局部利益最大化

贪心策略

回忆

- 0-1背包问题：有n个物品，每个物品i有它的重量w_i和价值v_i，现在有一个可以承重W的背包，问如何选择那些物品装入背包使得他的价值最大，而且不超过背包承重的限制。

有n个数对(w_i, v_i) $(i = 1 \cdots n)$，如何选择这些数对的一个子集S，使得$\sum_{i \in S} w_i < W$前提下，$\sum_{i \in S} v_i$的值达到最大

改变题目的部分条件

如果我们可以将物品分割，选择把物品的一部分放入背包？

不会出现装不满的情况

比如我们现在要装的不是n件物品，而是发现有n堆的金币、银币、宝石、珍珠。。。

优先选择重量最轻的物品

优先选择价值最大的物品

优先选择性价比最高的物品

贪心策略

贪心算法总是作出在当前看来最好的选择，
希望得到的最终结果也是整体最优的

贪心策略：局部最优 —希望得到→ 整体最优

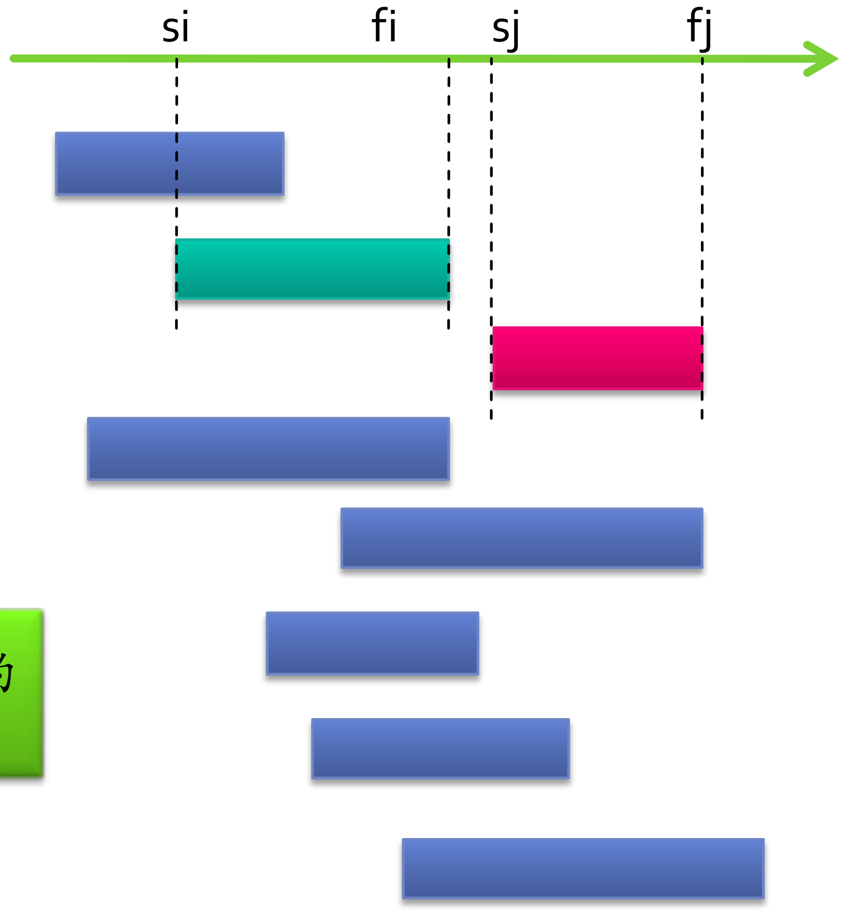
需要证明局部最优能够导致整体最优

动态规划：整体最优 ←相互关系→ 局部最优

需要找出整体最优和局部最优的最优子结构

活动安排问题

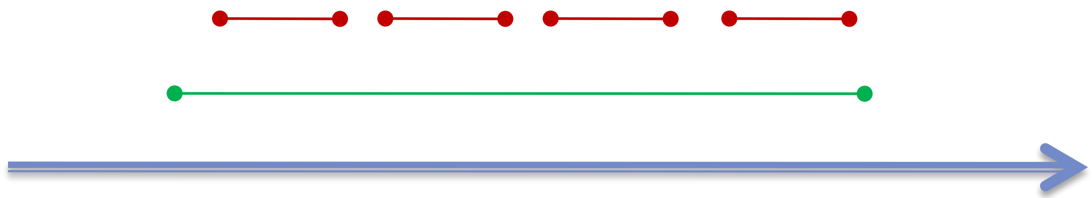
设有n个互斥的活动要使用同一资源，每个活动都有一个起始时间si和一个结束时间fi.两个活动i、j,如果满足si≥fj或者或sj≥fi，则称相容的



