PKU-04833060
算法设计与分析『实验班』

第4讲 动态规划 (2/2)

罗国杰

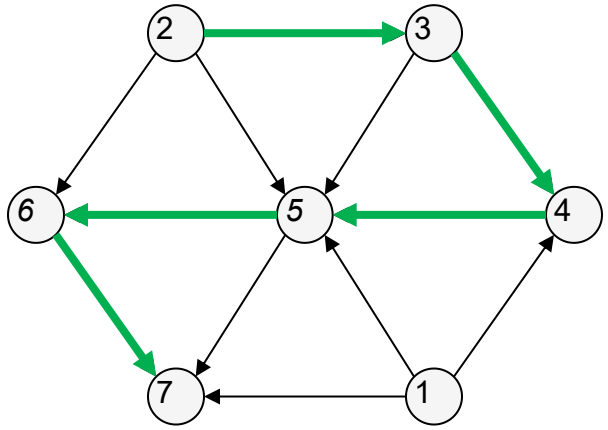
gluo@pku.edu.cn

2025年春季学期

有向无环图的最长路径

- 给定有向无环图 G，求最长路径

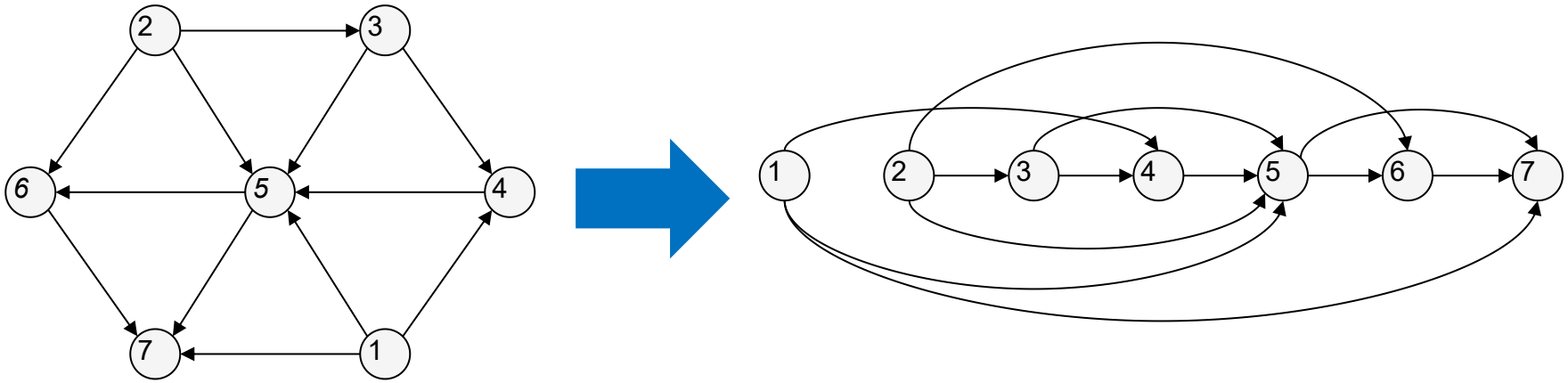
- 对于一般的有向图，最长路径问题是 NP 难的
- 有向哈密顿路径是该问题的特例



有向无环图的最长路径：动态规划

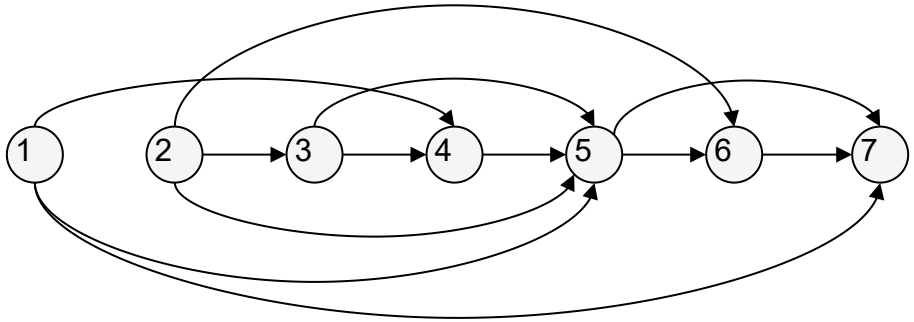
- 如何确定动态规划的阶段?
- 利用 G 是有向无环图的性质

- 节点的拓扑序



有向无环图的最长路径：动态规划

- 假设已对节点做标记
- 只有 $i < j$ 时，(i, j) 才存在有向边



- 令 $OPT(j)$ = 以 j 为终点的最长路径长度
- 假设 $(i_1, i_2), (i_2, i_3), \ldots, (i_{k-1}, i_k), (i_k, j)$ 时实现 OPT(j) 的一条路径，则
- 性质1：$i_1 \leq i_2 \leq \ldots \leq i_k \leq j$.
- 性质2：$(i_1, i_2), (i_2, i_3), \ldots, (i_{k-1}, i_k)$ 是以 i_k 为终点的一条最长路径
- $OPT(j) = 1 + OPT(i_k)$.

有向无环图的最长路径：动态规划

- 假设已对节点做标记
- 只有 $i < j$ 时，(i, j) 才存在有向边

- 令 $OPT(j)$ = 以 j 为终点的最长路径长度

$$OPT(j) = \begin{cases} 0 & If\ j\ is\ a\ source \\ 1 + \max_{i:(i,j)\ \text{an edge}} OPT(i) & o.w \end{cases}$$

有向无环图的最长路径：输出最长路径

```
Let G be a DAG given with a topological sorting:
   For all edges (i,j) we have i < j.
Initialize Parent[j]=-1 for all j.
Compute-OPT(j){
   if (in-degree(j) == 0)
     return 0
   if (M[j] == empty)
     M[j] = 0;
     for all edges (i,j)
       if (M[j] < 1+Compute-OPT(i))
         M[j] = 1 + Compute-OPT(i)
         Parent[j] = i
   return M[j]
}
Let k be the maximizer of Compute-OPT(1),…,Compute-OPT(n)
While (Parent[k]!=-1)
   Print k
   k = Parent[k]
```

