PKU-04833060
算法设计与分析『实验班』

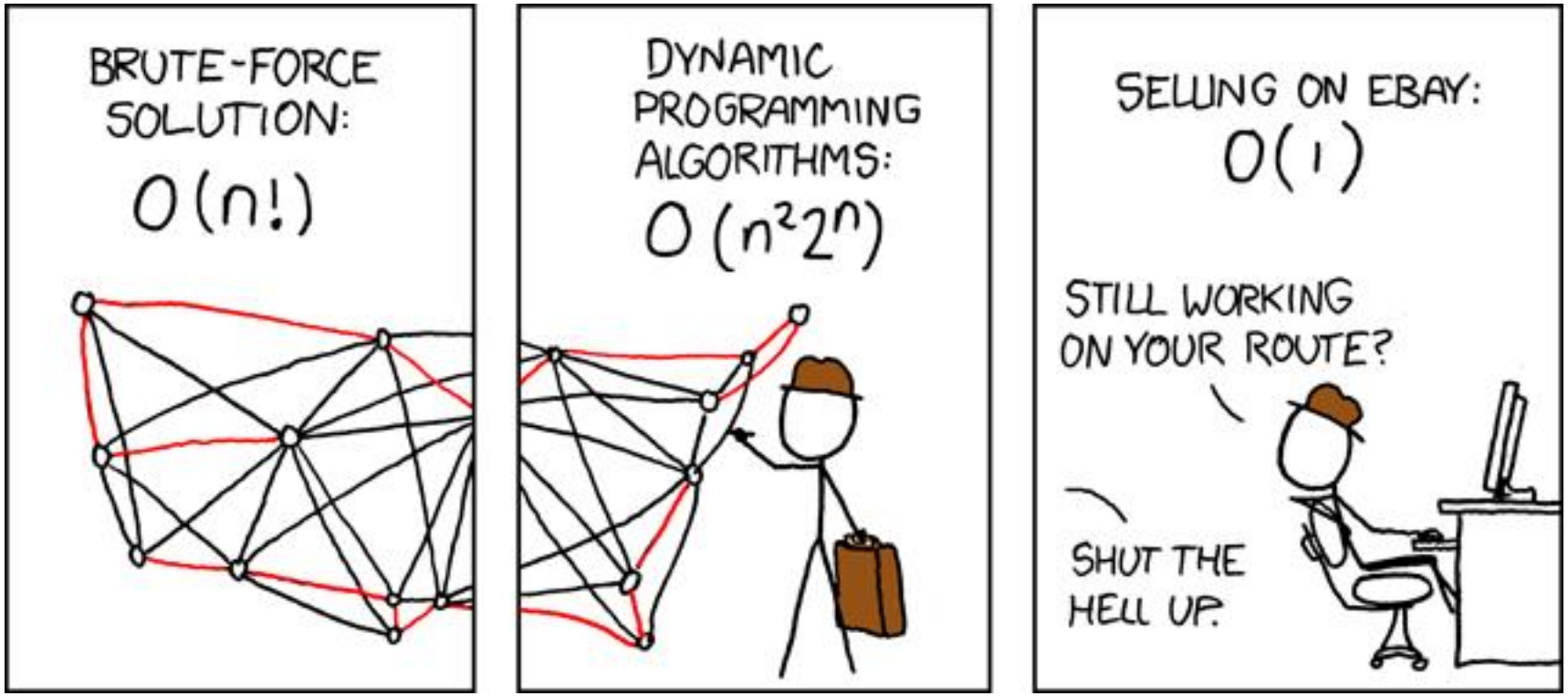
第10讲 问题的复杂度分析 (上)

罗国杰

gluo@pku.edu.cn

2025年春季学期

Travelling Salesman Problem



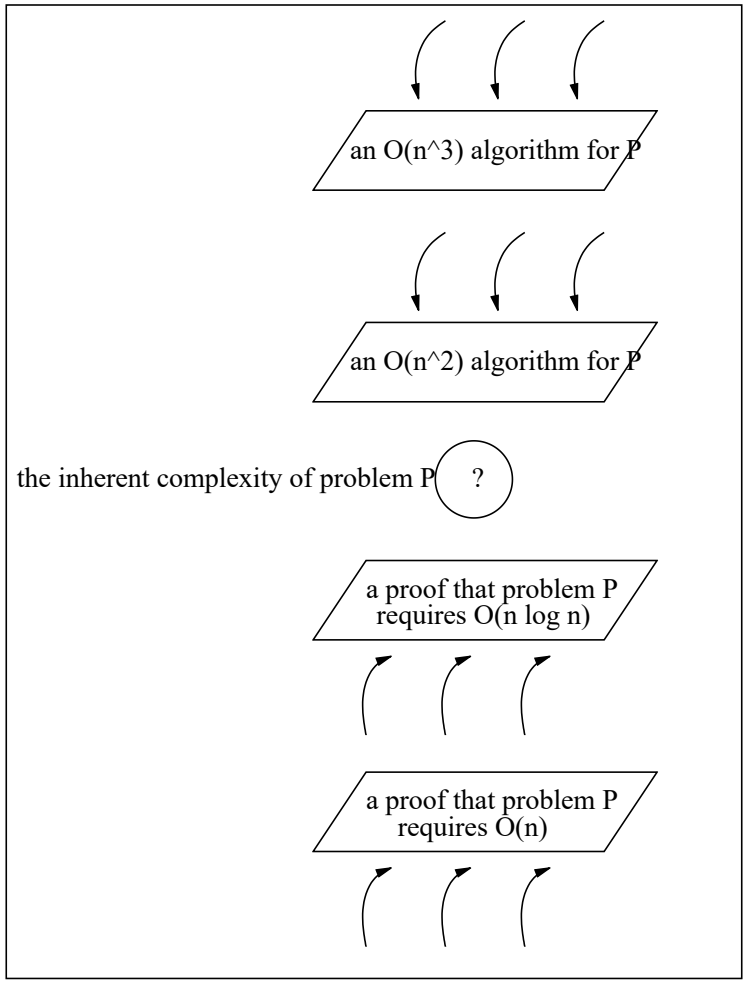
https://xkcd.com/399/

问题的复杂度

≈

“最优”算法的复杂度

寻找最优算法的途径



http://www.cs.ucc.ie/~dgb/courses/toc/handout28.pdf

寻找最优算法的途径

(设计算法) 设计算法 A 并求 $W_A(n)$，得到求解该问题算法类最优算法的复杂度上界

(分析下界) 寻找函数 $F(n)$，使得对任何算法都存在规模为 n 的输入使得算法在这个输入下至少要做 $F(n)$ 次基本运算，得到该算法类最坏时间复杂度的一个下界

(判断) 如果 $W_A(n) = F(n)$ 或 $W_A(n) = \Theta(F(n))$，则 A 是最优的.