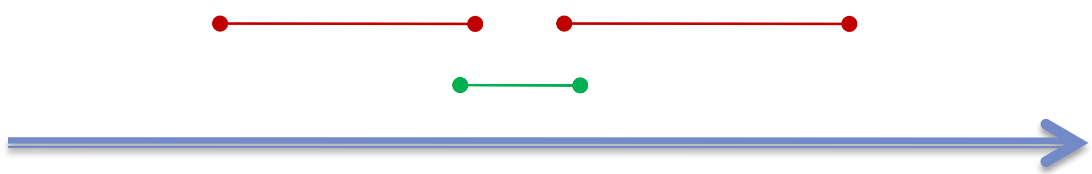
能否设计一种算法，使得有尽量多的活动使用这个资源

选择贪婪策略

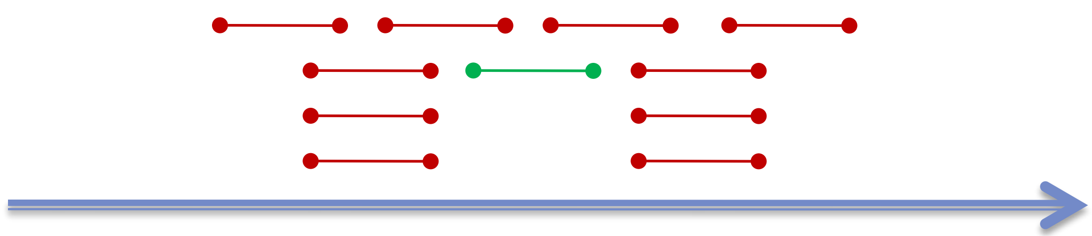
优先选择最早开始的



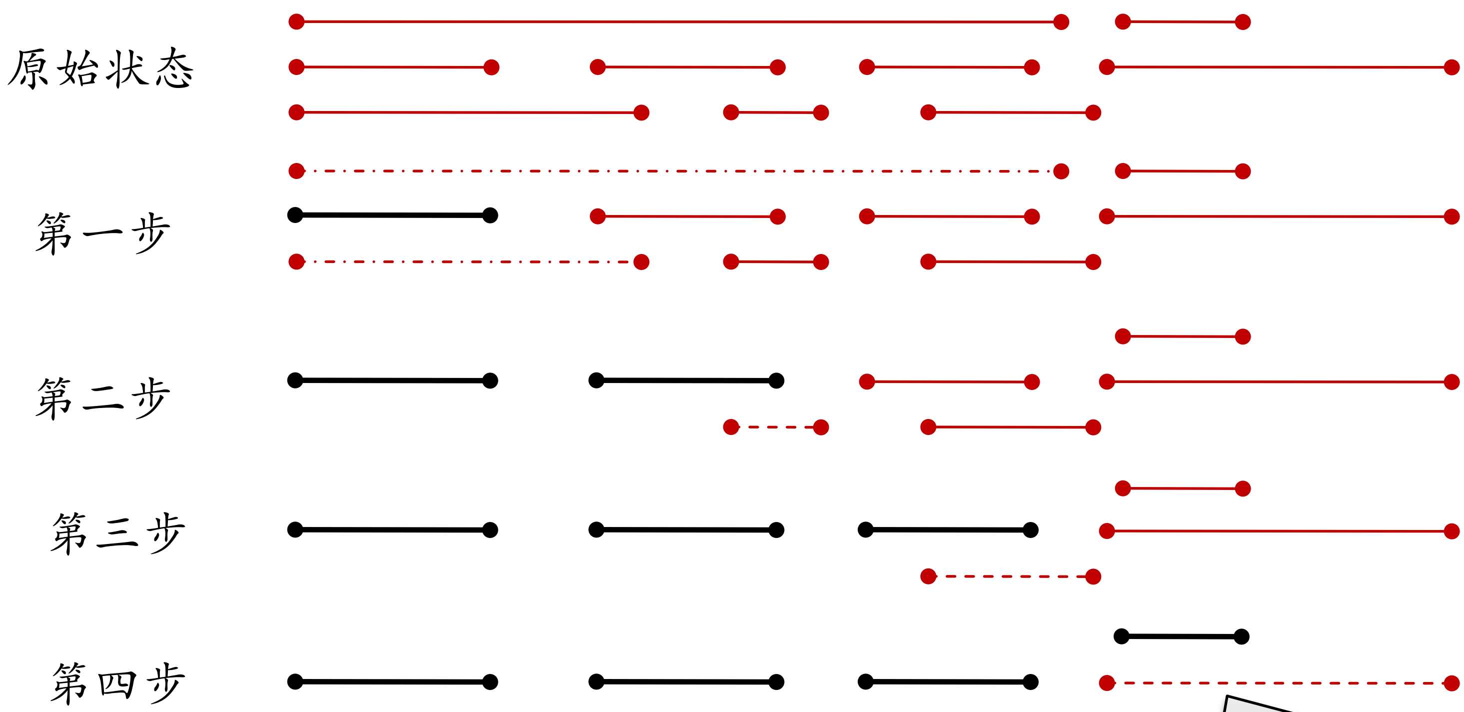
优先选择占用时间最短的



优先选择和其他互斥最少的



优先选择最早完成的活动，这样可以给安排其他活动剩余更多的时间



贪心算法只是希望得到的解是最优解，而并不能得到所有的最优解

如果最后一个选择此活动也是一个最优解

算法分析

由此得到的集合是相容的

假设S是我们得到的集合，O是任意一个最优解，S中的第k个活动的结束时间早于O中第k个活动的结束时间

我们记i_k为S中第k个活动，j_k为O中第k个活动，用数学归纳法
显然$f(i_1) <= f(j_1)$
假设$f(i_{k-1}) <= f(j_{k-1})$，我们证明$f(i_k) <= f(j_k)$
由于$f(i_{k-1}) <= f(j_{k-1})$，所以集合中与S的前k-1个相容个活动一定包含与O的前k-1个相容个活动，根据我们选择最早结束的相容活动，所以$f(i_k) <= f(j_k)$

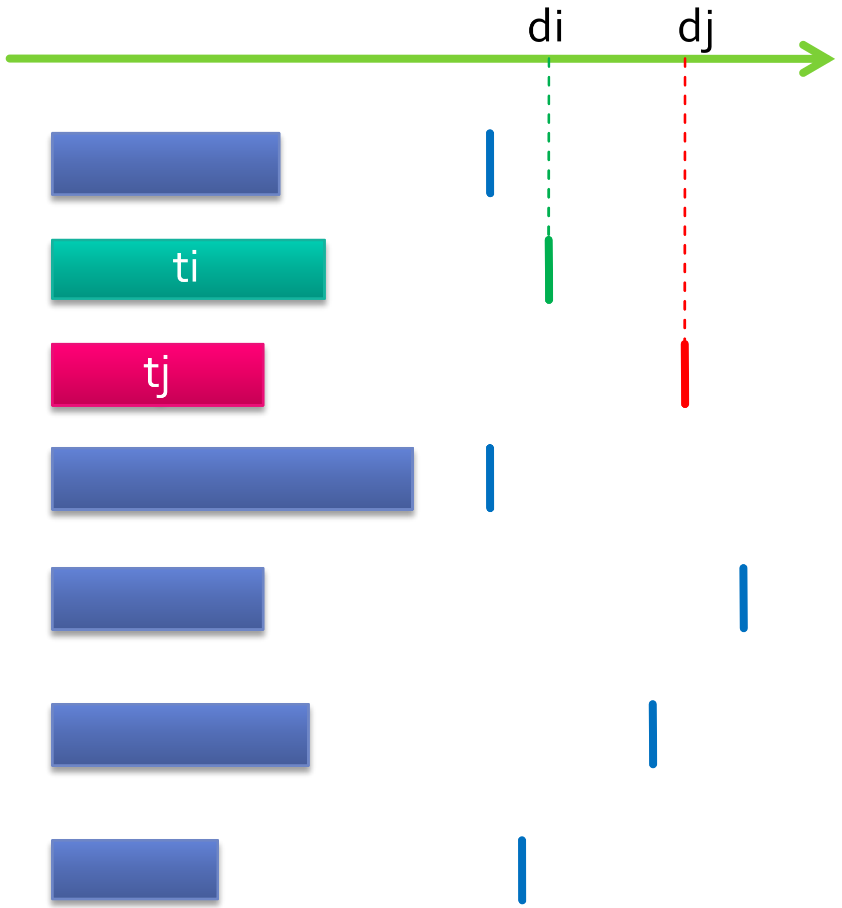
贪心法的领先概念：贪心法的每一步做的都比一个最优解好，所以他也是最优解

贪心算法返回一个最优解

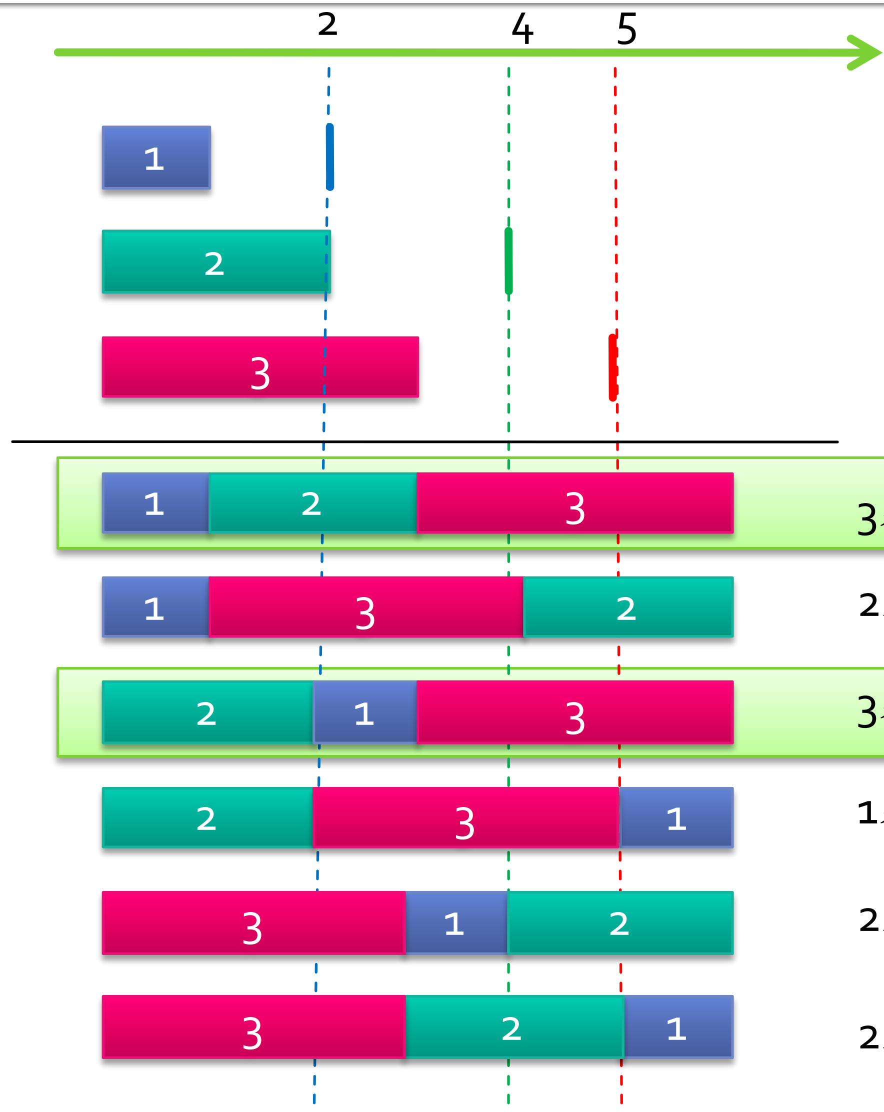
活动安排问题变形

设有n个互斥的活动要使用同一资源，每个活动都更加灵活，我们不规定他的具体起始时间和终止时间，而是确定一个最后期限(截止时间deadline)di，要求在这个期限内完成，第i个活动需要ti的连续时间来完成