记录获得 OPT(j) 的前驱节点

树的最大独立集

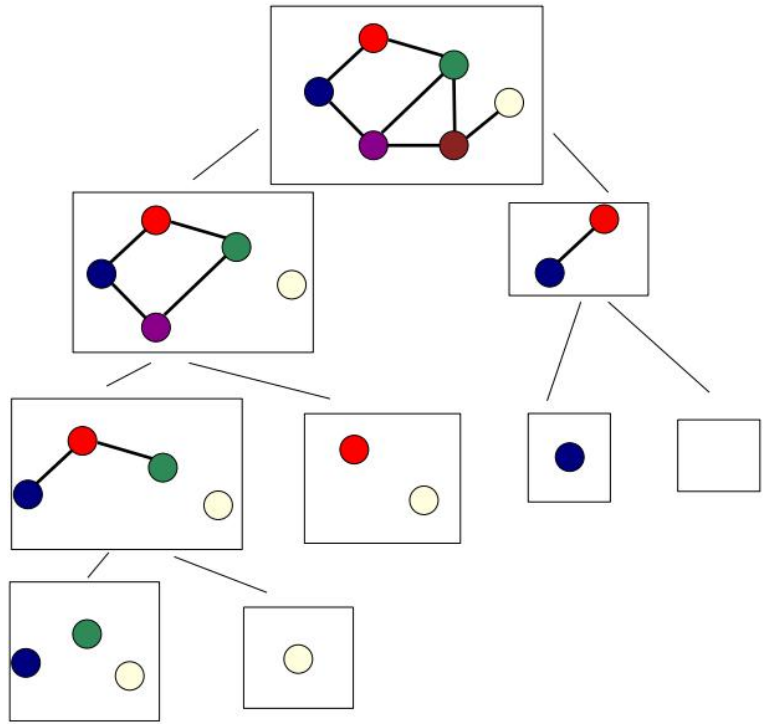
- 给定一棵无向树 $T = (V, E)$
- $S \subset V$ 称为独立集，如果 S 任意两个顶点都没有边，
- 给出线性时间算法求解树的最大独立集

- 记 $F(v)$ 为以 v 为根的子树的最大独立集，则
- $F(v) = \max\{\quad 1 + \sum_{g:\, grandchild\ of\ v} F(g),\quad \sum_{c:\, child\ of\ v} \mathrm{OPT}(c)\quad\}$

图的最大点独立集

- 在一般图上执行类似算法，时间 $O(2^n)$

```
MIS(G):
  if G is empty then return 0
  v ← a vertex of G
  a = MIS(G - v)
  b = 1 + MIS(G - v - neighbors(v))
  return max(a,b)
```



为什么树上动态规划有效?

- 每个节点（包括根节点）都是割点
- 点割集（separator）：给定图 G=(V,E)，若点集 S⊂V 使 G-S 包含至少两个连通分量，则 S 是 G 的点割集。
- 平衡点割集：若 G-S 的每个连通分量包含最多 |V| * 2/3，则 S 是平衡点割集

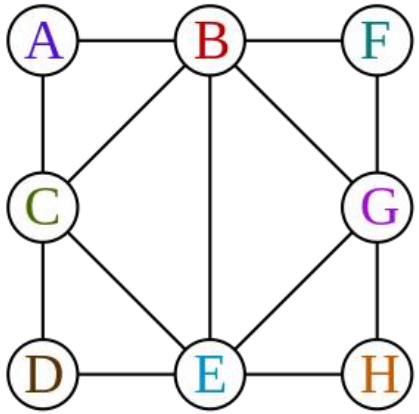
- 性质：每棵树都有只包含一个节点的平衡点割集

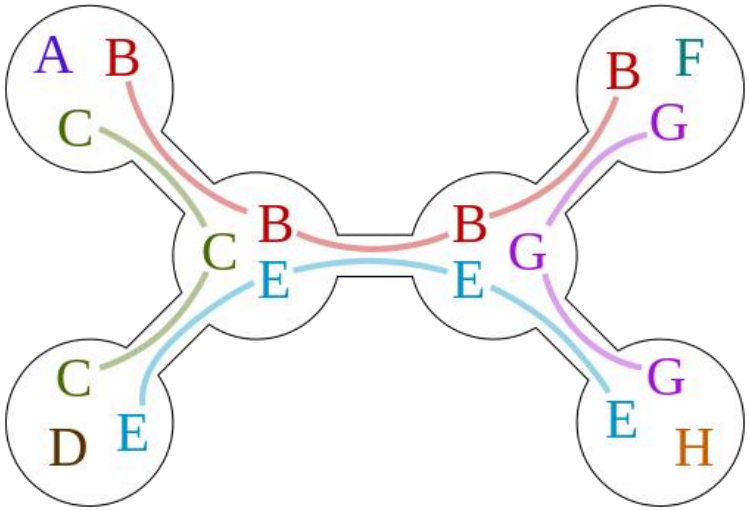
图的树分解

- 给定图 $G = (V, E)$，G 的树分解是一对 (T, χ)，其中 $T = (N, F)$ 是树，并且 $\chi: N \to 2^V$ 为每个节点分配一组顶点（称为节点的包），满足以下条件：
  - (node coverage) 对于每个顶点 $v \in V$，存在节点 $i \in N$ 使得 $v \in \chi(i)$。
  - (edge coverage) 对于每一条边 $(v, w) \in E$，存在一个 $i \in N$ 且 $v \in \chi(i)$ 且 $w \in \chi(i)$。
  - (coherence) 对于每个 $i, j, k \in N$：如果 j 位于 i 和 k 之间的路径上，则 $\chi(i) \cap \chi(k) \subseteq \chi(j)$。
    - 等价地，任意 $v \in V$ 在 T 中对应的节点是连通的，形成一棵子树。
- 树分解的宽度定义为 $\max_{i \in N} |\chi(i)| - 1$
- 图的树宽是所有树分解中的最小宽度
  - 通俗地，图 G 的树宽是 k，意味着 G 可以被大小不超过 k 的若干平衡点割集递归地分解

