

正确设计	递归/分治/暴力/贪心/动态规划	常见问题
<p>① Divide: 大问题分解为 $\frac{n}{2}$ 与 $\frac{n}{2}$ (etc)</p> <p>② Conquer: 求解子问题</p> <p>③ 合: 子问题是递归 + 合并 合是递归 合是递归</p>	<p>① 递归: $T(n) = T(n/2) + f(n)$ ② $f(n) = O(n^2)$, 则 $T(n) = O(n^2)$ ③ $f(n) = O(n \log n)$, 则 $T(n) = O(n \log n)$ ④ $f(n) = O(1)$, 则 $T(n) = O(\log n)$ ⑤ $f(n) = O(n)$, 则 $T(n) = O(n)$ ⑥ $f(n) = O(n^2)$, 则 $T(n) = O(n^2)$ ⑦ $f(n) = O(n^3)$, 则 $T(n) = O(n^3)$ ⑧ $f(n) = O(n^4)$, 则 $T(n) = O(n^4)$ ⑨ $f(n) = O(n^5)$, 则 $T(n) = O(n^5)$ ⑩ $f(n) = O(n^6)$, 则 $T(n) = O(n^6)$ ⑪ $f(n) = O(n^7)$, 则 $T(n) = O(n^7)$ ⑫ $f(n) = O(n^8)$, 则 $T(n) = O(n^8)$ ⑬ $f(n) = O(n^9)$, 则 $T(n) = O(n^9)$ ⑭ $f(n) = O(n^{10})$, 则 $T(n) = O(n^{10})$ ⑮ $f(n) = O(n^{11})$, 则 $T(n) = O(n^{11})$ ⑯ $f(n) = O(n^{12})$, 则 $T(n) = O(n^{12})$ ⑰ $f(n) = O(n^{13})$, 则 $T(n) = O(n^{13})$ ⑱ $f(n) = O(n^{14})$, 则 $T(n) = O(n^{14})$ ⑲ $f(n) = O(n^{15})$, 则 $T(n) = O(n^{15})$ ⑳ $f(n) = O(n^{16})$, 则 $T(n) = O(n^{16})$ ㉑ $f(n) = O(n^{17})$, 则 $T(n) = O(n^{17})$ ㉒ $f(n) = O(n^{18})$, 则 $T(n) = O(n^{18})$ ㉓ $f(n) = O(n^{19})$, 则 $T(n) = O(n^{19})$ ㉔ $f(n) = O(n^{20})$, 则 $T(n) = O(n^{20})$ ㉕ $f(n) = O(n^{21})$, 则 $T(n) = O(n^{21})$ ㉖ $f(n) = O(n^{22})$, 则 $T(n) = O(n^{22})$ ㉗ $f(n) = O(n^{23})$, 则 $T(n) = O(n^{23})$ ㉘ $f(n) = O(n^{24})$, 则 $T(n) = O(n^{24})$ ㉙ $f(n) = O(n^{25})$, 则 $T(n) = O(n^{25})$ ㉚ $f(n) = O(n^{26})$, 则 $T(n) = O(n^{26})$ ㉛ $f(n) = O(n^{27})$, 则 $T(n) = O(n^{27})$ ㉜ $f(n) = O(n^{28})$, 则 $T(n) = O(n^{28})$ ㉝ $f(n) = O(n^{29})$, 则 $T(n) = O(n^{29})$ ㉞ $f(n) = O(n^{30})$, 则 $T(n) = O(n^{30})$ ㉟ $f(n) = O(n^{31})$, 则 $T(n) = O(n^{31})$ ㊱ $f(n) = O(n^{32})$, 则 $T(n) = O(n^{32})$ ㊲ $f(n) = O(n^{33})$, 则 $T(n) = O(n^{33})$ ㊳ $f(n) = O(n^{34})$, 则 $T(n) = O(n^{34})$ ㊴ $f(n) = O(n^{35})$, 则 $T(n) = O(n^{35})$ ㊵ $f(n) = O(n^{36})$, 则 $T(n) = O(n^{36})$ ㊶ $f(n) = O(n^{37})$, 则 $T(n) = O(n^{37})$ ㊷ $f(n) = O(n^{38})$, 则 $T(n) = O(n^{38})$ ㊸ $f(n) = O(n^{39})$, 则 $T(n) = O(n^{39})$ ㊹ $f(n) = O(n^{40})$, 则 $T(n) = O(n^{40})$ ㊺ $f(n) = O(n^{41})$, 则 $T(n) = O(n^{41})$ ㊻ $f(n) = O(n^{42})$, 则 $T(n) = O(n^{42})$ ㊼ $f(n) = O(n^{43})$, 则 $T(n) = O(n^{43})$ ㊽ $f(n) = O(n^{44})$, 则 $T(n) = O(n^{44})$ ㊾ $f(n) = O(n^{45})$, 则 $T(n) = O(n^{45})$ ㊿ $f(n) = O(n^{46})$, 则 $T(n) = O(n^{46})$ ㉀ $f(n) = O(n^{47})$, 则 $T(n) = O(n^{47})$ ㉁ $f(n) = O(n^{48})$, 则 $T(n) = O(n^{48})$ ㉂ $f(n) = O(n^{49})$, 则 $T(n) = O(n^{49})$ ㉃ $f(n) = O(n^{50})$, 则 $T(n) = O(n^{50})$ ㉄ $f(n) = O(n^{51})$, 则 $T(n) = O(n^{51})$ ㉅ $f(n) = O(n^{52})$, 则 $T(n) = O(n^{52})$ ㉆ $f(n) = O(n^{53})$, 则 $T(n) = O(n^{53})$ ㉇ $f(n) = O(n^{54})$, 则 $T(n) = O(n^{54})$ ㉈ $f(n) = O(n^{55})$, 则 $T(n) = O(n^{55})$ ㉉ $f(n) = O(n^{56})$, 则 $T(n) = O(n^{56})$ ㊰ $f(n) = O(n^{57})$, 则 $T(n) = O(n^{57})$ ㊱ $f(n) = O(n^{58})$, 则 $T(n) = O(n^{58})$ ㊲ $f(n) = O(n^{59})$, 则 $T(n) = O(n^{59})$ ㊳ $f(n) = O(n^{60})$, 则 $T(n) = O(n^{60})$ ㊴ $f(n) = O(n^{61})$, 则 $T(n) = O(n^{61})$ ㊵ $f(n) = O(n^{62})$, 则 $T(n) = O(n^{62})$ ㊶ $f(n) = O(n^{63})$, 则 $T(n) = O(n^{63})$ ㊷ $f(n) = O(n^{64})$, 则 $T(n) = O(n^{64})$ ㊸ $f(n) = O(n^{65})$, 则 $T(n) = O(n^{65})$ ㊹ $f(n) = O(n^{66})$, 则 $T(n) = O(n^{66})$ ㊺ $f(n) = O(n^{67})$, 则 $T(n) = O(n^{67})$ ㊻ $f(n) = O(n^{68})$, 则 $T(n) = O(n^{68})$ ㊼ $f(n) = O(n^{69})$, 则 $T(n) = O(n^{69})$ ㊽ $f(n) = O(n^{70})$, 则 $T(n) = O(n^{70})$ ㊾ $f(n) = O(n^{71})$, 则 $T(n) = O(n^{71})$ ㊿ $f(n) = O(n^{72})$, 则 $T(n) = O(n^{72})$ ㉀ $f(n) = O(n^{73})$, 则 $T(n) = O(n^{73})$ ㉁ $f(n) = O(n^{74})$, 则 $T(n) = O(n^{74})$ ㉂ $f(n) = O(n^{75})$, 则 $T(n) = O(n^{75})$ ㉃ $f(n) = O(n^{76})$, 则 $T(n) = O(n^{76})$ ㉄ $f(n) = O(n^{77})$, 则 $T(n) = O(n^{77})$ ㉅ $f(n) = O(n^{78})$, 则 $T(n) = O(n^{78})$ ㉆ $f(n) = O(n^{79})$, 则 $T(n) = O(n^{79})$ ㉇ $f(n) = O(n^{80})$, 则 $T(n) = O(n^{80})$ ㉈ $f(n) = O(n^{81})$, 则 $T(n) = O(n^{81})$ ㉉ $f(n) = O(n^{82})$, 则 $T(n) = O(n^{82})$ ㊰ $f(n) = O(n^{83})$, 则 $T(n) = O(n^{83})$ ㊱ $f(n) = O(n^{84})$, 则 $T(n) = O(n^{84})$ ㊲ $f(n) = O(n^{85})$, 则 $T(n) = O(n^{85})$ ㊳ $f(n) = O(n^{86})$, 则 $T(n) = O(n^{86})$ ㊴ $f(n) = O(n^{87})$, 则 $T(n) = O(n^{87})$ ㊵ $f(n) = O(n^{88})$, 则 $T(n) = O(n^{88})$ ㊶ $f(n) = O(n^{89})$, 则 $T(n) = O(n^{89})$ ㊷ $f(n) = O(n^{90})$, 则 $T(n) = O(n^{90})$ ㊸ $f(n) = O(n^{91})$, 则 $T(n) = O(n^{91})$ ㊹ $f(n) = O(n^{92})$, 则 $T(n) = O(n^{92})$ ㊺ $f(n) = O(n^{93})$, 则 $T(n) = O(n^{93})$ ㊻ $f(n) = O(n^{94})$, 则 $T(n) = O(n^{94})$ ㊼ $f(n) = O(n^{95})$, 则 $T(n) = O(n^{95})$ ㊽ $f(n) = O(n^{96})$, 则 $T(n) = O(n^{96})$ ㊿ $f(n) = O(n^{97})$, 则 $T(n) = O(n^{97})$ ㉀ $f(n) = O(n^{98})$, 则 $T(n) = O(n^{98})$ ㉁ $f(n) = O(n^{99})$, 则 $T(n) = O(n^{99})$ ㉂ $f(n) = O(n^{100})$, 则 $T(n) = O(n^{100})$</p>	<p>① 归并排序</p> <p>② Fibonacci 数: $f(n) = f(n-1) + f(n-2)$</p> <p>③ 矩阵乘法: $C = A \times B = (a_{ij} \times b_{jk})$</p> <p>④ Strassen 矩阵乘法: $T(n) = 7T(n/2) + O(n^2)$</p> <p>⑤ 快速排序: $T(n) = O(n \log n)$</p>