能否设计一种算法，使得所有被延迟的活动中**最大的延迟时间最小**



最大最小问题



能否设计一种算法，使得所有被延迟的活动中**最大的延迟时间最小**

这两种方法有什么特点，或者其中一个有什么规律

3延迟了1个时间单位

2延迟了2个时间单位

3延迟了1个时间单位，1延迟了1个时间单位

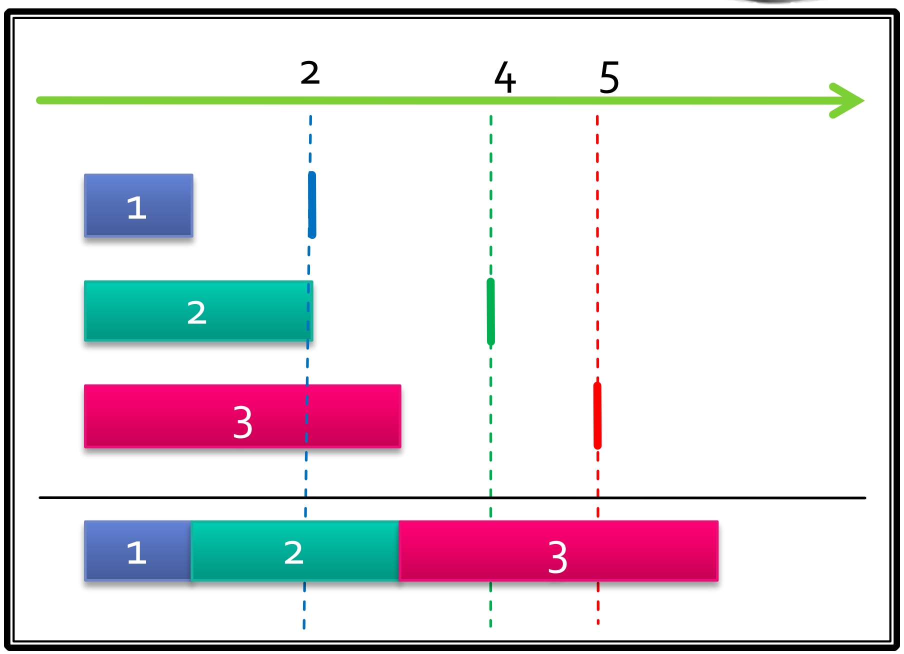
1延迟了4个时间单位

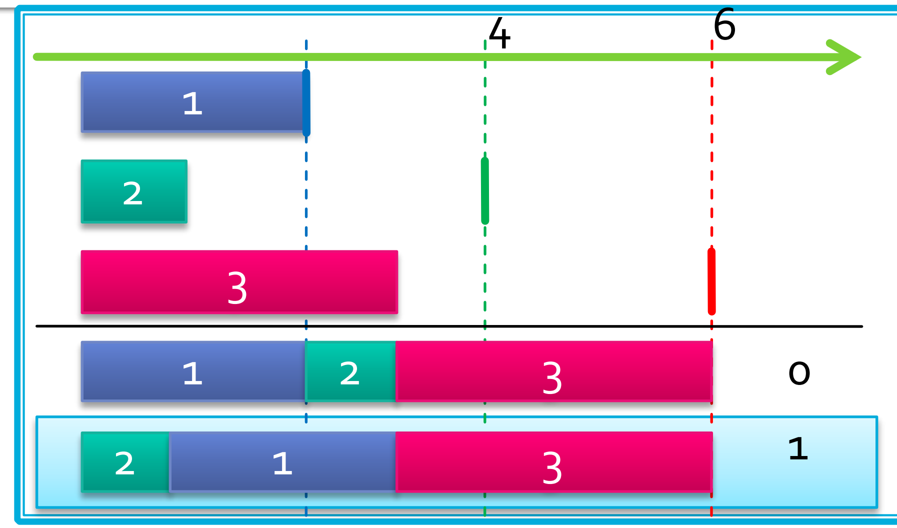
2延迟了2个时间单位，1延长了2个时间单位

2延迟了1个时间单位，1延长了4个时间单位

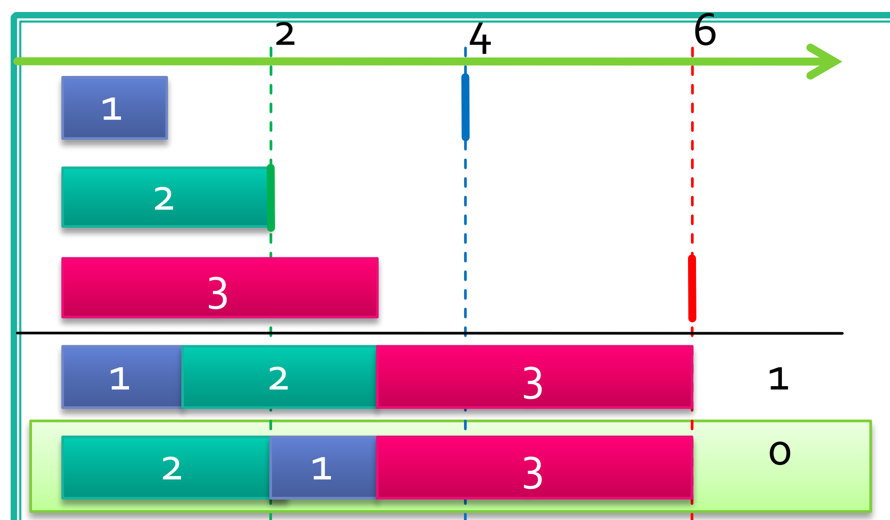
选择贪心策略

最小长度优先





最早截止时间优先



算法分析

存在一个没有空闲时间的最优算

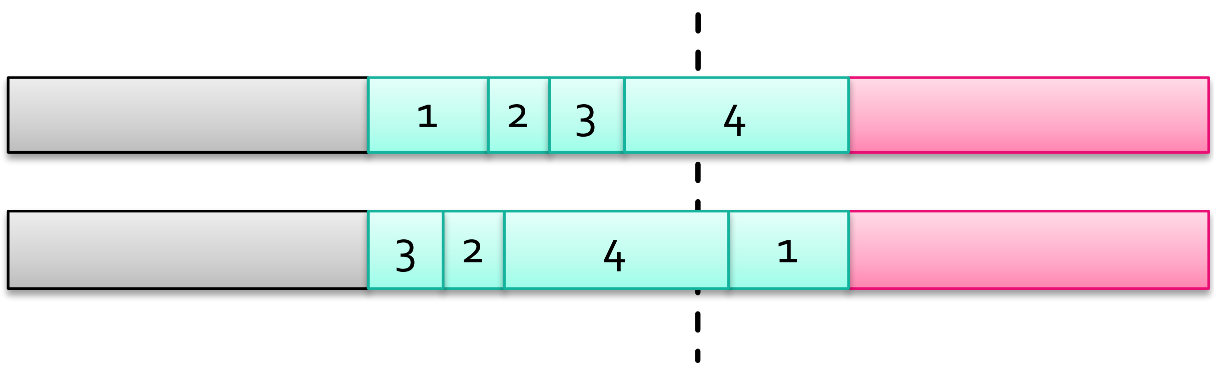
反证法

所有没有逆序也没有空闲时间的调度有相同的最大延迟

我们将所有任务按结束时间由小到大排序，如果di<dj，而dj又被安排在di之前，我们则称这个调度有一个逆序

这样的调度可能有很多个，因为具有相同结束时间的活动可能有很多个

有相同结束时间的活动被连接在一起安排，这些活动的顺序不影响他们产生的最大延迟延迟，最大延迟只由这些活动中最后完成的那个的完成时间决定



算法分析

存在一个没有空闲时间的最优算法

反证法

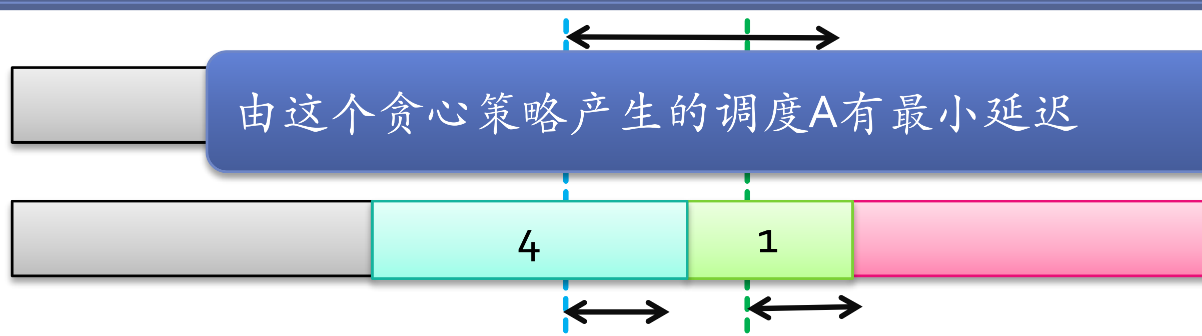