（形式化证明略）

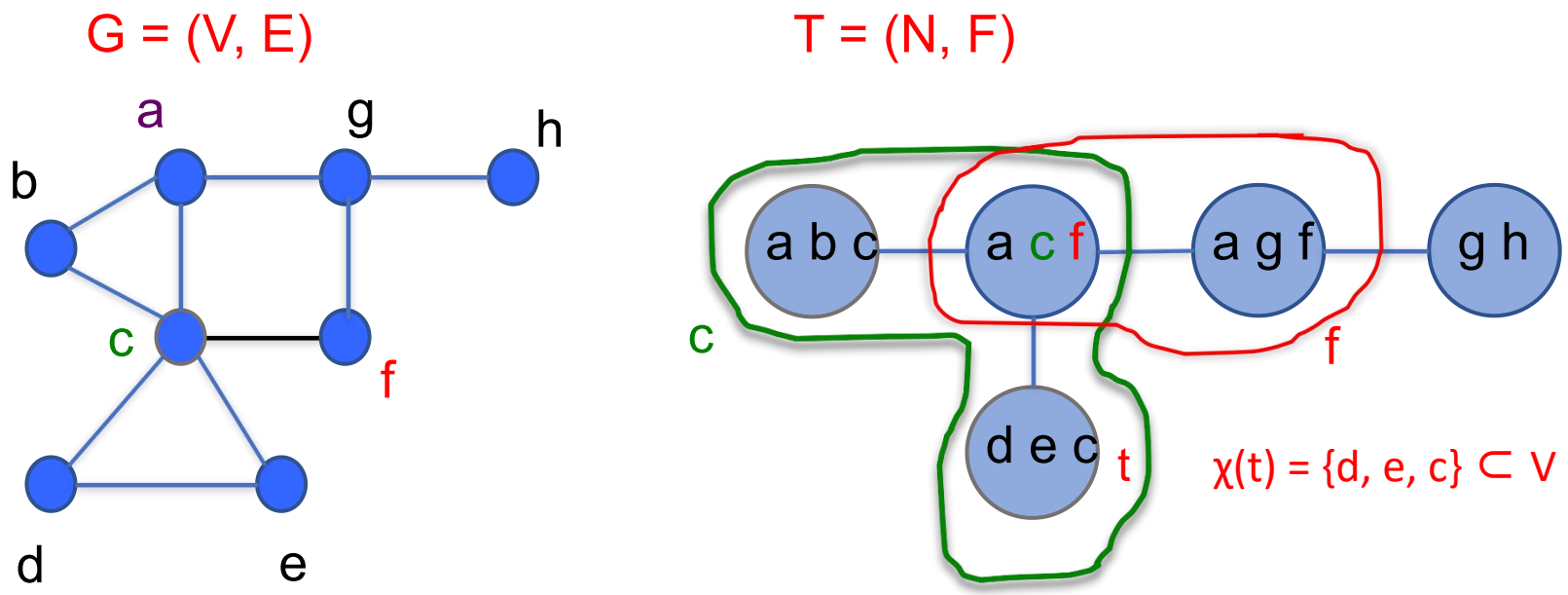
图的树分解：例子





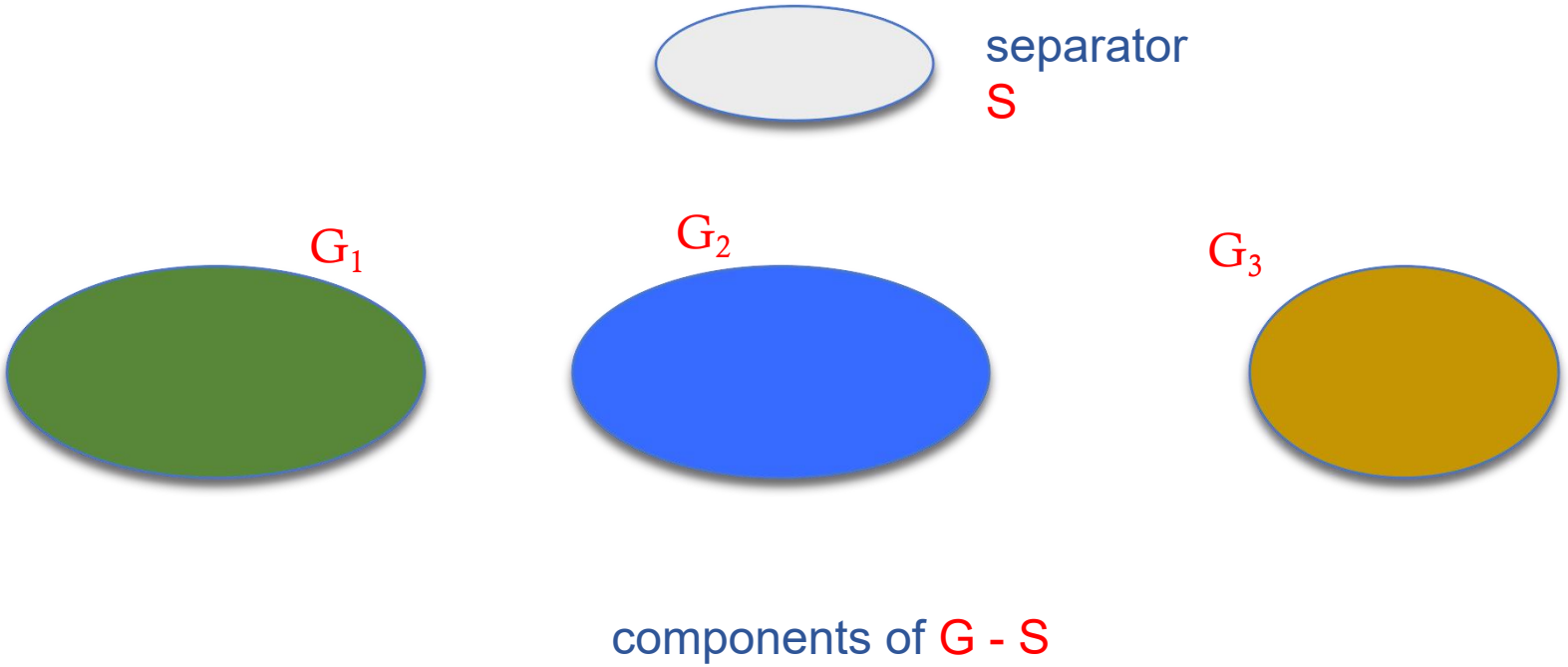
source: liyanxu.blog

图的树分解：例子



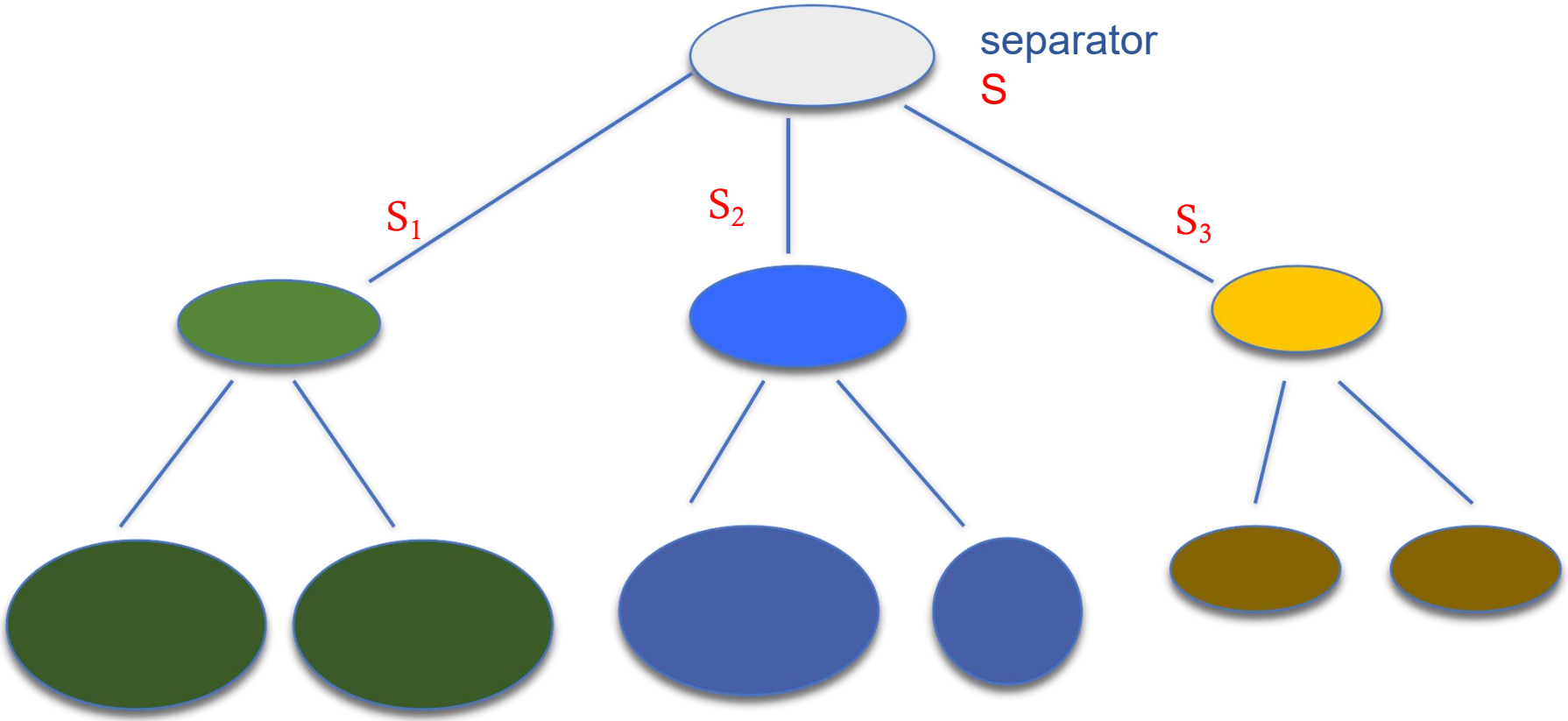
图的树分解：示意图

