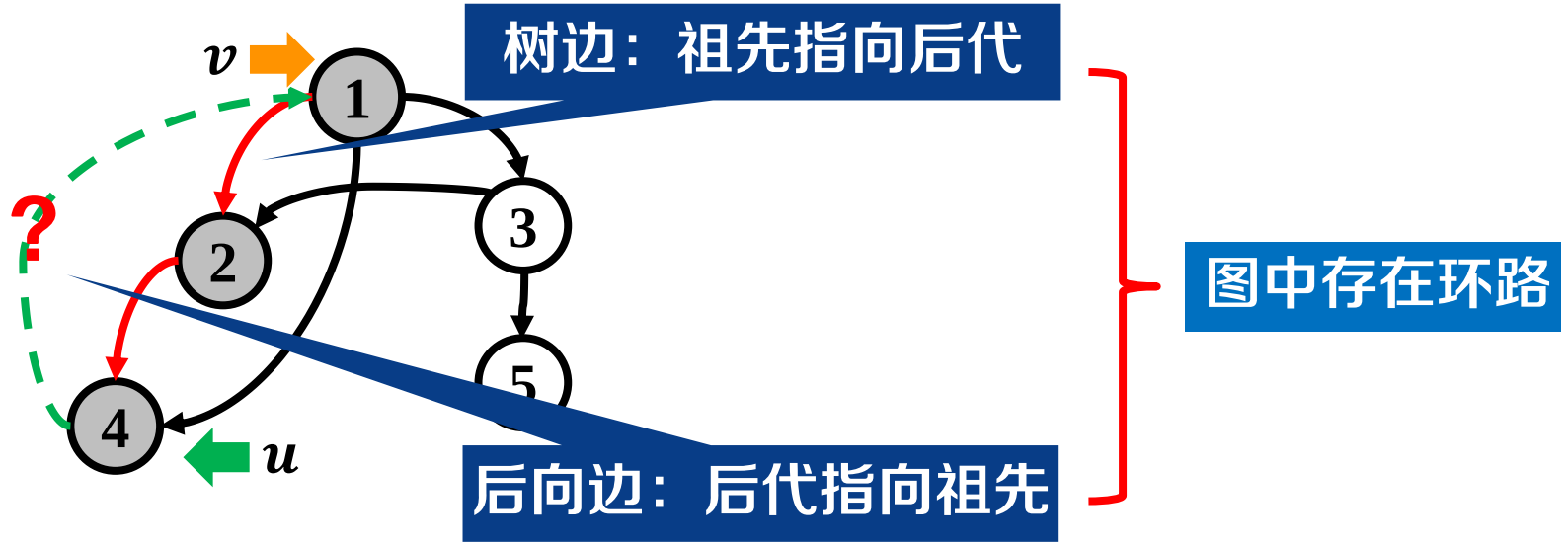
正确性证明

- 已知深度优先搜索确定的顺序：顶点完成时刻的逆序
- 已知拓扑序：对任意边(u,v)，u在v前面
- 对任意边(u,v)，完成时刻满足：$f(u)>f(v)$ ⟶ 算法正确
  - 证明：设当前顶点为u，搜索顶点v
    - 若v为白色，v是u的后代，$f(u)>f(v)$ （括号化定理）
    - 若v为黑色，v已经完成，u尚未完成，$f(u)>f(v)$
    - 若v是灰色?



正确性证明

- 已知深度优先搜索确定的顺序：顶点完成时刻的逆序
- 已知拓扑序：对任意边(u,v)，u在v前面
- 对任意边(u,v)，完成时刻满足：$f(u)>f(v)$ ⟶ 算法正确
  - 证明：设当前顶点为u，搜索顶点v
    - 若v为白色，v是u的后代，$f(u)>f(v)$ （括号化定理）
    - 若v为黑色，v已经完成，u尚未完成，$f(u)>f(v)$
    - 若v是灰色？不可能！因为有向无环图不存在后向边