所有没有逆序也没有空闲时间的调度有相同的最大延迟

存在一个既没有逆序也没有空闲时间的最优调度

交换论证：从任意一个没有空闲时间的最优调度；在不增大最大延迟的前提，下将其转化成一个没有逆序的调度

设O是一个不具有空闲的最优调度
如果O有一个逆序，那么存在一对任务i和j，dj<di，而且j被直接安排在i后面
交换i和j，我们可以减少一个逆序
这个新的被交换得到的调度的最大延迟不大O的最大延迟

由这个贪心策略产生的调度A有最小延迟



证明贪心算法的两种方法

贪心法的领先概念：贪心法的每一步做的都比一个最优解好，所以他也是最优解

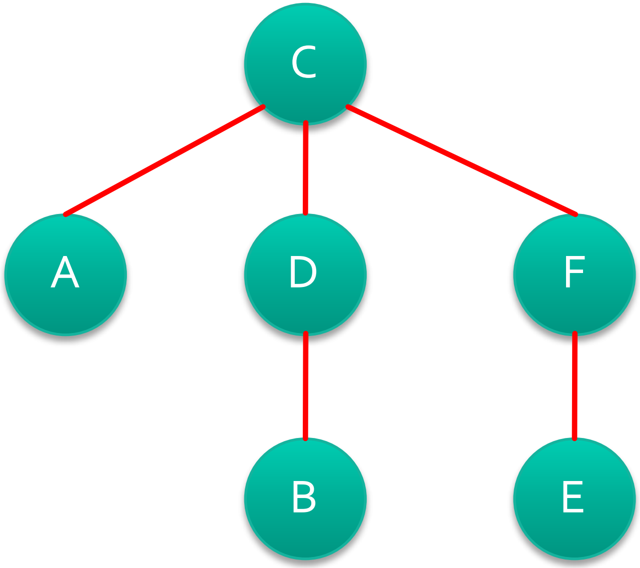
交换论证：从任何一个最优解出发，在保证最优解的情况下将其结果变为和由贪心算法生成的结果相同

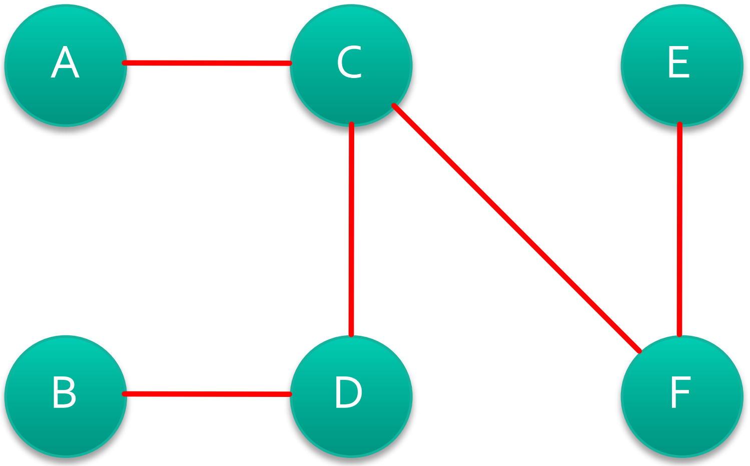
数据结构中哪些算法包含贪心策略

温故而知新

树和图

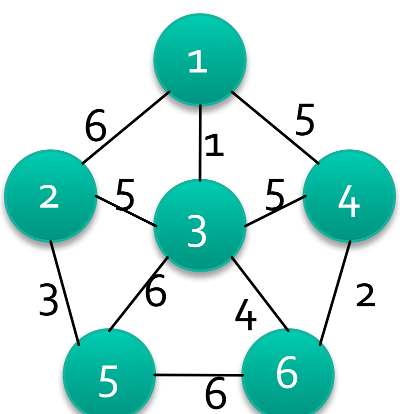
树是一个连通且无环的无向图

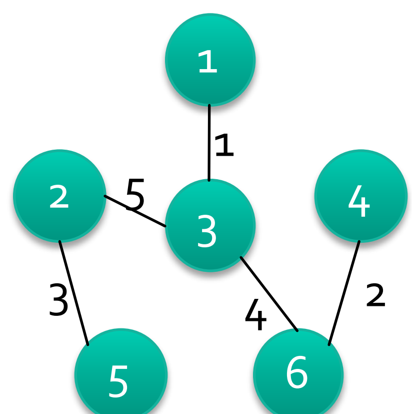




最小生成树问题

设G=(V,E)是一个无向连通带权图，即一个网络，E中每条边(v,w)的权为c[v][w]，如果G的一个子图G'是一颗包含G的所有定点的树，则称G'为G的生成树，生成树上各边权重的总和称作该生成树的耗费，如何在G的所有生成树中找到耗费最小的生成树?