(判断) 如果 $W_A(n) > F(n)$，A 不是最优的或者 $F(n)$ 的下界过低.

(重新设计算法) 改进 A 或设计新算法 A' 使得 $W_{A'}(n) < W_A(n)$.

(重新分析下界) 重新证明新下界 $F'(n)$ 使得 $F'(n) > F(n)$.

重复上述两步, 最终得到 $W_{A'}(n) = F'(n)$ 或者 $W_{A'}(n) = \Theta(F'(n))$.

主要内容

- 平凡下界
- 决策树模型
  - 检索问题的复杂度下界分析
  - 排序问题的复杂度下界分析
  - 冒泡排序、堆排序、排序算法的决策树与时间复杂度下界
- 选择问题的复杂度下界分析
  - 找最大和最小问题、找第二大问题、找中位数的问题

平凡下界（Ordinary Lower Bound）

- 算法的输入规模和输出规模是它的平凡下界
- 问题：写出所有的 n 阶置换
  - 求解的时间复杂度下界为 $\Omega(n!)$
- 问题：求 n 次实系数多项式在给定x的值
  - 求解的时间复杂度下界为 $\Omega(n)$
- 问题：求两个 n×n 矩阵的乘积
  - 求解的时间复杂度下界是 $\Omega(n^2)$
- 问题：货郎问题
  - 求解的时间复杂度下界是 $\Omega(n^2)$

检索问题的复杂度：顺序检索算法的复杂度分析

检索问题：给定按递增顺序排列的数组 L (项数 $n \geq 1$)和数 x，
如果 x 在 L 中, 输出 x 的下标; 否则输出0.

算法 顺序检索
输入: L, x
输出: j

1. $j \leftarrow 1$
2. while $j \leq n$ and $L(j) \neq x$ do $j \leftarrow j+1$
3. if $j > n$ then $j \leftarrow 0$

分析：假设 x 在L中每个位置和空隙的概率都是$1/(2n+1)$

$W(n)=n$

$A(n)=[(1+2+...+n)+n(n+1)]/(2n+1) \approx 3n/4$.

检索问题的复杂度：二分检索的最坏时间复杂度

定理1　$W(n)=\lfloor \log n \rfloor + 1 \qquad n \geq 1$

证　对n归纳

n=1时, 左= $W(1)$=1, 右 =$\lfloor \log 1 \rfloor$+1 = 1.

假设对一切k, $1 \leq k < n$, 命题为真, 则

$$
\begin{aligned}
W(n) &= 1 + W\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \\
&= 1 + \left\lfloor \log \left\lfloor \frac{n}{2} \right\rfloor \right\rfloor + 1 \\
&= \begin{cases} \lfloor \log n \rfloor + 1 & n\text{为偶数} \\ \lfloor \log(n-1) \rfloor + 1 & n\text{为奇数} \end{cases} \\
&= \lfloor \log n \rfloor + 1
\end{aligned}
$$

检索问题的复杂度：二分检索的平均时间复杂度 (1/2)

令 $n=2^k-1$, S_t 是算法做 t 次比较的输入个数, $1 \le t \le k$

则

$$S_1=1=2^0,\ S_2=2=2^1,\ S_3=2^2,\ S_4=2^3, \ldots,$$

$$S_t=2^{t-1},\ t<k$$

$$S_k=2^{k-1}+n+1$$

其中 2^{k-1} 为 x 在表中做 k 次比较的输入个数

$$A(n)=\frac{1}{2n+1}(1S_1+2S_2+\ldots+kS_k)$$

检索问题的复杂度：二分检索的平均时间复杂度 (2/2)

$$A(n) = \frac{1}{2n+1}(1S_1 + 2S_2 + \ldots + kS_k)$$

$$= \frac{1}{2n+1}[\sum_{t=1}^{k} t2^{t-1} + k(n+1)]$$

$$= \frac{1}{2n+1}[(k-1)2^k + 1 + k(n+1)]$$

$$\approx \frac{k-1}{2} + \frac{k}{2} = k - \frac{1}{2} = \lfloor \log n \rfloor + \frac{1}{2}$$

检索问题的决策树

设A是一个检索算法，对于给定输入规模 n，A的一棵决策树是一棵二叉树，其结点被标记为$1, 2, \ldots, n$, 且标记规则是:

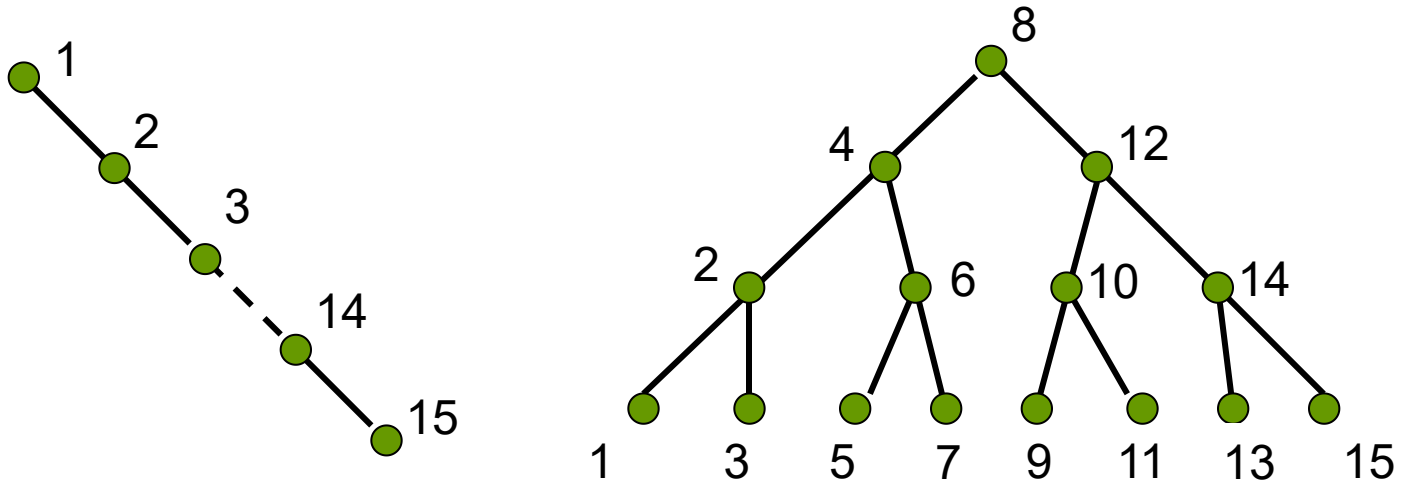
(1) 根据算法A，首先与 x 比较的 L 的项的下标标记为树根.