正确性证明

- 已知深度优先搜索确定的顺序：顶点完成时刻的逆序
- 已知拓扑序：对任意边(u,v)，u在v前面
- 对任意边(u,v)，完成时刻满足：$f(u)>f(v)$ ⟶ 算法正确
  - 证明：设当前顶点为u，搜索顶点v
    - 若v为白色，v是u的后代，$f(u)>f(v)$ （括号化定理）
    - 若v为黑色，v已经完成，u尚未完成，$f(u)>f(v)$
    - 若v是灰色？不可能！因为有向无环图不存在后向边

伪代码

- **Topological-Sort-DFS(G)**

输入: 图G
输出: 顶点拓扑序
$L \leftarrow DFS(G)$
return $L.reverse()$

数组中元素逆序排列

伪代码

- **Topological-Sort-DFS(G)**

输入: 图G
输出: 顶点拓扑序
$L \leftarrow DFS(G)$
return $L.reverse()$

问题：如何在搜索过程中得到按完成时刻顺序排列的顶点?

伪代码

- **DFS(G)**

```
输入: 图 G
新建数组 color[1..V], L[1..V]
for v ∈ V do
    color[v] ← WHITE
end
for v ∈ V do
    if color[v] = WHITE then
        L' ← DFS-Visit(G, v)
        向L结尾追加L'
    end
end
return L
```

- **DFS-Visit(G, v)**

```
输入: 图 G, 顶点 v
输出: 按完成时刻从早到晚排列的顶点 L
color[v] ← GRAY
for w ∈ G.Adj[v] do
    if color[w] = WHITE then
        L ← DFS-Visit(G, w)
    end
end
color[v] ← BLACK
向L结尾追加顶点v
return L
```