- 通俗地，图 G 树宽是 k，意味着 G 可被大小不超过 k 的平衡点割集递归地分解



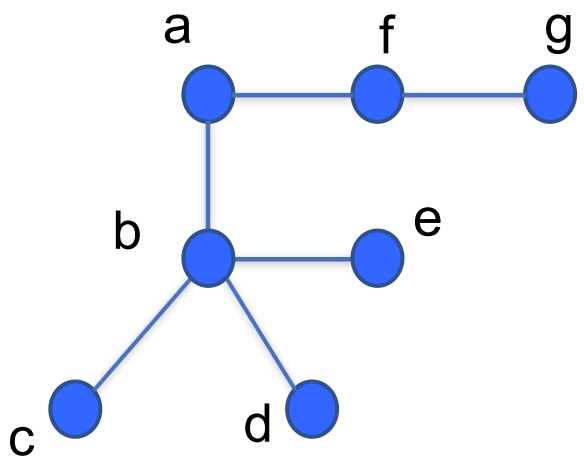
图的树分解：示意图

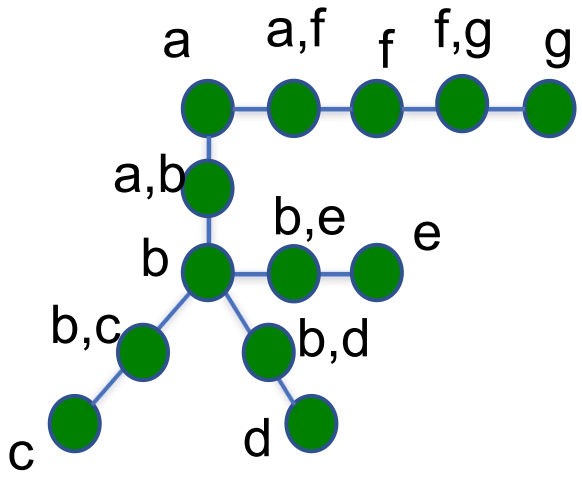
- 通俗地，图 G 树宽是 k，意味着 G 可被大小不超过 k 的平衡点割集递归地分解