(2) 假设某结点被标记为i ，

- i 的左儿子是：当 $x<L(i)$时，算法A下一步与x比较的项的下标
- i 的右儿子是：当 $x>L(i)$时，算法A下一步与x比较的项的下标
- 若 $x<L(i)$ 时算法 A 停止，则 i 没有左儿子.
- 若 $x>L(i)$ 时算法 A 停止，则 i 没有右儿子.

检索问题的决策树：实例

顺序检索算法和二分检索算法的决策树，$n=15$



给定输入，算法 A 将从根开始，沿一条路径前进，直到某个结点为止。所执行的基本运算次数是这条路径的结点个数。最坏情况下的基本运算次数是树的深度+1。

决策树（Decision Tree）

- 决策树是一棵二叉树，对于给定问题（以比较运算作为基本运算）规定一个决策树的构造规则. 求解这个问题的不同算法所构造的决策树结构不一样.
- 给定一个算法的决策树. 对于任何输入实例，算法将从树根开始，沿一条路径向下，在每个结点做一次基本操作（比较）. 然后根据比较结果（<, =, >）走到某个子结点或者在该处停机. 对于给定实例的计算恰好对应了一条从树根到树叶或者某个内部结点的路径.
- 给定一个算法和输入的规模 n，对于不同的输入，算法将在对应决策树的某个结点（树叶或者内部结点）停机. 将该结点标记为一类输入. 问题的输入分类的数量对应于决策树的停机结点数（结点总数或者叶结点数）.

决策树与问题复杂度

- 决策树的特点：
  - 以比较作基本运算的算法模型
  - 一个问题确定了一类决策树，具有相同的构造规则，该决策树类决定了求解该问题的一个算法类
  - 结点数（或树叶数）等于输入分类的总数
  - 最坏情况下的时间复杂度对应于决策树的深度
  - 平均情况下的时间复杂度对应于决策树的平均路径长度
- 用决策树模型界定确定问题难度
  - 给定结点数（或树叶数）的决策树的深度至少是多少？
  - 给定结点数（或树叶数）的决策树的平均路径长度至少是多少？

决策树：二叉树的性质

命题1 在二叉树的 t 层至多 2^t 个结点（根为0层）

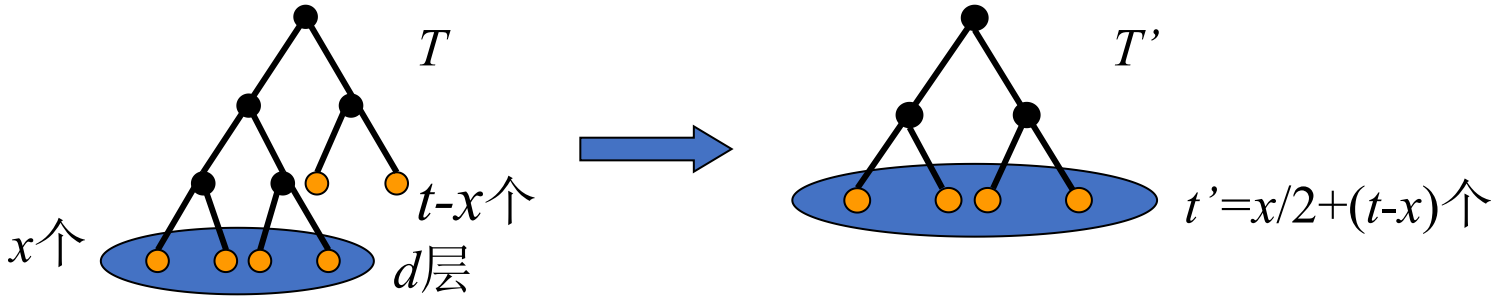
命题2 深度为 d 的二叉树至多 $2^{d+1}-1$ 个结点.

命题3 n个结点的二叉树的深度至少为 $\lfloor \log n \rfloor$.

命题4 设 t 为二叉树的树叶个数，d 为树深，如果树的每个内结点都有2个儿子，则 $t \le 2^d$.

命题4归纳法：$d=0$，命题为真。假设对小于d 的深度为真，设 T 深度为 d，树叶数 t。取走 T 的 d 层 树叶，得到T'。T' 的深度为$d-1$，树叶数 t'。

$$t = t' + x/2 \le 2^{d-1} + 2^{d-1} = 2^d$$ （“≤”由归纳假设和命题1得到）



检索问题的复杂度分析

- 定理2 对于任何一个搜索算法存在某个规模为 n 的输入使得该算法至少要做 $\lfloor \log n \rfloor + 1$ 次比较.
- 证 由命题3，n 个结点的决策树的深度 d 至少为 $\lfloor \log n \rfloor$, 故

$$W(n) = d+1 = \lfloor \log n \rfloor + 1.$$

- 结论：对于有序表搜索问题，在以比较作为基本运算的算法类中，二分法在最坏情况下是最优的.