顶点按完成时刻排列

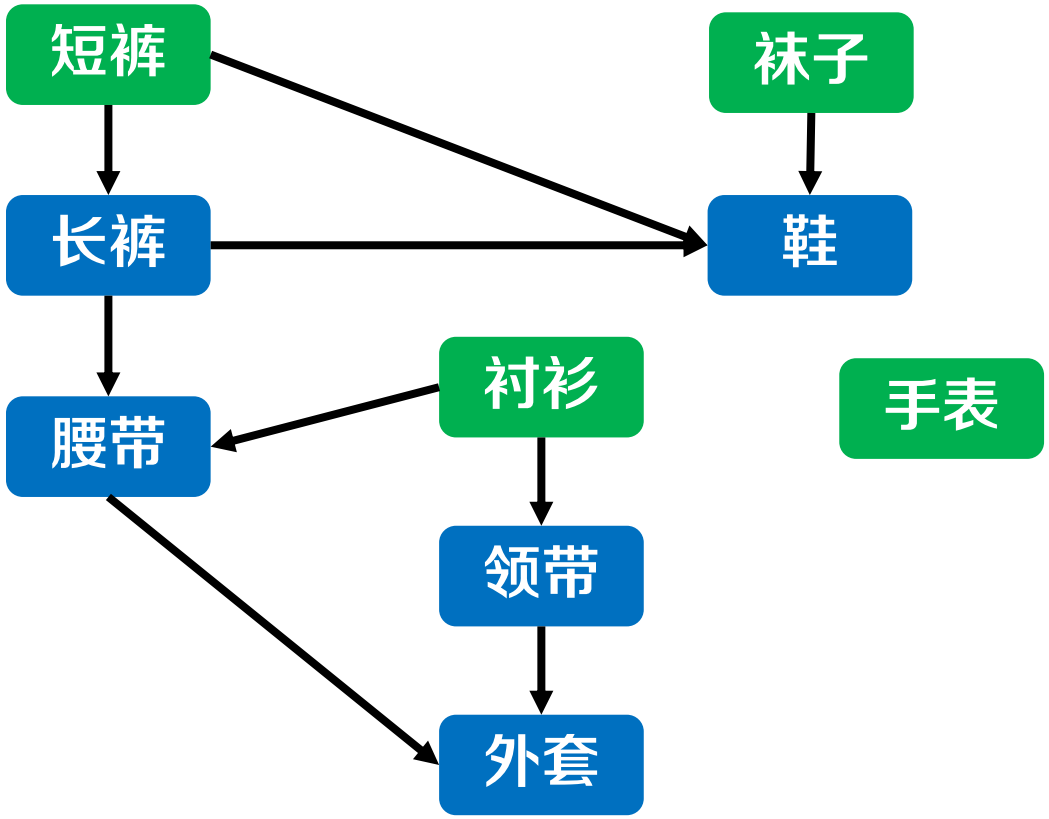
时间复杂度

- **Topological-Sort-DFS(G)**

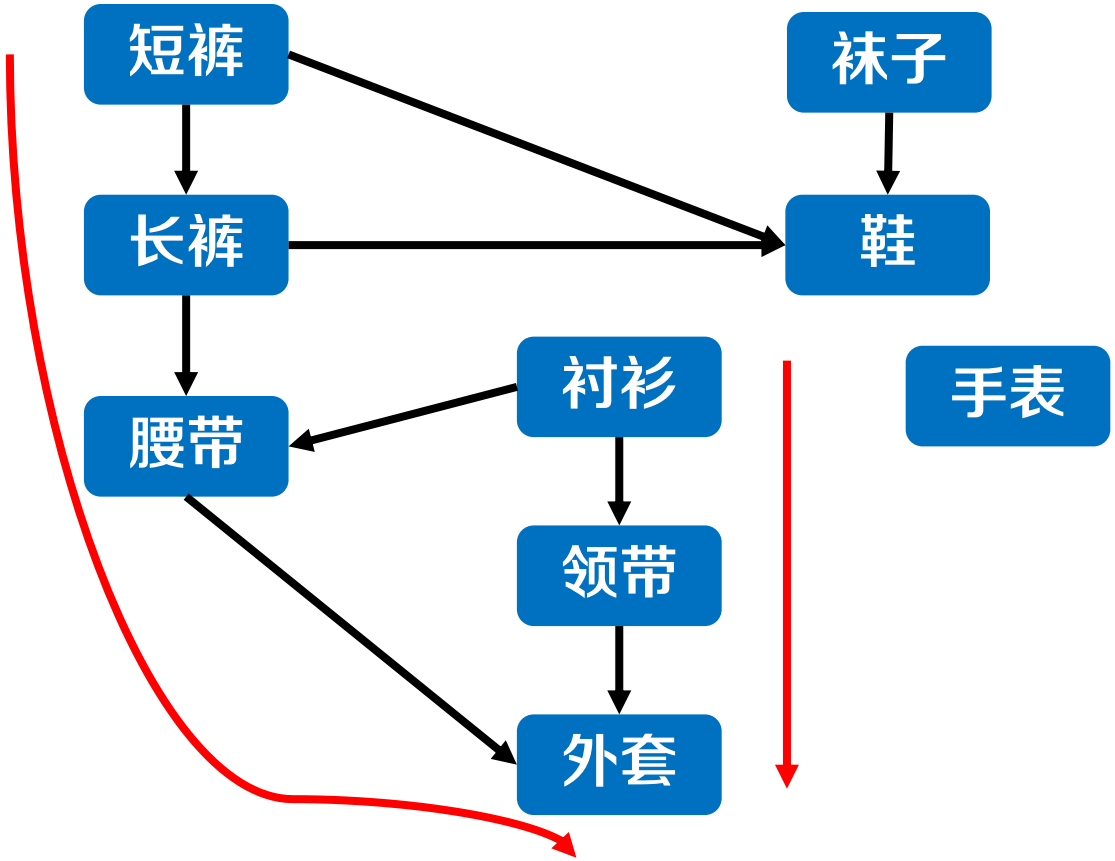
输入: 图G
输出: 顶点拓扑序
$L \leftarrow DFS(G)$
return $L.reverse()$

时间复杂度：$O(|V| + |E|)$

小结



广度优先
顺序思想：把容易完成的事优先完成



深度优先
逆序思想：把不易完成的事放到后面