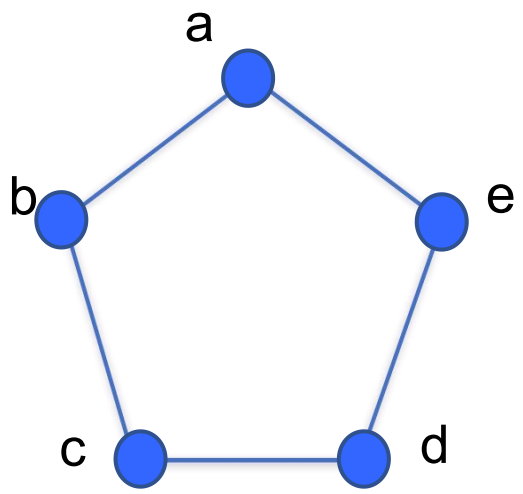


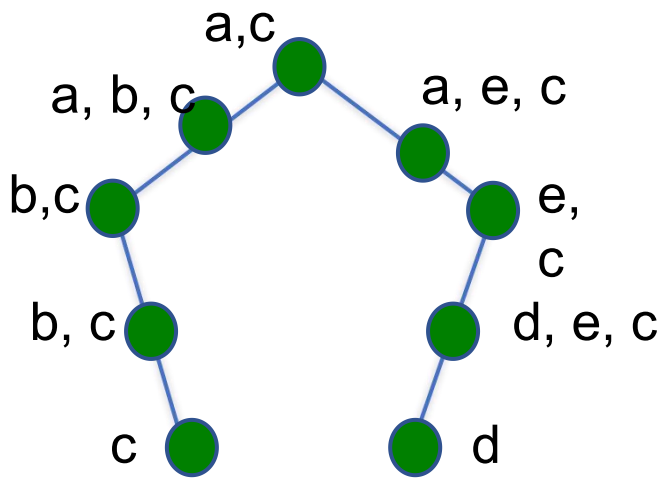
图的树分解：典型图的树宽

- tw(Tree) = 1
- tw(Cycle) = 2
- tw(Kn) = n - 1
  - Kn: n阶完全图
- tw(k×k Grid) = k - 1
- tw(Planar) = $O(n^{1/2})$
  - via [Lipton-Tarjan]









图的树分解：最大独立集

- OPT(t, S)：满足 $I \cap \chi(t) = S$ 且 $I \subseteq \chi(\text{subtree}(t))$ 的独立集 I 中的最大独立集大小
  - t 的子树节点，去掉 $\chi(t)$ 后，两两交集为空（根据树分解的第三项条件）
  - $\text{OPT}(t, S) = |S| + \Sigma_{c \in \text{children}} (A(S \cap \chi(c), c, t) - |S \cap \chi(c)|)$
  - $A(S, c, t) = \max \{ \text{OPT}(c, S') : S' \subset \chi(c) \text{ and } S = S' \cap \chi(t) \}$ （子树 c 包含 S 中元素的最大独立集大小）
- 每个节点计算次数 $\leq 2^{k+1}$，其中 k 是树宽
- 从叶子至根（树分解上的动态规划）总时间 $O(k\, 2^{k+1}\, N)$，其中 N 是树分解节点数

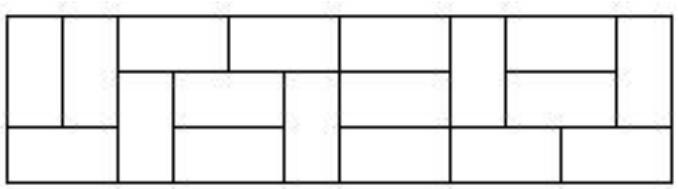
- 给定图 G 的树宽为 k 的树分解，最大独立集算法的时间上界是 $O(k\, 2^{k+1}\, n)$
  - 对于固定的 k，多项式时间算法
- 平面图的最大独立集算法的时间上界是 $2^{O(\sqrt{n})}$
  - 利用平衡点割集、以及树分解上的动态规划

动态规划 + 图的树分解 解 低树宽NP难问题

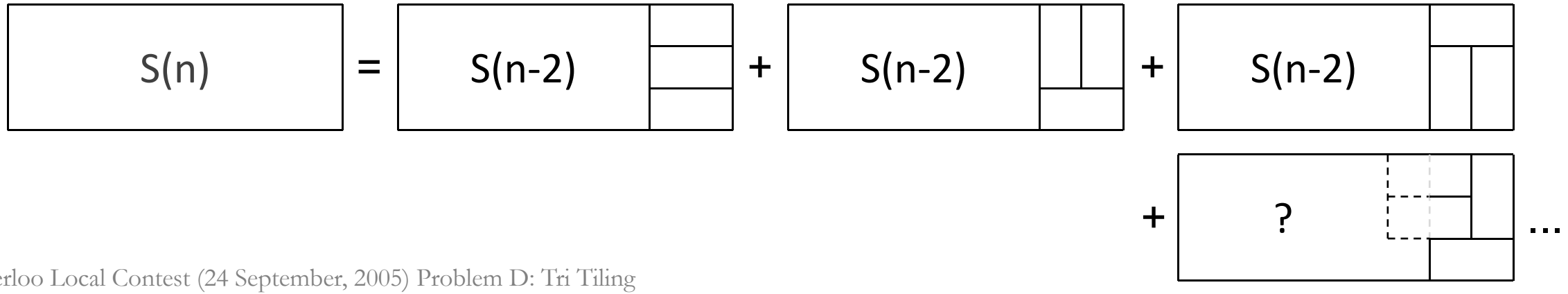
- [Arnborg-Corneil-Proskurowski'87]
  - 给定 G 和 k，判定问题 "tw(G) ≤? k" ∈ NPC
- [Bodleander'93]
  - 存在 $O(k^{k^3}n)$ 时间算法判断 tw(G) ≤ k
  - 即对于固定的 k，线性时间
- [Bodleander et al'13]
  - $O(c^k n)$ 时间的"5-近似"算法
- [Feige-Hajiaghayi-Lee'05]
  - 多项式时间输出 ≤ c tw(G) √log tw(G) 树分解
- 启发式方法求 tw 上界
  - H. L. Bodlaender and A. M. C. A. Koster, "Treewidth computations I. Upper bounds," Inf. Comput., vol. 208, no. 3, pp. 259–275, Mar. 2010.