检索问题的复杂度分析

- 检索问题
  - 给定按递增顺序排列的数组 L（项数 $n \geq 1$）和数 x，
  - 如果 x 在 L 中，输出 x 的下标；否则输出0。
- 思考
  - 根据平凡下界，检索问题输入数组 L 导致至少 Ω(n) 复杂度
  - 上述分析的 $\lfloor \log n \rfloor + 1$ 次比较是什么?
- 检索问题（$\lfloor \log n \rfloor + 1$ 下界版）
  - 给定O(1)时间查询递增顺序排列的数组 L（项数 $n \geq 1$）的查询器，以及数 x，
    - 查询器 $g(i,x) = \begin{cases} "L(i) < x" & \text{如果 } L(i) < x \\ "L(i) = x" & \text{如果 } L(i) = x \\ "L(i) > x" & \text{如果 } L(i) > x \end{cases}$ （将 L 排除出本问题的输入）
  - 如果 x 在 L 中，输出 x 的下标；否则输出0。

排序问题的时间复杂度分析

- 冒泡排序
  - 最坏和平均复杂性均为 $\Theta(n^2)$
- 快速排序
  - 最坏情况 $O(n^2)$；平均情况 O(nlogn)
- 二分归并排序
  - 最坏情况 O(nlogn)；平均情况 O(nlogn)
- 堆排序
  - 最坏情况 O(nlogn)；平均情况 O(nlogn)
- 排序问题的复杂度下界

排序问题的决策树

考虑以比较运算作为基本运算的排序算法类，

任取算法 A, 输入 $L=\{x_1, x_2, \ldots, x_n\}$, 如下定义决策树：

1．A第一次比较的元素为 x_i , x_j，那么树根标记为 i, j

2．假设结点 k 已经标记为 i, j，比较 x_i 和 x_j

(1) $x_i < x_j$

若算法结束，k 的左儿子标记为输出；

若下一步比较元素 x_p, x_q，那么 k 的左儿子标记为 p,q

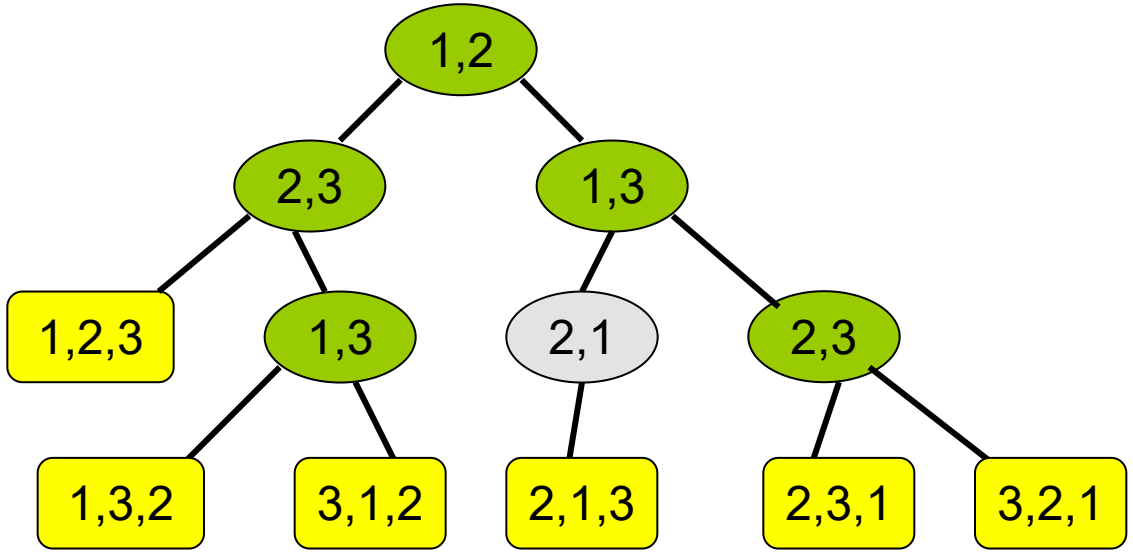
(2) $x_i > x_j$

若算法结束，k 的右儿子标记为输出；

若下一步比较元素 x_p, x_q，那么 k 的右儿子标记为 p,q

一棵冒泡排序的决策树

设输入为 x_1, x_2, x_3，冒泡排序的决策树如下



任意输入：对应了决策树树中从树根到树叶的一条路经，

算法最坏情况下的比较次数：树深

删除非二叉的内结点（灰色结点），得到二叉树叫做 B-树

B-树深度不超过决策树深度，B-树有$n!$片树叶.

排序问题的决策树：引理

引理1 设 t 为B-树中的树叶数，d 为树深，则 $t \le 2^d$.

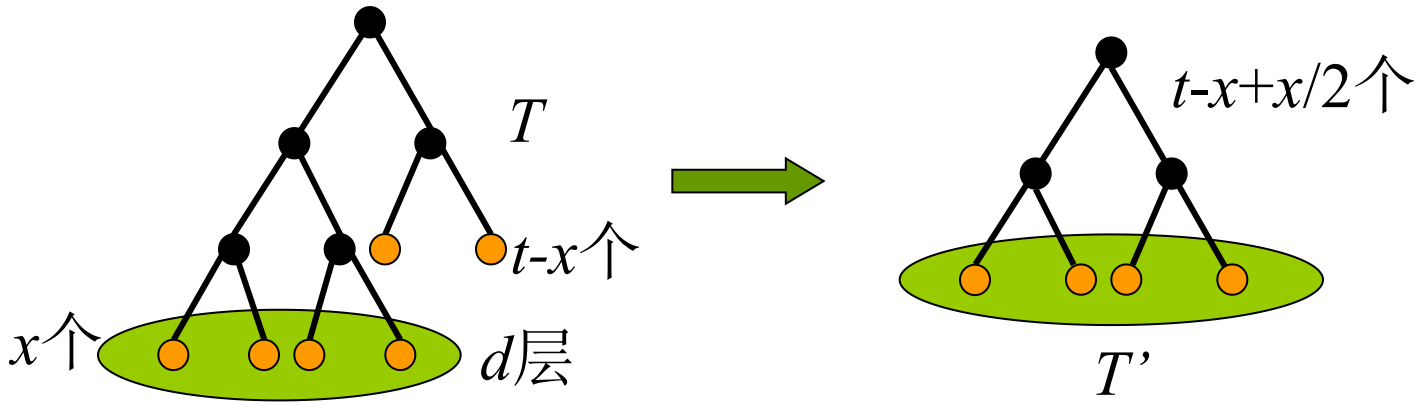
证明 归纳法.

d=0, 树只有1片树叶，深度为0，命题为真.

假设对一切小于d 的深度为真，设 T 是一棵深度为 d 的树，树叶数为 t. 取走 T 的 d 层的 x片树叶，得到树T'. 则T'的深度为$d-1$，树叶数 t'。那么

$$t' = (t-x) + x/2 = t - x/2, \quad x \le 2^d$$

$$t = t' + x/2 \le 2^{d-1} + 2^{d-1} = 2^d$$



排序问题的决策树：最坏情况复杂度的下界

引理2 对于给定的n，任何通过比较对 n 个元素排序的算法的决策树的深度至少为$\lceil \log n! \rceil$.