动态规划	子问题	常见问题
<p>① 与分治类似, 但子问题重叠</p> <p>② 最优子结构: 要求 $f(n)$, $f(n)$ 可由子问题得出, 如 $f(n) = f(n-1) + p$, 又 $f(n-1)$ 可由子问题得出, 则 $f(n)$ 可由子问题得出 (反证法)</p> <p>③ 比贪心优秀: 有重叠子问题, 如 Fibonacci: $F(n) = F(n-1) + F(n-2)$</p>	<p>① 子问题 $g(n)$</p> <p>② 子问题 $h(n)$</p> <p>$f(n) = g(n)h(n)$</p>	<p>① 背包问题: 每个物品有重量和价值</p> <p>② Rod Cutting: 每个长度 (子问题) 有最优解, $f(n) = \max(V(k) + f(n-k))$</p> <p>③ 矩阵链乘法: 安排乘法顺序, 使总次数最少: $f(i,j) = \min(f(i,k) + f(k,j) + calculate(k))$</p> <p>④ LCS (最长公共子序列): 最长公共子序列 (LCS) 问题: $f(a,b) = \max(f(a-1,b), f(a,b-1), f(a-1,b-1) + 1)$</p> <p>⑤ 最长公共子串: 有 a, b 两个字符串, a, b 不相同, 使公共子串最长: 找最长公共子串 (LCS) 问题</p>

贪心	排序	常见问题
<p>① SDP 类似, 也有最优子结构</p> <p>② 选择: 最优的是局部最优选择</p> <p>证明: 找到局部最优选择, 设一个局部最优选择, 用局部最优替代, 证明仍是局部最优</p> <p>另一种思路: 拟阵: $M = (S, I)$:</p> <p>S: 有限个元素, I: 独立集</p> <p>① 拟阵: $A \subseteq B, B \in I \Rightarrow A \in I$</p> <p>② 交换性: $A, B \in I, A < B , \exists x \in B - A, A \cup \{x\} \in I$</p> <p>③: 交换性: $A, B \in I, A < B , \exists x \in B - A, A \cup \{x\} \in I$</p>	<p>排序: $n \lg n$</p> <p>① 判断是否独立: $f(n)$</p> <p>② 拟阵: $O(n \lg n + n f(n))$</p> <p>③ 拟阵: $O(n \lg n + n f(n))$</p> <p>④ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑤ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑥ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑦ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑧ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑨ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑩ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑪ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑫ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑬ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑭ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑮ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑯ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑰ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑱ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑲ 拟阵: $O(n \lg n + n f(n))$</p> <p>⑳ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉑ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉒ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉓ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉔ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉕ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉖ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉗ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉘ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉙ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉚ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉛ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉜ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉝ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉞ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉟ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊰ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊱ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊲ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊳ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊴ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊵ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊶ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊷ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊸ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊹ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊺ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊻ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊼ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊽ 拟阵: $O(n \lg n + n f(n))$</p> <p>㊿ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉀ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉁ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉂ 拟阵: $O(n \lg n + n f(n))$</p> <p>㉃ 拟阵: $O(n$</p>	<p>①: 是 生成树: 每次取最小时, 不冲突不加入 (即不选)</p> <p>② activity selection: 一系列有起始、终止时间的任务, 安排使互不冲突的任务数最多, 每次找最早结束的</p> <p>③: task scheduling 有 deadline (截止时间), 求任务数最多的, 判断: 任务是否能在截止时间前完成 (即不 late) \Rightarrow 判断: 任务是否能在截止时间前完成</p> <p>④: Huffman (哈夫曼) 编码, 字符有出现次数, 构造最优的二叉树 (只在叶子节点编码)</p> <p>⑤: 哈夫曼树 (哈夫曼树)</p>