多米诺覆盖窄长棋盘

- 给定 n，求用 1×2 多米诺骨牌覆盖 3×n 棋盘的方案总数

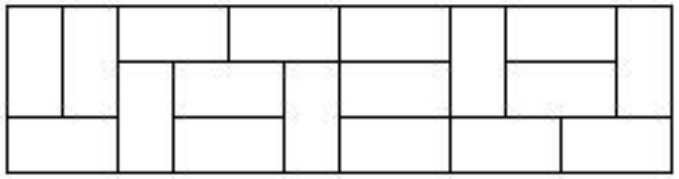


- 尝试：S(n) 表示覆盖 3×n 棋盘的方案总数
- 子问题：

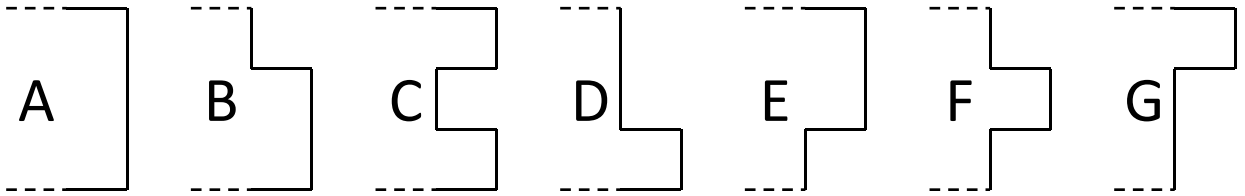


多米诺覆盖窄长棋盘

- 给定 n，求用 1×2 多米诺骨牌覆盖 3×n 棋盘的方案总数



- 尝试：T(n,X) 表示覆盖 3×n 棋盘（除最右列外）的方案总数，X∈{A,B,C,D,E,F,G}



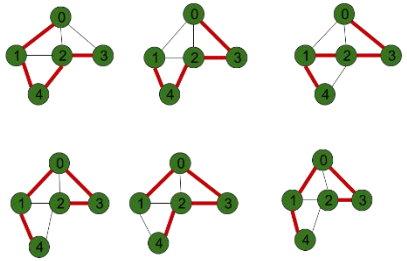
- 子问题：T(n,A) = 3*T(n-2,A) + T(n-1,B) + T(n-1,E) + ...

多米诺覆盖棋盘

- 给定 n，求用 1×2 多米诺骨牌覆盖 m×n 棋盘的方案总数
- a closed problem: $$\prod_{j=1}^{\lceil \frac{m}{2} \rceil} \prod_{k=1}^{\lceil \frac{n}{2} \rceil} \left(4\cos^2 \frac{\pi j}{m+1} + 4\cos^2 \frac{\pi k}{n+1} \right).$$
  - [TeF61] H. N. V. Temperley and M. E. Fisher, “Dimer problem in statistical mechanics-an exact result,” Philos. Mag., vol. 6, no. 68, pp. 1061–1063, Aug. 1961, doi: 10.1080/14786436108243366.
  - [Kas61] P. W. Kasteleyn, “The statistics of dimers on a lattice,” Physica, vol. 27, no. 12, pp. 1209–1225, Dec. 1961, doi: 10.1016/0031-8914(61)90063-5.

货郎问题

- 给定 n 座城市和他们两两之间的距离 d_{ij}
- 目标：求访问每座城市刚好一次的最短路径长度
- （为简化表达，本例的 TSP 问题考虑哈密顿路径）



- 暴力：$n! \sim 2^{O(n\log n)}$ 时间

- 子问题：$T(v,S)$ 表示终止在 v 点、访问 S 每座城市刚好一次的最短路径长度
- 观察得：$T(v,S) = \min_{u \in S-v} \{ T(u, S-v) + d_{uv} \}$

货郎问题

- 暴力：$n! \sim 2^{O(n\log n)}$ 时间
- 子问题：$T(v,S)$ 表示终止在 v 点、访问 S 每座城市刚好一次的最短路径长度
- 算法：

```
Set T(v,{v}) = 0 for all v

for k = 2,...,n
   for all sets S of size k
      for all v in S
         T(v,S) = min_{u∈S-v} { T(u,S-V) + d_uv }
```

运行时间：$O(n2^n)$

- $n2^n$ vs $2^{O(n\log n)}$：当 n=30 时，前者≈3.2×10^{10}，后者≈2.2×10^{13}

“动态规划”名字的历史

- I spent the Fall quarter (of 1950) at RAND. My first task was to find a name for multistage decision processes. An interesting question is, Where did the name, dynamic programming, come from? The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word research. I’m not using the term lightly; I’m using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term research in his presence. You can imagine how he felt, then, about the term mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word “programming”. I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying I thought, lets kill two birds with one stone. Lets take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is it's impossible to use the word dynamic in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities. ——出自 Bellman 自传《Eye of the Hurricane》

Richard Bellman

- 一句话：他老板厌恶数学，他需要造个酷的、不那么数学的术语。
  - “dynamic” ~ 多阶段、随时间变化
  - “programming”也有军事日程的意思。

动态决策问题（动规最初研究的问题）

- 给定有向图 G 和初始顶点 v_0
- 每个顶点表示某个状态，每条边 e 有回报 r_e（可以是负的）
- 定义路径 $p = v_0 v_1 v_2 \cdots$ 的回报

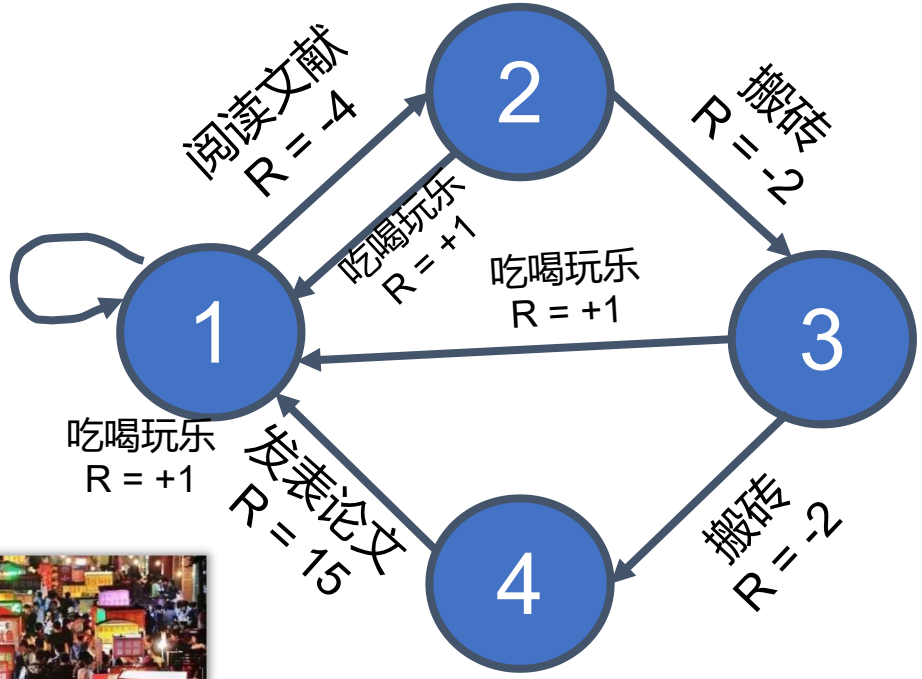
$$R(p) = r_{v_0 v_1} + \gamma \cdot r_{v_1 v_2} + \gamma^2 \cdot r_{v_2 v_3} + \cdots$$