证明 判定树的树叶有$n!$个，由引理1得证.

定理4 任何通过比较对 n 个元素排序的算法在最坏情况下的时间复杂性是$\lceil \log n! \rceil$, 近似为 $n \log n - 1.5n$.

证明 最坏情况的比较次数为树深，由引理2树深至少为

$$\log n! = \sum_{j=1}^{n} \log j \geq \int_1^n \log x dx = \log e \int_1^n \ln x dx$$

$$= \log e(n \ln n - n + 1)$$

$$= n \log n - n \log e + \log e$$

$$\approx n \log n - 1.5n$$

结论：归并排序和堆排序算法在最坏情况阶达到最优.

排序问题的决策树：平均情况分析

epl(T)：假设所有的输入等概分布，令 epl(T) 表示 B-树中从根到树叶的所有路径长度之和，epl(T)/n! 的最小值对应平均情况复杂度的下界.

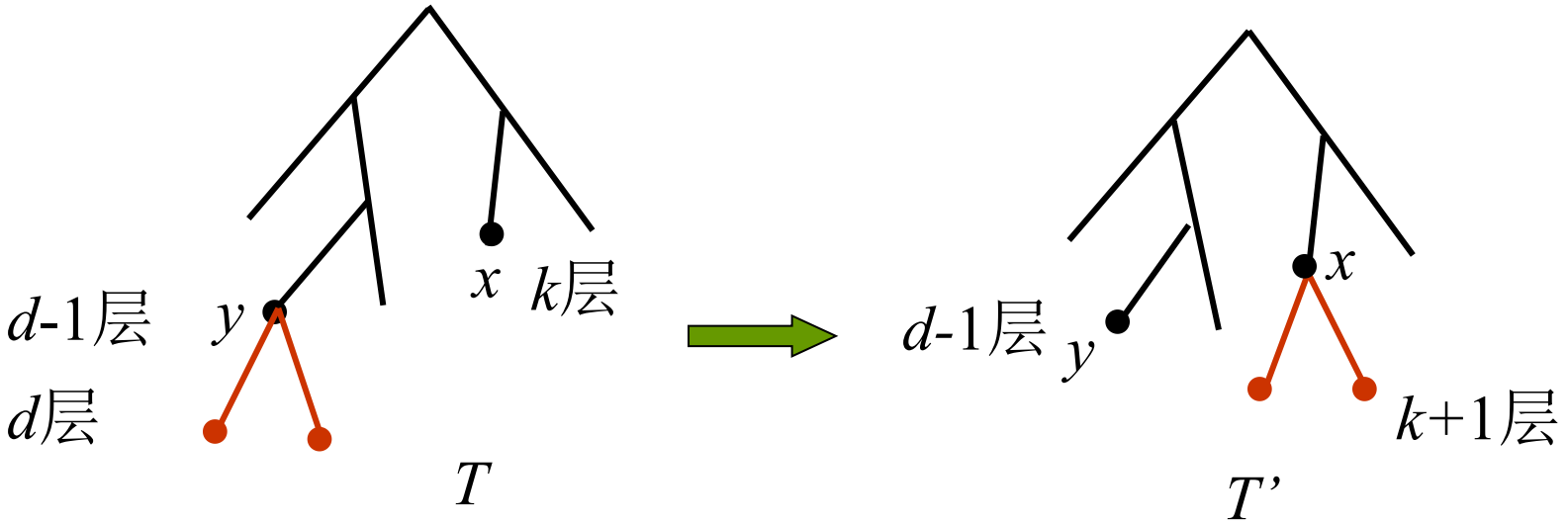
思路：分析具有最小 epl(T) 值的树的结构求得这个最小值.

引理3 在具有 t 片树叶的所有 B-树中,树叶分布在两个相邻层上的树的 epl 值最小

证明：反证法.

设树 T 的深度为 d，假设树叶 x 在第 k 层，$k<d-1$. 取 d-1层的某个结点 y，y 有两个儿子是第 d 层的树叶.将 y 的两个儿子作为 x 的儿子得到树 T'.

排序问题的决策树：具有最小 epl 值的树结构



$$\mathrm{epl}(T) - \mathrm{epl}(T') = (2d+k) - [(d-1)+2(k+1)]$$
$$= 2d+k - d+1-2k-2 = d-k-1 > 0 \qquad (d > k+1)$$

T'的树叶相距层数小于 T 的树叶相距的层数，

而 T' 的 epl 值小于 T 的 epl 值

排序问题的决策树：epl 值的下界

引理4　具有t片树叶且 epl 值最小的 B-树 T 满足

$$\mathrm{epl}(T) = t\lfloor \log t \rfloor + 2(t - 2^{\lfloor \log t \rfloor})$$

证明：由引理1 树 T 的深度 $d \geq \lceil \log t \rceil$,

由引理3 树 T 只有 d 和 d-1层有树叶.

Case1　$t = 2^k$. 必有$d = k$,

$$\mathrm{epl}(T) = t\,d = t\,k = t\,\lfloor \log t \rfloor$$

排序问题的决策树：epl 值的下界（续）

Case2 $t \neq 2^k$.

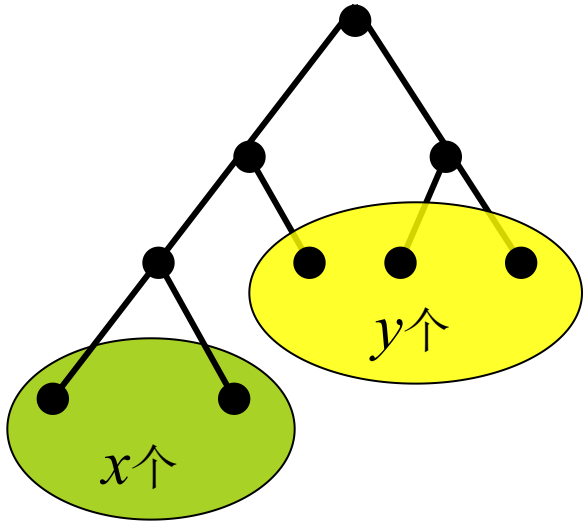
设 d 层和 $d-1$ 层树叶数分别为 x, y，

$$x + y = t$$

$$x/2 + y = 2^{d-1}$$

解得 $x = 2t - 2^d,\ y = 2^d - t.$

$$\begin{aligned}
\text{epl}(T) &= x\,d + y\,(d-1) \\
&= (2t - 2^d)d + (2^d - t)(d-1) \\
&= td - 2^d + t = t(d-1) + 2(t - 2^{d-1}) \\
&= t\lfloor \log t \rfloor + 2(t - 2^{\lfloor \log t \rfloor}) \quad (\lfloor \log t \rfloor = d-1)
\end{aligned}$$



排序问题的决策树：平均复杂度的下界

定理4　在输入等概分布下任何通过比较对n个项排序的算法平均比较次数至少为$\lfloor \log n! \rfloor$, 近似为 $n\log n - 1.5\, n$.