$\frac{1}{C^*} \sim \frac{1}{C}$, $\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq p(n)$
 (随机化化, $\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq p(n)$)

一般设计思路
 1. 作为NPL
 2. 用近似代替
 3. 求解
 4. 近似
 5. 近似: 444.44...

问题	描述	时间	
治版	每个顶点 (P_i, V_i, V_j) 一个 有环图，问题是否可解/是否有几个可能		随机 近
团问题	给定 G 和 k ，问题是否有 k 个顶点两两间均有边		
哈密顿	给定 G 和 k ，问题是否可用 k 个顶点覆盖所有边	团问题， $k=1$ 团， 有 k 个顶点在团中均存在，有 k 个顶点在团外均不存在，团问题也可解	
Hamilton	是否有 Hamilton 回路	用哈密顿回路	
TSP	给定 G 和 k ，是否有长为 k 的 Hamilton 回路 (完全图)	团 Hamilton $f(G) = G!$ 其中 $\begin{cases} G \text{ 中所有边} \\ \text{无边} = \text{边} = 1 \end{cases}$ $k=0$ 所有 Hamilton 回路均有长为 0 的 TSP	
集合划分	给定值 sum ，是否有 sum 的子集 $S' \subseteq S, S' \neq S$	用集合划分	

以度 $\frac{7}{8}m$ ($C^* \leq m$, $\frac{7}{8}m$)

① 贪心: 任取一条边 $e \in G$, 取 u, v
删去与 u, v 相关的所有边, 以此类推
为(近似) (贪心在 (u, v) 中各取一条边)

② 集合覆盖 (集合覆盖问题): 贪心: 每次取
和元素最多的
集合覆盖

③ 带权覆盖: { 查数内划, $\sum_{m,n} u(v) \times (v)$
 $S + \gamma(u) + \gamma(v) \geq 1$
 $\gamma(v) \in \{0, 1\}$

④ 近似 (LP)

⑤ 要求满足三角不等式, 求最短路径, 然后沿支持树 (查数内划) 是 TSP 近似 (查数内划) 是 TSP 近似 (查数内划)

2. 均摊	vector	二倍扩容
加上主键位或 取上限	$C_i = \begin{cases} 1 & 2^{n-1} + 1 \\ 1 & (2^{n-1} + 2^n) \end{cases}$ $n + \sum (2^{n-1} - 1)^k = 2^{n+1} + 1 \leq 3n$	$(1 - \frac{n}{2} + 2 - \frac{n}{4} + 3 - \frac{n}{8} \dots) \leq 2n$
每次存入 C, 保证 $\sum C_i \leq C_i$ (C 为最大容量)	每次存入 3 倍 - 2 倍	每次存入 2 倍
每次存入	$\phi_i = 2n \text{ num}_i - \text{size}$	$\phi = b_i$
		$d(n) = d(n-1) - d(n-2)$
		Sl-ti $\text{Ceset(b)} / \text{ite}$

求一个 trimming parameter σ ,
 ① 每次 merge 个 σ 入 L 中 $\sigma \leq 1$,
 ($L = L$ 再合并 σ 入 L 再合并 σ 入 L)
 ② ~~每次~~ 移除 σ 个
 ③ (与插入不同) (trim 移除 σ 个)
 使得 $L \leq L' \leq H$
 的 L'
 取 L' 为 L
 参数 σ
 上取 σ 个