选择某些边或者删除某些边

使用贪婪策略

策略1：起初没有任何边，将边按照权值由小到大插入到图中，保证新插入的边不构成环，否则跳过此边

Kruskal算法

策略3：起初集合A中只有一个点s，依次选择和A中的点直接相连的所有点中边的权值最小的点，把这个点连同边一起插入到A中，知道A中包含所有的点

Prim算法

策略2：起初保留图中所有的边，在保证图依然是连通的前提下根据边的权值由高到低删除边

逆删除算法

算法分析

Kruskal算法

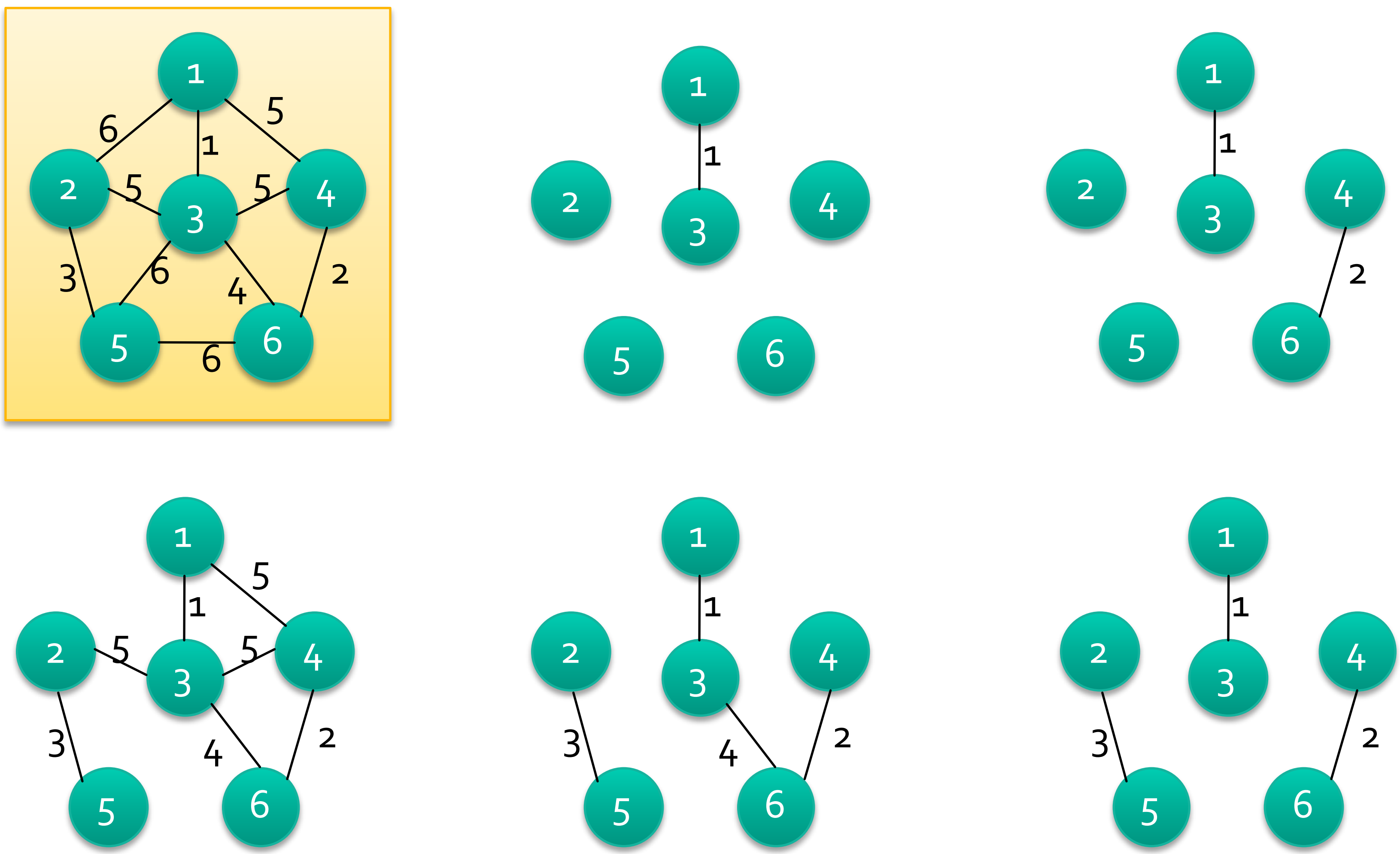
Prim算法

逆删除算法

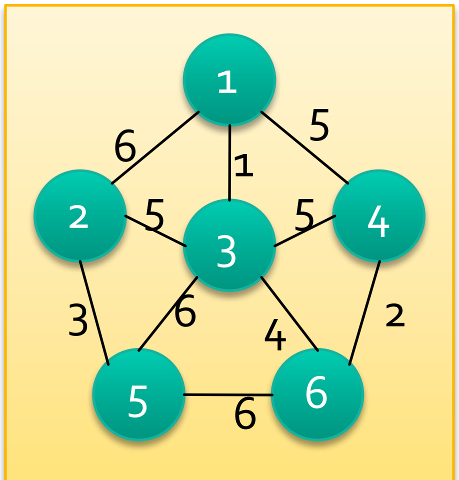
令S是任意节点子集，它既不是空集也不是全集V，令e=(v,w)是一端在S中，另一端在V-S中的最小权重边，那么没棵最小生成树中都包含e

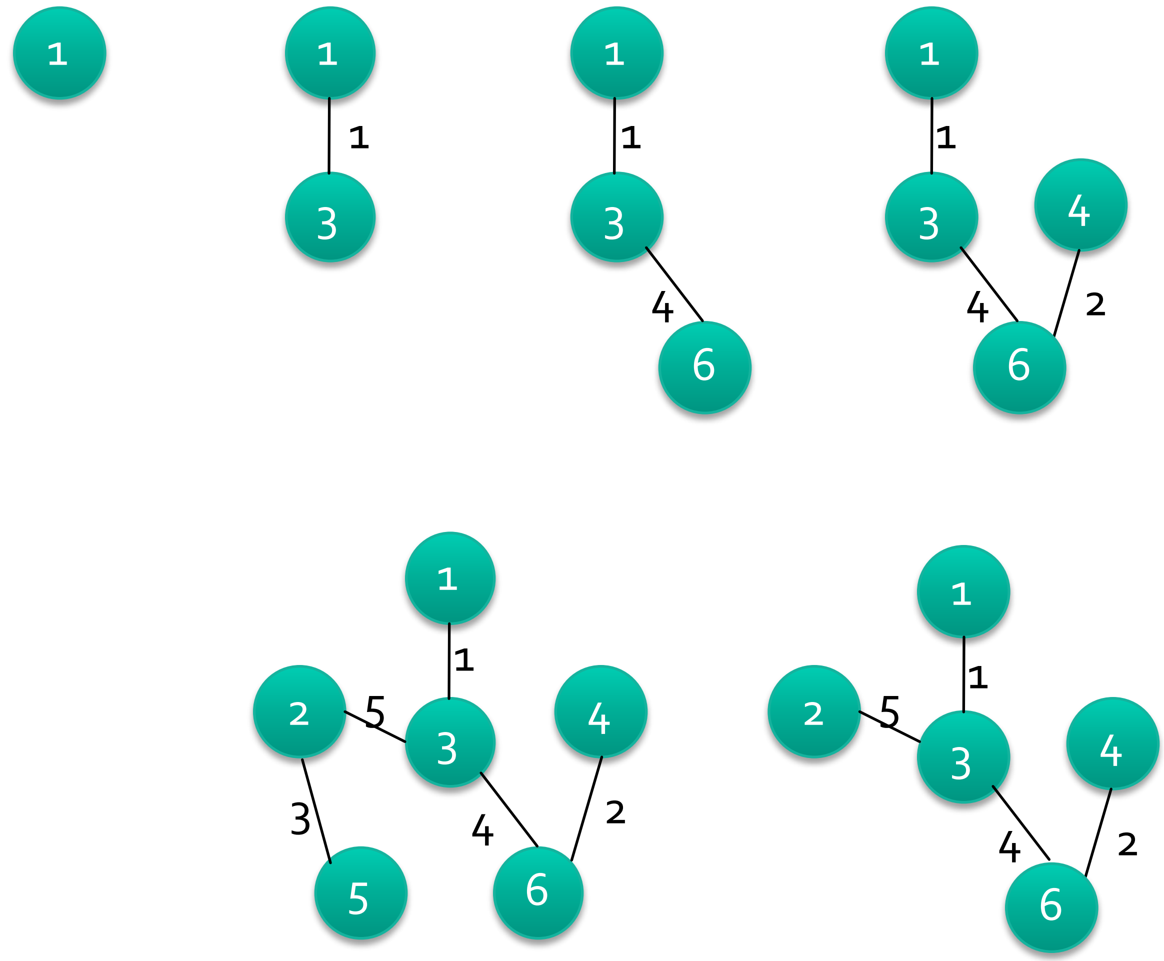
对于权值完全不同的图，令C是G中的一个环，令边e=(v,w)是属于C中最贵的边，那么e不属于G的任何一颗最小生成树中。