证明：算法类中任何算法的平均比较次数是该算法决策树T的 epl(T)/n!, 根据引理4

$$A(n) \geq \frac{1}{n!} epl\ (T)$$

$$= \frac{1}{n!}(n!\lfloor \log\ n! \rfloor + 2(n! - 2^{\lfloor \log\ n! \rfloor})$$

$$= \lfloor \log\ n! \rfloor + \varepsilon, \qquad 0 \leq \varepsilon < 1$$

$$\approx n\ \log\ n - 1.5n$$

$$0 \leq n! - 2^{\lfloor \log n! \rfloor} < n! - 2^{\log n! - 1} = n! - \frac{n!}{2} = \frac{n!}{2}$$

结论：堆排序在平均情况下阶达到最优.

几种排序算法的比较

算法	最坏情况	平均情况	占用空间	最优性
冒泡排序	$O(n^2)$	$O(n^2)$	原地	
快速排序	$O(n^2)$	$O(n\log n)$	$O(\log n)$	平均最优
归并排序	$O(n\log n)$	$O(n\log n)$	$O(n)$	最优
堆排序	$O(n\log n)$	$O(n\log n)$	原地	最优

选择问题

问题：从给定的集合 L 中选择第 i 小的元素

不妨设 L 为 n 个不等的实数

i=1，称为最小元素；

i=n，称为最大元素；

i=n-1，称为第二大元素；

位置处在中间的元素，称为中位元素

当n为奇数时，中位数只有1个，i=(n+1)/2；

当n为偶数时，中位数有2个，i=n/2, n/2+1. 也可以规定其中的一个

选最大

算法 FindMax

输入：n 个数的数组 L

输出：max, k

```
1.  max←L[1]; k←1
2.  for i←2 to n do
3.      if max <L[i]
4.      then max←L[i]
5.           k←i
6.  return max, k
```

算法最坏情况下的时间复杂度 $W(n)=n-1$

找最大问题的复杂度

下界：在 n 个数的数组中找最大的数, 以比较做基本运算的算法类中的任何算法在最坏情况下至少要做 n-1 次比较.

证 因为*MAX*是唯一的, 其它的 n-1 个数必须在比较后被淘汰。一次比较至多淘汰一个数，所以至少需要 n-1 次比较.

结论：FindMax 算法是最优算法.

选择算法的有关结果

	算法	最坏情况	空间
选最大	顺序比较	$n-1$	$O(1)$
选最大和最小	顺序比较	$2n-3$	$O(1)$
	算法FindMaxMin	$\lceil 3n/2 \rceil-2$	$O(1)$
选第二大	顺序比较	$2n-3$	$O(1)$
	锦标赛方法	$n+\lceil \log n \rceil-2$	$O(n)$
选中位数	排序后选择	$O(n\log n)$	$O(\log n)$
	算法Select	$O(n)$~$2.95n$	$O(\log n)$

选最大算法 FindMax是最优的算法

选最大和最小

通常算法：顺序比较

复杂性：$W(n)=2n-3$

算法 FindMaxMin

输入：n个数的数组L

输出：max，min

1．将n个元素两两一组分成$\lfloor n/2 \rfloor$组

2．每组比较，得到$\lfloor n/2 \rfloor$个较小和$\lfloor n/2 \rfloor$个较大

3．在$\lceil n/2 \rceil$个(n为奇数，是$\lfloor n/2 \rfloor+1$)较小中找最小min

4．在$\lceil n/2 \rceil$个(n为奇数，是$\lfloor n/2 \rfloor+1$)较大中找最大max

复杂性：行2 比较$\lfloor n/2 \rfloor$次，行3--4 比较至多$2\lceil n/2 \rceil-2$次，

$$W(n)=\lfloor n/2 \rfloor+2\lceil n/2 \rceil-2=n+\lceil n/2 \rceil-2=\lceil 3n/2 \rceil-2$$

选择算法的时间复杂度分析

下界证明方法：构造最坏输入 / 对手论证（adversary arguments）

- 任意给定一个算法 A，A 对于任意输入 x 都存在一个确定的操作序列 τ
- τ 中的操作分成两类：
  - 决定性的：能够对确定输出结果提供有效信息
  - 非决定性的：对确定结果没有帮助的冗余操作
- 根据算法 A 构造某个输入实例 x，使得 A 对 x 的操作序列 τ 包含尽量多的非决定性操作。
- 给出冗余操作+必要的操作的计数公式

选最大与最小算法

- 定理：任何通过比较找最大和最小的算法至少需要$\lceil 3n/2 \rceil$-2次比较。
- 证明思路：任给算法A，根据算法 A 的比较结果构造输入T ，使得 A 对 T 至少做$\lceil 3n/2 \rceil$-2 次比较。
- 证：不妨设n个数彼此不等，A为任意找最大和最小的算法。max是最大，A必须确定有n-1个数比max小，通过与max的比较被淘汰。min是最小，A也必须确定有n-1个数比min大，通过与min的比较而淘汰.总共需要2n-2个信息单位。

基本运算与信息单位

数的状态标记及其含义：

N：没有参加过比较　　W：赢

L：输　　WL：赢过且至少输1次

如果比较后数的状态改变，则提供信息单位，状态不变不提供信息单位，每增加 1 个W 提供 1个信息单位

每增加 1 个L 提供 1 个信息单位.

两个变量通过一次比较增加的信息单位个数不同：0,1,2

case1：N,N → W,L：增加2个信息单位

case2：W,N → W,L：增加1个信息单位

case3：W,L → W,L：增加0个信息单位

算法输出与信息单位

算法输出的条件：
n-2 个数带有 W 和 L 标记，最大数只带 W 标记，最小数只带 L 标记，总计 2n-2个信息单位

对于任意给定的算法，构造输入的原则是：
根据算法的比较次序，针对每一步参与比较的两个变量的状态，调整对参与比较的两个变量的赋值，使得每次比较后得到的信息单位数达到最小. 从而使得为得到 输出所需要的 2n-2个信息单位，该算法对所构造的输入至少要做 $\lceil 3n/2 \rceil$-2 次比较.