  - 其中衰减因子 $0 < \gamma < 1$
- 目标：找到从 v_0 出发、回报 $R(p)$ 最大的路径 p

- （马尔可夫决策过程的简化版）

动态决策问题：例子

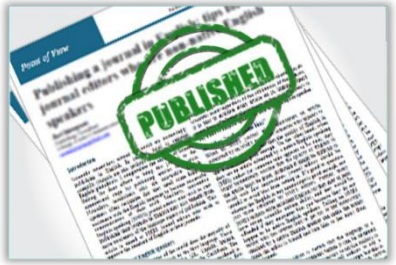
- 起始状态为顶点1
- 问题：最高价值的路径?



* 以上奖励值均为虚构

- 两类自然的选项：

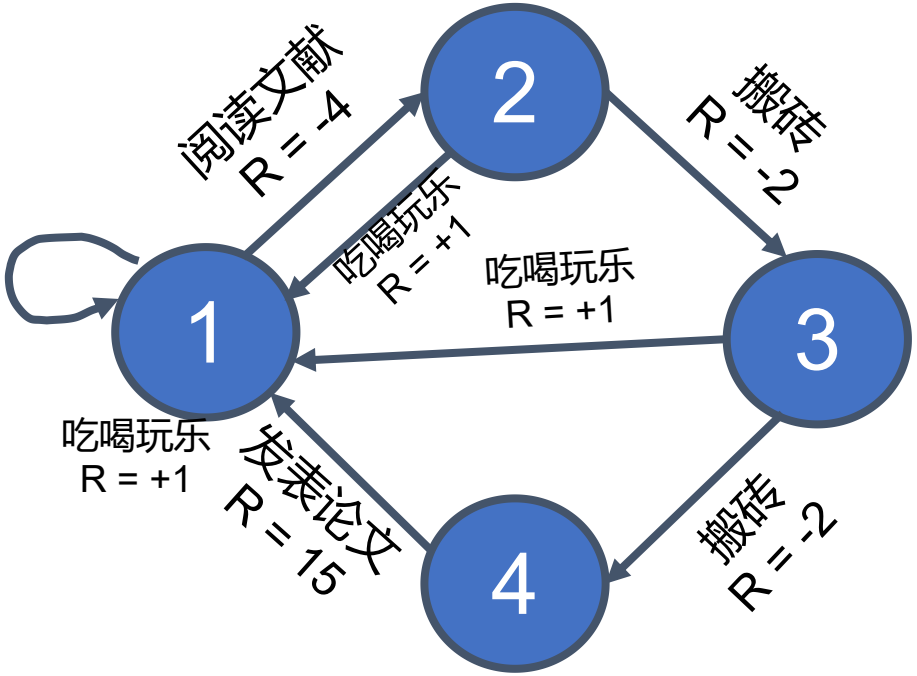
1. 每天坚持吃喝玩乐 (1,1,1,1,1,...)
2. 沉迷于发表论文 (1,2,3,4,1,2,3,4,...)



- 哪条是最优路径?
- 取决于 γ （或者说，还剩多少天能拿回报……）

动态决策问题：例子

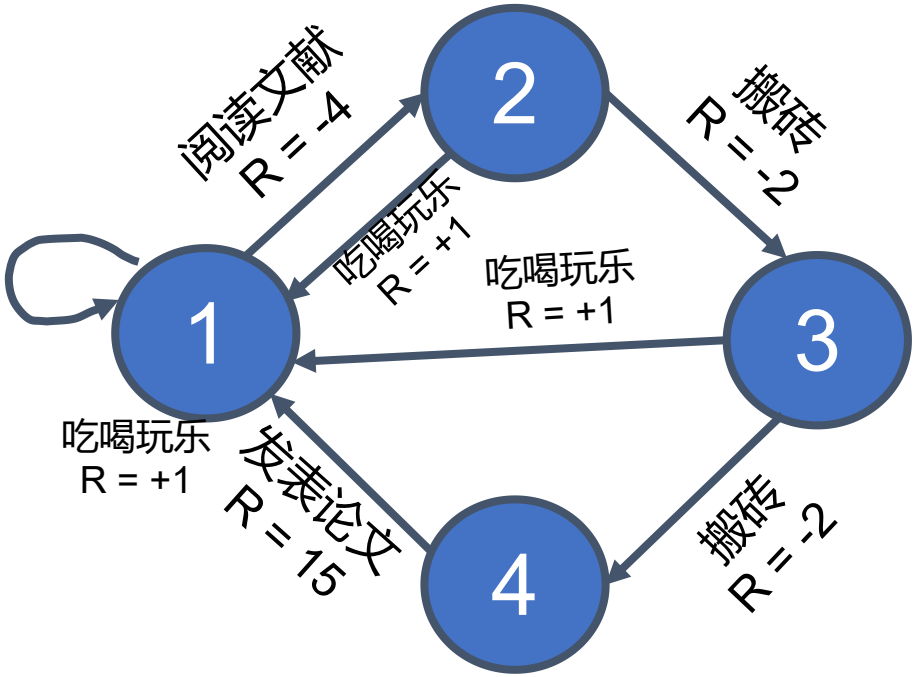
- 初始状态为顶点1
- 路径1：$1,1,1,\cdots.$
- 总回报 $1+\gamma^1+\gamma^2+\cdots=\frac{1}{1-\gamma}$