Kruskal算法

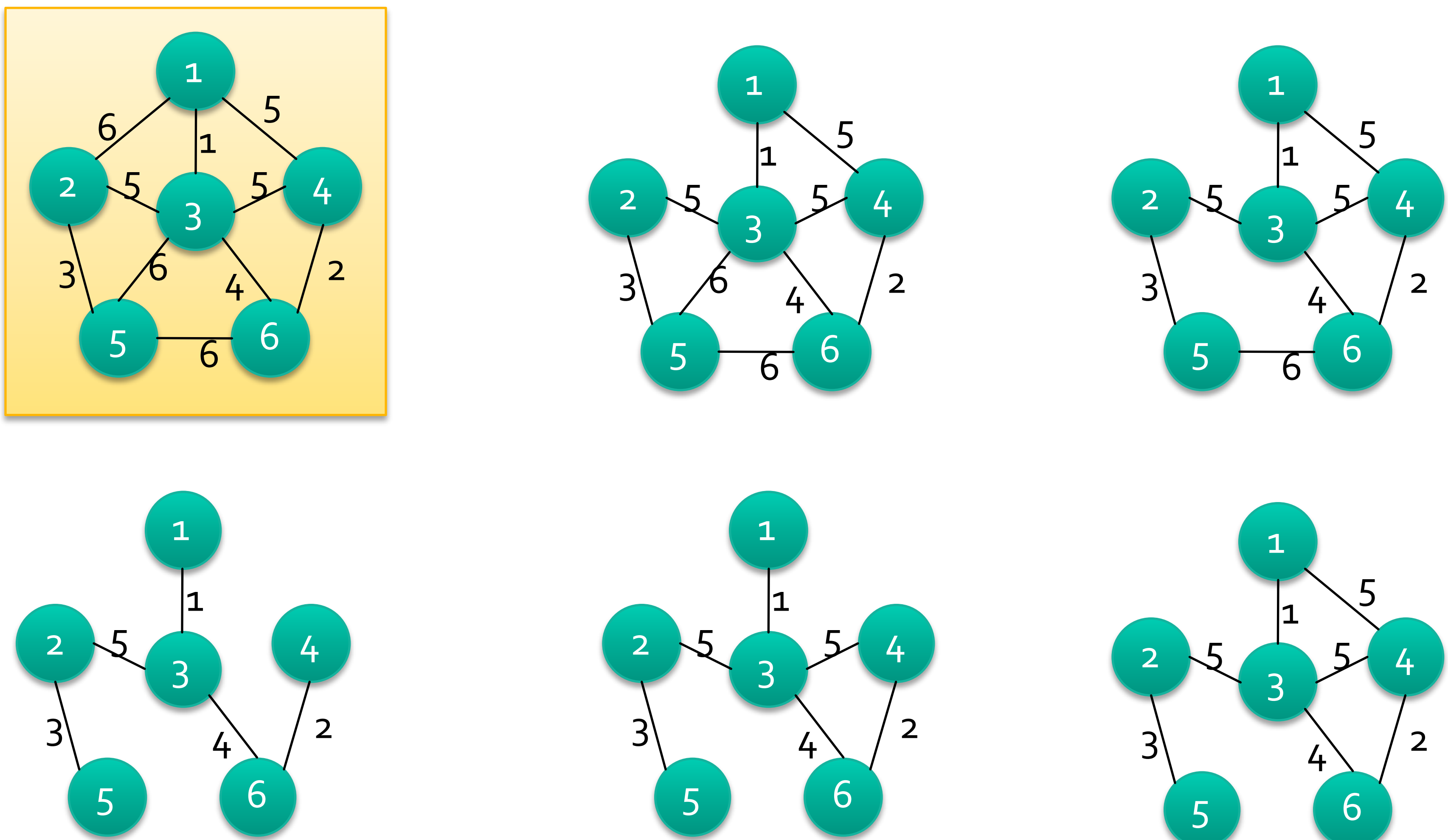


Prim算法





逆删除算法

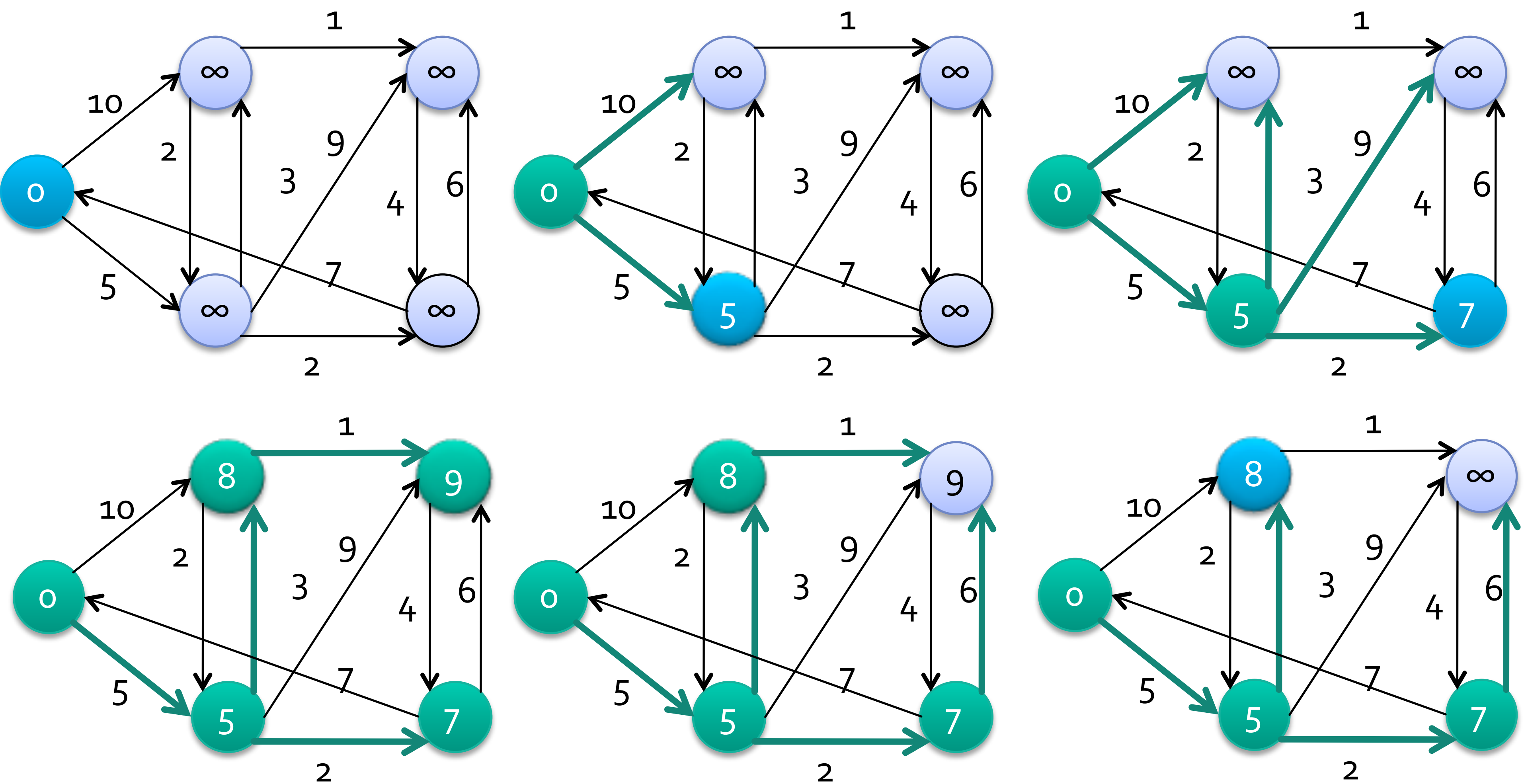


单源最短路径

在一个带权有向图中，给定一个点，称为源，现在要计算从源到其他各个顶点的最短路径长度

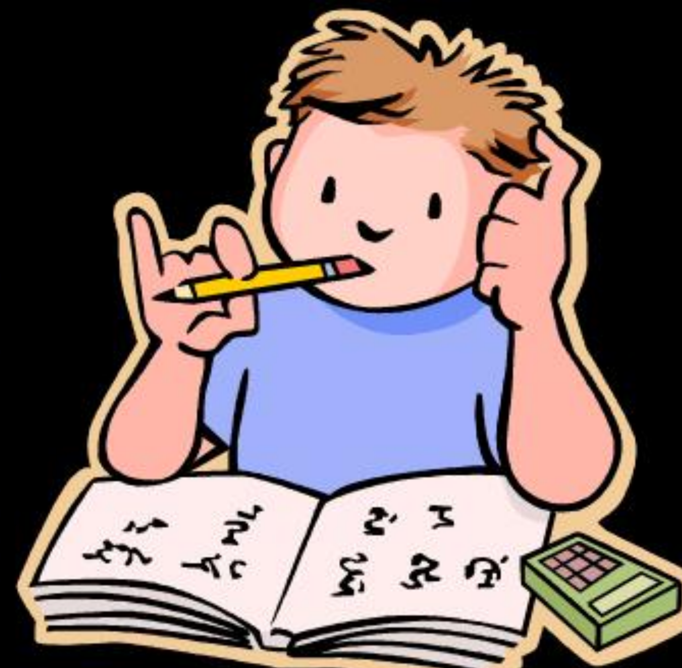
Dijkstra算法：S起初只含有源s，通过如下的方法进行扩充；假设S中含有所有已经计算路径d(p)的点p；然后考虑V-S中所有和S中的点直接相连的点，假设V-S中的v与S中的u直接相连，那么他们的特殊长度记为d(v)=d(u)+w(u,v)，并将特殊长度最小的点划入到S中

计算的推进方式与Prim算法类似



1 10 2 3 9 4 6 5 7 2 0 ∞ ∞ ∞ ∞
1 10 2 3 9 4 6 5 7 2 0 ∞ ∞ 5 ∞
1 10 2 3 9 4 6 5 7 2 0 ∞ ∞ 5 7
1 10 2 3 9 4 6 5 7 2 0 8 9 5 7
1 10 2 3 9 4 6 5 7 2 0 8 9 5 7
1 10 2 3 9 4 6 5 7 2 0 8 ∞ 5 7

Huffman Code



之所以不讲，是因为。。。。

钓鱼问题(Gone Fishing)

ACM/ICPC Regional Contest East Central North America 1999. Problem G

问题描述

- 在一条水平的路边，有n个钓鱼池，从左到右编号为1、2、3、…、n。佳佳有H小时的空余时间，他希望用这些时间调到尽量多的鱼。他从湖1出发向右走，有选择的在一些湖边停留一定的时间钓鱼，最后在某一个湖边结束钓鱼。佳佳测出从第i个湖到第i+1个湖需要走5*Ti分钟的路，还测出在第i个湖边停留，第一个5分钟可以钓到鱼Fi，以后再每钓5分钟鱼，鱼量减少Di。请编程求出能钓最多鱼的方案。

问题的简化

- 钓5分钟鱼称为钓1次鱼
- 枚举所有他可能走的湖波数X，即从1走到X，则路上花去的时间为 $\sum_{i=1}^{X} Ti$