对输入变量的赋值原则

x 与 y 的状态	赋值策略	新状态	信息单位
N,N	$x>y$	W,L	2
W,N； WL,N	$x>y$	W,L； WL,L	1
L,N	$x<y$	L,W	1
W,W	$x>y$	W,WL	1
L,L	$x>y$	WL,L	1
W,L； WL,L； W,WL	$x>y$	不变	0
WL,WL	保持原值	不变	0

一个赋值的实例

每个步骤分解：① 算法发起比较请求；② 构造最坏输入；③ 返回比较结果

步骤	比较请求	比较结果	x_1		x_2		x_3		x_4		x_5		x_6	
			状态值		状态值		状态值		状态值		状态值		状态值	
			N	*	N	*	N	*	N	*	N	*	N	*
1	x_1, x_2	$x_1>x_2$	W	20	L	10	N	*	N	*	N	*	N	*
2	x_1, x_5	$x_1>x_5$	W	20	L	10	N	*	N	*	L	5	N	*
3	x_3, x_4	$x_3>x_4$	W	20	L	10	W	15	L	8	L	5	N	*
4	x_3, x_6	$x_3>x_6$	W	20	L	10	W	15	L	8	L	5	L	12
5	x_3, x_1	$x_3>x_1$	WL	20	L	10	W	25	L	8	L	5	L	12
6	x_2, x_4	$x_2>x_4$	WL	20	WL	10	W	25	L	8	L	5	L	12
7	x_5, x_6	$x_5>x_6$	WL	20	WL	10	W	25	L	8	WL	5	L	3
8	x_6, x_4	$x_6>x_4$	WL	20	WL	10	W	25	L	2	WL	5	WL	3

构造的输入为 (20, 10, 25, 2, 5, 3)

问题复杂度的下界

为得到$2n$-2个信息单位，对上述输入A至少做$\lceil 3n/2 \rceil$-2 次比较.
一次比较得到2个信息单位只有case1. A至多有$\lfloor n/2 \rfloor$个case1，至多得到 $2\lfloor n/2 \rfloor \le n$个信息单位. 其它case, 1次比较至多获得1个信息单位，至少还需要 n-2次比较.

当 n 为偶数，A做的比较次数至少为

$$\lfloor n/2 \rfloor + n\text{-}2 = 3n/2 - 2 = \lceil 3n/2 \rceil - 2$$

当 n 为奇数，A做的比较次数至少为

$$\lfloor n/2 \rfloor + n\text{-}2 + 1 = (n\text{-}1)/2 + 1 + n\text{-}2 = \lceil 3n/2 \rceil - 2$$

结论：FindMaxMin是最优算法

找第二大

通常算法：顺序比较

1．顺序比较找到最大max;

2．从剩下的$n-1$个数中找最大，就是第二大$second$

复杂度：$W(n)=n-1+n-2=2n-3$

锦标赛算法：

两两分组比较，大者进入下一轮

每个元素用数表记录每次比较时小于自己的元素

锦标赛算法

算法 FindSecond

输入：n个数的数组L

输出：$Second$

1．$k \leftarrow n$

2．将 k 个元素两两一组，分成$\lfloor k/2 \rfloor$组

3．每组的2个数比较，找到较大的数

4．将被淘汰的较小的数在淘汰它的数所指向的链表中做记录

5．if k 为奇数 then $k \leftarrow \lfloor k/2 \rfloor + 1$

6．else $k \leftarrow \lfloor k/2 \rfloor$

7．if $k>1$ then goto 2

8．$max \leftarrow$最大数

9．$Second \leftarrow max$ 的链表中的最大

时间复杂度分析

命题2.2 max在第一阶段的分组比较中总计进行了$\lceil \log n \rceil$次比较.

证 设本轮参与比较的有 t 个元素，经过分组淘汰后进入下一轮的元素数至多是$\lceil t/2 \rceil$. 假设 k 轮淘汰后只剩下一个元素 max，利用

$$\lceil \lceil t/2 \rceil /2 \rceil = \lceil t/2^2 \rceil$$

的结果并对 k 归纳，可得到 $\lceil n/2^k \rceil = 1$.

若 $n=2^d$，那么有 $k=d=\log n=\lceil \log n \rceil$

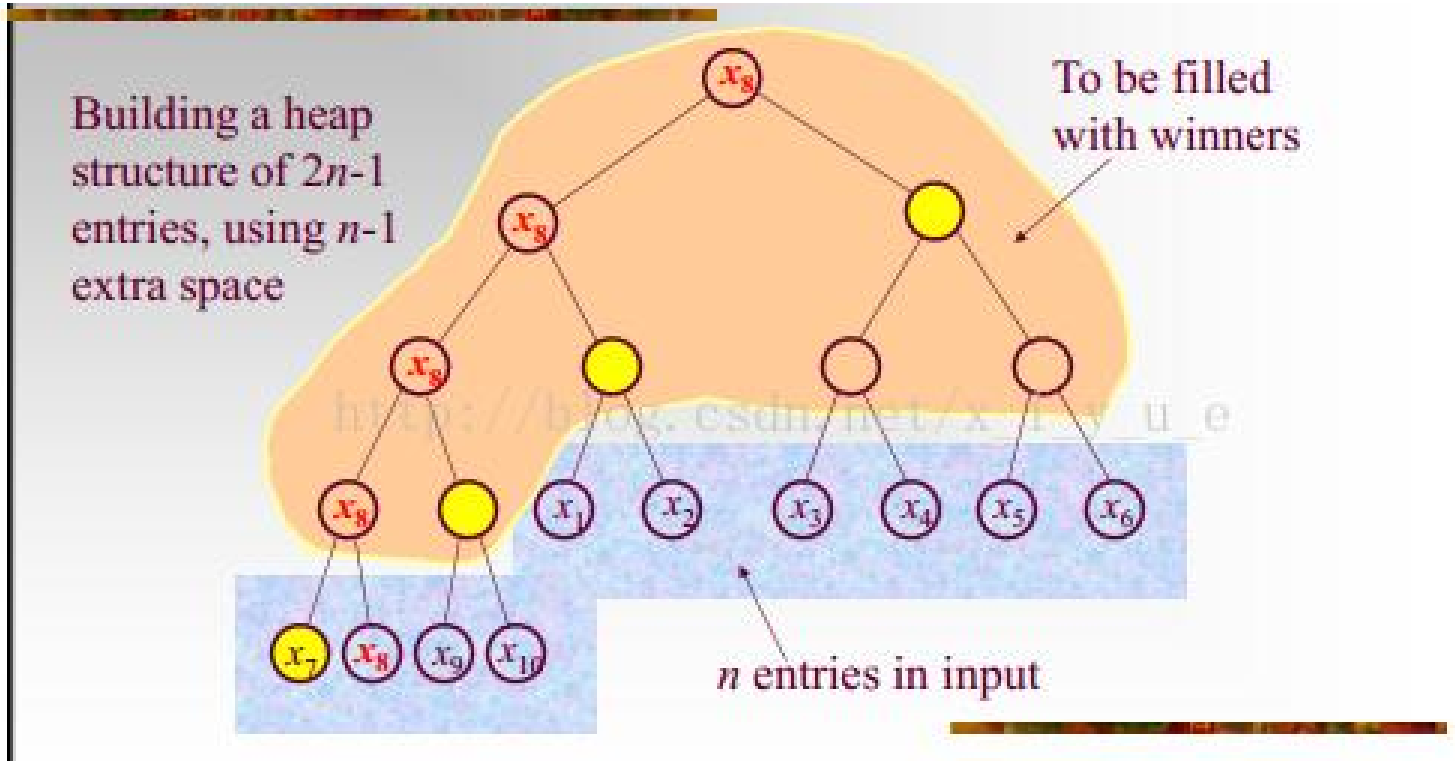
若 $2^d<n<2^{d+1}$，那么 $k=d+1=\lceil \log n \rceil$