* 以上奖励值均为虚构

- 衰减因子可看成每天能继续获得回报的概率
- $1/(1-\gamma)$ 也是期望能获得回报的天数
- 上述“总回报”是选择路径1期望的总回报

动态决策问题：例子



* 以上奖励值均为虚构

- 初始状态为顶点1
- 路径2：1,2,3,4,1,2,3,4, ⋯
- 总回报

$$
\begin{aligned}
&-4-2\gamma^1-2\gamma^2+15\gamma^3-4\gamma^4-\cdots \\
=&(-4-2\gamma^1-2\gamma^2+15\gamma^3)\cdot\sum_{k=0}^{\infty}\gamma^{4k} \\
=&\frac{-4-2\gamma^1-2\gamma^2+15\gamma^3}{1-\gamma^4}
\end{aligned}
$$

- 每日回报 $\dfrac{-4-2\gamma^1-2\gamma^2+15\gamma^3}{1-\gamma^4}\Big/\dfrac{1}{1-\gamma}=\dfrac{15\gamma^3-2\gamma^2-2\gamma-4}{\gamma^3+\gamma^2+\gamma+1}$
- 当 $\gamma\approx 0$，平均每日回报约为 -4（次于路径1）
- 当 $\gamma\approx 1$，平均每日回报约为 $7/4$（优于路径1）

动态决策问题

- 给定有向图 G 和初始顶点 v_0
- 每个顶点表示某个状态，每条边 e 有回报 r_e（可以是负的）
- 定义路径 $p = v_0 v_1 v_2 \cdots$ 的回报

$$R(p) = r_{v_0 v_1} + \gamma \cdot r_{v_1 v_2} + \gamma^2 \cdot r_{v_2 v_3} + \cdots$$

  - 其中衰减因子 $0 < \gamma < 1$
- 目标：找到从 v_0 出发、回报 $R(p)$ 最大的路径 p

- Bellman 提出如何用“动态规划”解决该问题
  - （提示）分析最大回报的性质

Bellman 方程

- 给定顶点 v_0，设 $p = v_0 v_1 v_2 \cdots$ 是拥有最大回报的路径
- 令 $q = v_1 v_2 v_3 \cdots$，观察到

$$\begin{aligned} R(p) &= r_{v_0 v_1} + \gamma \cdot r_{v_1 v_2} + \gamma^2 \cdot r_{v_2 v_3} + \gamma^3 \cdot r_{v_3 v_4} + \cdots \\ &= r_{v_0 v_1} + \gamma \left(r_{v_1 v_2} + \gamma \cdot r_{v_2 v_3} + \gamma^2 \cdot r_{v_3 v_4} + \cdots \right) \\ &= r_{v_0 v_1} + \gamma \cdot R(q) \end{aligned}$$

- 因此，如果如果 p 是起点在 v_0 的最大回报路径，则 q 是在 v_1 的最大回报路径

- 令 $R(v)$ 是以 v 为起点路径的最大回报
- 有 $R(v) = \max_u [r_{vu} + \gamma \cdot R(u)]$ （Bellman 方程）

价值迭代

- Bellman 方程 $R(v) = \max_u [r_{vu} + \gamma \cdot R(u)]$
- 定义 $R^{(t)}(v) = \begin{cases} 0 & t = 0 \\ \max_u [r_{vu} + \gamma \cdot R^{(t-1)}(u)] & t > 0 \end{cases}$
  - （不同于 Bellman 方程，价值迭代是可实现的）
- 引理：令 $\epsilon_t = \max_u |R(u) - R^{(t)}(u)|$，则 $\epsilon_{t+1} \leq \gamma \cdot \epsilon_t$
- 证明：

$$
\begin{aligned}
& R(v) - R^{(t+1)}(v) \\
= & \max_u [r_{vu} + \gamma R(u)] - \max_u \left[r_{vu} + \gamma R^{(t)}(u) \right] \\
\leq & \max_u \left[r_{vu} + \gamma R(u) - r_{vu} - \gamma R^{(t)}(u) \right] \\
= & \gamma \cdot \max_u \left[R(u) - R^{(t)}(u) \right] \leq \gamma \cdot \epsilon_t
\end{aligned}
$$

价值迭代

```
Input: accuracy target δ
R[u,0]=0 for all u
R_max = max_e |r_e| / (1-γ)
T = ⌈log_γ(δ / R_max)⌉

for t = 1,2,...,T
   for v = 1 to n
      R[v,t] = max_u (r_uv + γ·R[u,t-1])
```

R_{max} 是 R 的某个上界
由 $\epsilon_{t+1} \leq \gamma \cdot \epsilon_t$ 和 $\epsilon_0 \leq R_{max}$，得 $\epsilon_T \leq \delta$

运行时间：$O(mT) \approx O(m \log(\frac{1}{\delta})/(1-\gamma))$
所以，在 γ 接近0时，运行时间极快

Open Question：当 γ 接近 1时能求解多快?

超级马里奥兄弟

- 给定
  - 本关完整信息（横版右移N个像素通关）
  - 屏幕大小（w×h）
  - 状态 C
    - 本关进度 0 ≤ n ≤ N
    - 角色位置/速度/加速度等信息 O(1)
    - 障碍和怪兽位置等信息 m^{wh}
    - 当前得分 0 ≤ score ≤ S 和剩余时间 0 ≤ time ≤ T
  - 转移函数 δ: (C, action) → C’
    - Action: nothing ↑ ↓ ← → B A press/release

source: wikipedia.org

超级马里奥兄弟

- 子问题 $DP(C)$：从状态C开始，最快通关时的剩余时间
  - $nm^{wh}ST$ 个子问题
  - 总时间 T 是个常数

- 递推方程：最大化剩余时间

$$DP(C) = \begin{cases} T & n == N \\ -\infty & \text{挂掉或者倒计时结束} \\ max_A\ (DP(\delta(C,A)) - t_A) & \text{对可能的行动}A \end{cases}$$

source: wikipedia.org

DP and Approximate DP

DP is all about **multistage decision processes** with terminology floating around terms such as **policy, optimal policy or expected reward** making it very similar to Reinforcement Learning which in fact is sometimes called **approximate dynamic programming**.

https://int8.io/bellman-equations-reinforcement-learning/

内容总结

- 若干动态规划例子
- 动态规划的特征
  - 最优子结构
  - 重叠子问题
- 递推的步骤
- 有向无环图的动规
- 树/分解树的动规
- 状态表示的例子
- 动规规划的历史