- 在这种情况下我们可以不用考虑在湖间移动的时间，可以认为是“瞬间转移”
- 即可认为在每一时刻都可以从湖波1到X中任选一个钓1次鱼

```
Time_trans = 0;
Fishmax= 0;
For (X=1;X<=N;x++) {
    time_fish = H – time_trans;
    fish = fishing(X,time_fish);
    if (fish>fishmax)   fishmax = fish;
    time_trans +=Ti
}
```

采取贪心策略

贪心策略：每次选择可以钓最多鱼的湖

在time_fish单位时间内
在1至X的湖中钓鱼
fish = fishing(X,time_fish);

我们要在湖1至4中钓7个时间单位的鱼

71

湖中鱼的数量与在该湖钓鱼的次数有关，而与总的钓鱼次数无关

1 F=10;D=1

2 F=9;D=1

3 F=13;D=2

4 F=10;D=2

O=13;
F=11

O=11;
F=9

O=10;
F=9

O=10;
F=8

O=9;
F=8

O=9;
F=8

O=9;
F=7

1 O=19;T=2

2 O=9;T=1

3 O=33;T=3

4 F=10;T=1

时间复杂度

- 要对需要钓的湖进行枚举，n种可能
- 如果需要钓k个单位时间的鱼，k次选择
- 每个单位时间选择的时间复杂度为O(n)

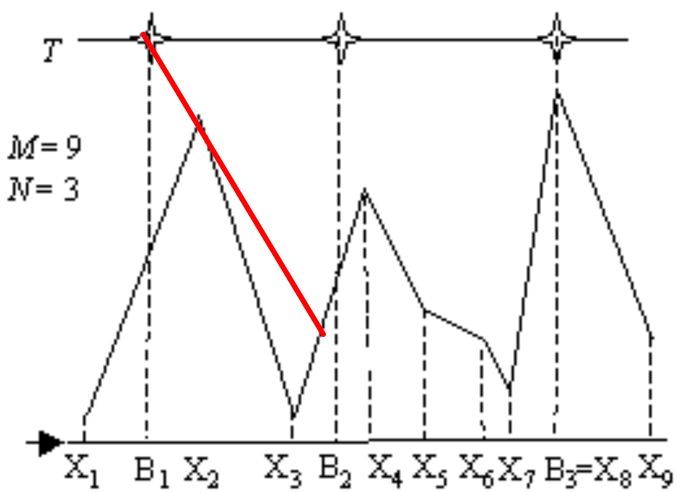
时间复杂度$O(kn^2)$

照亮的山景
(Enlightened Landscape)

Central-European Olympiad in Informatics(CEOI) 2000, Day 2 Problem 3

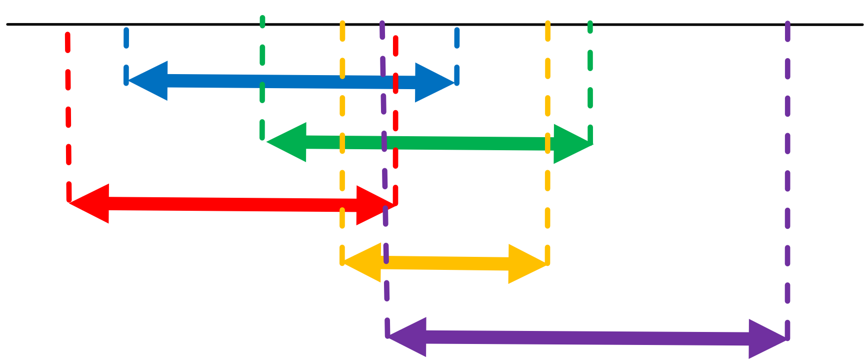
问题描述

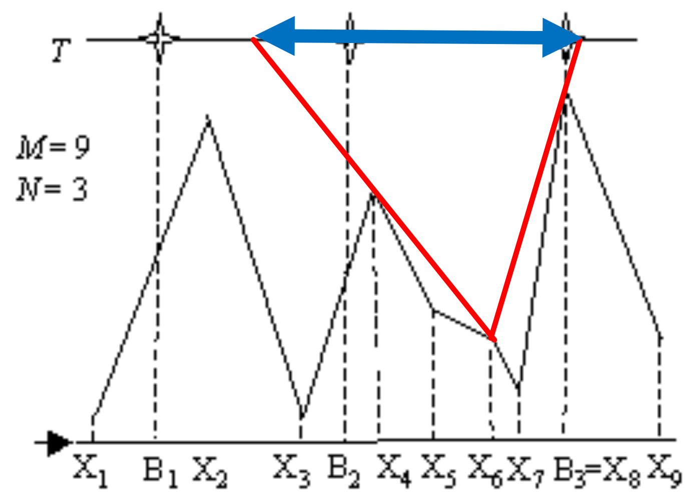
- 在一片上的上空，高度为K处有N个处于不同水平位置的灯泡；如果山的边界上的某一点与灯i的连线不经过山上的其他点，我们称i可以照亮该点。开尽量少的灯，使得整个山景都被照亮。