算法时间复杂度是

$$W(n)=n-1+\lceil \log n \rceil -1= n+\lceil \log n \rceil -2.$$

找第二大问题

- 锦标赛算法



找第二大问题：问题复杂度分析

元素x的权：$w(x)$, 表示以x为根的子树中的结点数

初始，$w(x_i)=1$，$i=1, 2, \ldots, n$; 若 x>y，[w(x), w(y)]:=[w(x)+w(y), 0]

赋值原则： 在比较的时候进行赋值或者调整赋值。只对没有失败过的元素（权大于0的元素）进行赋值。权大者胜，原来胜的次数多的仍旧胜，构造较大的输入值.

1. $w(x), w(y)>0$:

   若 $w(x)>w(y)$, 构造 x 的值大于 y 的值；// 权大者胜

   若 $w(x)=w(y)$, 构造 x 的值大于 y 的值；// 权等，任意分配

2. $w(x)=w(y)=0$, 构造 x, y 值不变; // x与 y 比较对于确定第二大无意义

实例：构造最坏输入

每个步骤分解：① 算法发起比较请求；② 构造最坏输入；③ 返回比较结果

	比较请求	比较结果	$w(x_1)$	$w(x_2)$	$w(x_3)$	$w(x_4)$	$w(x_5)$	值
初始			1	1	1	1	1	*, *, *, *, *
第1步	x_1, x_2	$x_1>x_2$	2	0	1	1	1	20, 10, *, *, *
第2步	x_1, x_3	$x_1>x_3$	3	0	0	1	1	20, 10, 15, *, *
第3步	x_5, x_4	$x_5>x_4$	3	0	0	0	2	20, 10, 15, 30, 40
第4步	x_1, x_5	$x_1>x_5$	5	0	0	0	0	41, 10, 15, 30, 40

构造树

根据算法A 的比较次序, 在比最大的过程中如下构造树:

1. 初始是森林，含有n 个结点;
2. 如果 x, y 是子树的树根，则算法比较 x, y;
3. 若 x, y 以前没有参加过比较，任意赋值给 x, y，比如 $x>y$；那么将 y 作为 x 的儿子;
4. 若 x, y已经在前面的比较中赋过值，且 $x>y$，那么把 y 作为 x 的儿子，以 y 为根的子树作为 x 的子树;

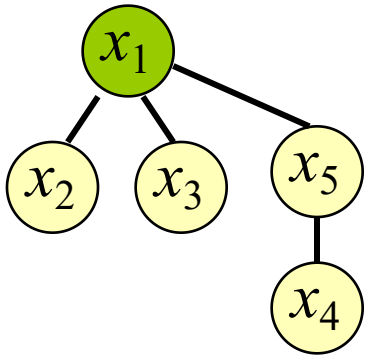
初始

$x_1>x_2$

$x_1>x_3$

$x_5>x_4$

$x_1>x_5$



实例

找第二大问题复杂度下界

针对这个输入，估计与max比较而淘汰的元素数
根的权为n，其它的结点权为0，根为max

w_k表示 max 在它第 k 次比较后形成以max为根子树的结点总数，则 $w_k \le 2w_{k-1}$，设 K 为max最终与权不为0的结点的比较次数，则

$$n = w_K \le 2^K w_0 \le 2^K \Rightarrow K \ge \log n \Rightarrow K \ge \lceil \log n \rceil$$

确定最大需要淘汰 $N-1$ 个元素，确定第二大需要淘汰 $K-1$ 个元素，至少用 $N+\lceil \log n \rceil - 2$ 次比较.

结论：锦标赛方法是找第二大的最优算法.

几种选择算法的总结

问题	算法	最坏情况	问题下界	最优性
找最大	**Findmax**	$n-1$	$n-1$	最优
找最大最小	**FindMaxMin**	$\lceil 3n/2 \rceil-2$	$\lceil 3n/2 \rceil-2$	最优
找第二大	锦标赛	$n+\lceil \log n \rceil-2$	$n+\lceil \log n \rceil-2$	最优
找中位数	**Select**	$O(n)$	$3n/2-3/2$	阶最优
找第k小	**Select**	$O(n)$	$n+\min\{k,n-k+1\}-2$	阶最优

本课小结

- 寻找最优算法的途径
  - 1. 设计算法；2. 分析问题下界；
  - 3-∞. 努力设计算法和分析下界，直至两者相遇
- 问题复杂度下界
  - 平凡下界
  - 直接计算最少运算次数（例：无序数组找最大)
  - 决策树模型（例：检索问题、排序问题）
  - 构造最坏输入/对手论证（例：选择问题）

（未完待续）