问题转化

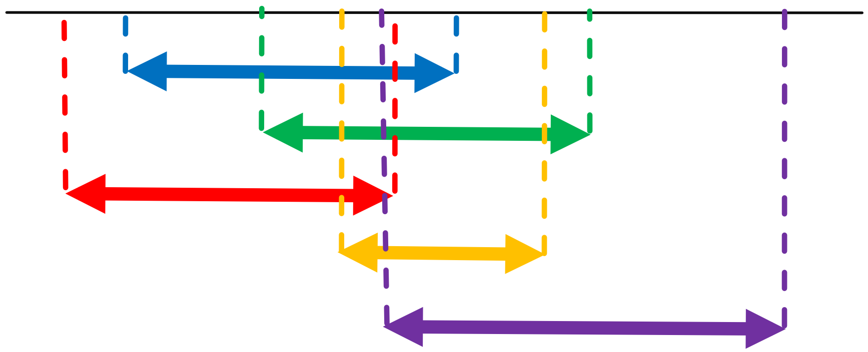
- 照亮山与照亮山个M个转折点等价
- 每个灯得照射范围可能是不连续的M的点的集合
- 我们换个角度考虑，山的每个转折点能被那些范围内的灯照亮。如果两盏灯能都能照亮一个转折点i，那么这两个灯之间的灯都可以照亮这个转折点I
- 能照亮一个转折点的所有灯构成一个连续区间(li,ri)





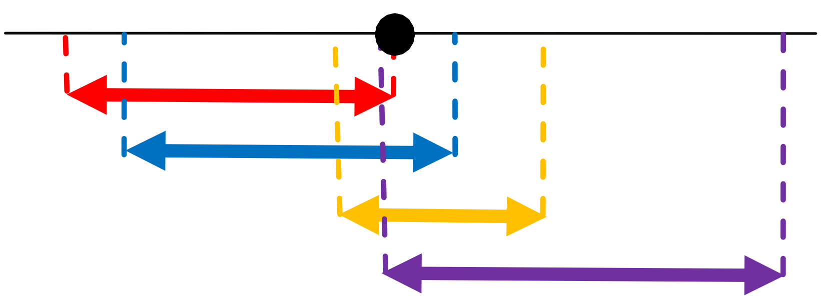
问题转化与应用贪婪策略

给定M个区间，选出尽量少的点，使得每个区间至少有一个点被选出来



可以出去包含其他区间的区间

将区间的起始位置由左向右排序，易见区间的终止位置也是从左向右排序的



贪婪策略：取第一个区间最右边的点，使得有更多的区间得到满足

Color a Tree （PKU 2054）

问题描述

Bob is very interested in the data structure of a tree. A tree is a directed graph in which a special node is singled out, called the "root" of the tree, and there is a unique path from the root to each of the other nodes.

Bob intends to color all the nodes of a tree with a pen. A tree has N nodes, these nodes are numbered 1, 2, ..., N. Suppose coloring a node takes 1 unit of time, and after finishing coloring one node, he is allowed to color another. Additionally, he is allowed to color a node only when its father node has been colored. Obviously, Bob is only allowed to color the root in the first try.

Each node has a "coloring cost factor", Ci. The coloring cost of each node depends both on Ci and the time at which Bob finishes the coloring of this node. At the beginning, the time is set to 0. If the finishing time of coloring node i is Fi, then the coloring cost of node i is Ci * Fi.

For example, a tree with five nodes is shown in Figure-1. The coloring cost factors of each node are 1, 2, 1, 2 and 4. Bob can color the tree in the order 1, 3, 5, 2, 4, with the minimum total coloring cost of 33.

Given a tree and the coloring cost factor of each node, please help Bob to find the minimum possible total coloring cost for coloring all the nodes

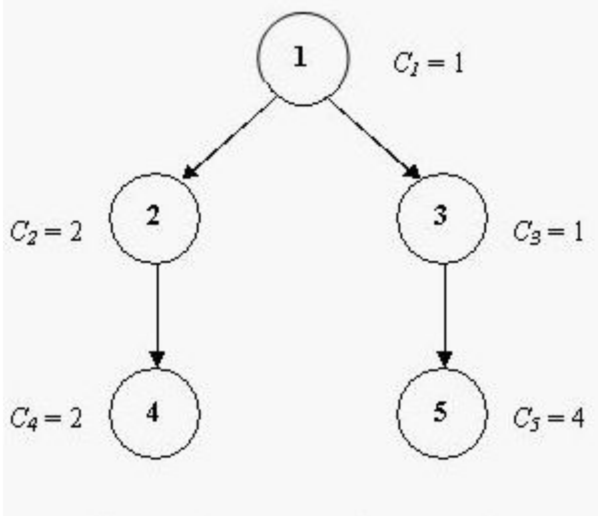


Figure-1. A tree with five nodes

问题解释

- 给你一个有N个节点的树，每个节点都有一个权值。现在你要给这棵树染色，染色的规则是：
  - 根节点可以随时给它染色
  - 若要给某个节点染色，必须已经给它的父节点染好了色
- 每次染色的代价是（T*C[J]）T代表第几次染色，C[J]是点J的权值。
- 求给整棵树染色的最小代价

分析

- 这道题就是要求Σ(i *Di) (i = 1 .. n) 的值最小，{Di} 是节点费用{Ci}的一个排列，同时要满足父节点要出现在子节点前面。
- 如果没有父节点出现在子节点前面这个限制，那么答案很明显。当{ Ci }按降序排列的时候，Σ的值是最小的。
- 当有这个限制的时候情况也是类似的。考虑某一个可行解，就是{ Ci }的某一个排列。找到其中的最大值，比如为Ck，它有一个父节点比如Cp。显然Cp要出现在Ck之前。更进一步，Cp就应该出现在Ck的前一个位置。只有这样才有可能Σ的值最小。不然我们可以将Ck位置向前移动，得到一个更小的Σ值，并且不破坏上面的约束。
- 既然Cp就出现在Ck的前一个位置，那么它们其实就是连在一起的，可以最为一个整体来看。这样问题的规模就由n减小到n-1。然后重复这一过程，直到所有的位置都确定下来。
- 寻找Ck，就是贪心的过程

具体算法

- 首先对模型进行扩充：每个结点有一个权值Ci，和一个染色所需要的时间Si，而且每个结点还有一个生成序列，表示依次是哪些结点合并而成。
- 所以我们的贪心思想也有所变化，尽量让Ci/Si比值最大的节点先染色。这很好理解，如果将合并了的结点拆开来看，事实上可以认为是Si个权值为Ci/Si的结点。
- 根据合并的方式，我们得到了一个算法：
  - 1.令所有节点S值均为1，每个节点生成序列中仅有一个元素，即为它本身。
  - 2.若树中只剩一个结点，则输出这个这个结点的生成序列。
  - 3.取出Ci/Si值最大的非根结点Max。
  - 4.将Max和其父亲合并，新合并出的结点Union的各个参数为：Cunion=CMax+CPa(max)，Sunion=SMax+SPa(Max)，同时Union的生成序列为Pa(Max)的生成序列与Max的生成序列连接而成。
  - 5.重复2~4步

为什么比较Ci/Si?

证明:
比较同一个父节点的两个儿子i和j,
i的节点序列为（R1，R2，……Rn），
j的节点序列为（Q1，Q2，……Qm），

Ci=sigma(C[Rk])，Cj=sigma(C[Qk])，Si=n，Sj=m。
假设时间已行进到t，则先染i节点的权值和为
sigma(C[Rk]*(t+k-1))+sigma(C[Qk]*(t+n+k-1))……①,
先染j节点的权值和为
sigma(C[Qk]*(t+k-1))+sigma(C[Rk]*(t+m+k-1))……②,

②-①得sigma(C[Rk]*m)-sigma(C[Qk]*n),
则上式>0等价于
sigma(C[Rk])/n>sigma(C[Qk])/m
即 Ci/Si>Cj/Sj
证毕。