4. 多线程: 把任务: T : work (线性时间) 并行
 T_p : 线性时间
 T_n : 多线程器上时间
 设计: (循环下标时) \Rightarrow 并行, $n \Rightarrow \lg n$
 (递归) \Rightarrow 并行: $T(n)$ 求 $G(n)$
 $\Rightarrow T(2, n)$

3. 递归: $\begin{cases} \text{递归树} \\ \text{时间复杂度} \end{cases}$ $\Rightarrow \Theta(\lg n)$

定义: $E P(X) = E(T(X))$ ~~...~~ $E(X) = \sum x_i P(X_i)$

问题: 矩阵乘法 (暴力: 前两层 for 循环 \Rightarrow 第 3 层 $O(n) + n = O(n^3)$) / 2 层 merge: 每层 $O(n \log n)$
 分治法 $(\frac{n}{2} \times \frac{n}{2} \times n) \Rightarrow T(\frac{n}{2}) + O(n^2)$ 分 2 层 merge
 分治法 merge: $\{ \text{merge: 分治} \Rightarrow \text{是 } O(n) \}$ 分 2 层 merge
 $\{ \text{merge: 分治} \Rightarrow \text{是 } O(n \log n) \}$ 递归: $O(n \log n)$



6/9/2019
排=存+排

排序:

算法	操作	复杂度
Naive		$O(n^2)$
merge	mergesort (前一半); mergesort (后一半); merge (前, 后)	$T(n) = 2T(\frac{n}{2}) + O(n) = O(n \lg n)$
快排	① 选 pivot, 左右分区 ② 递归前一半, 与 pivot 比 ③ 递归后一半, 与 pivot 比 ④ 递归 pivot 放中间	理想 $T(n) = 2T(\frac{n}{2}) + O(n) = O(n \lg n)$ (因为 $\alpha \leq (1-\alpha)n$) 最坏: $T(n) = O(n) + T(n-1) = O(n^2)$ 随机: $O(n \lg n)$
基数	排 b bit 数据, 取 $r = \lg b$, 每次取 r bit 再排 bit: 记 (高而, 低 r bit 为键值) 每个有 n 个 + 收集 (按键值) 数据 (同键值)	$T(n) = \frac{b}{r} (n + 2^r)$, 取 $r = \lg b$ 位, 为 $\frac{2b}{\lg b} n = O(n)$
插:	排 (0, 1) 随数: ① 每个数 x_i , 向有序表 $= i$, 放入插中 ② 插 i 做插入排序 ③ 收集	$T(n) = O(n) + E(\sum_{i=1}^n (n_i^2))$ $= O(n) + \sum_{i=1}^n E(n_i^2) = O(n) + n E(n_i^2)$ $= O(n)$
堆	① 堆 (二叉, Fibonacci) ② 初始堆顶, 堆顶数据 n 个	$T(n) = \begin{cases} O(n) + n O(\lg n) = n \lg n (=2) \\ O(1) + n O(\lg n) = n \lg n \text{ (Fibonacci)} \end{cases}$

平均时间
复杂度
Code turn
Code 复杂度

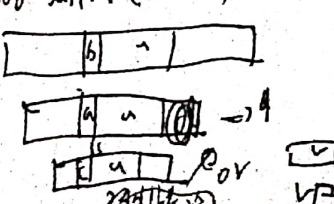
2 堆

算法	操作	复杂度
二叉堆	操作: 更新每个堆节点 自顶向下更新 (从下往上更新, 保证堆性质) 复杂度: $O(n)$	提取 min 并排序 操作: 将堆顶 \rightarrow 堆底 ① 堆顶与堆底比较 (与左子树比较, 若比左子树小, 则交换) 复杂度: $O(\lg n)$
Fibonacci 堆	操作: ① (合并) A 和 B 合并, 堆顶为 A 的堆顶 (≤ 1 个) ② 遍历 root list, 若某堆的 degree 在 A 中堆的 degree 中最大, 则合并 (degree + 1) 直到 A 中每个堆的 degree 都小于 A 中堆的 degree ③ 遍历 root list 与 min 复杂度: $c = \frac{1}{2} \log d(n)$, $\Delta \phi = d(n) - t$, $\phi = O(d(n)) = O(\lg n)$	插入一个 key 复杂度: $O(\lg n)$ 删除一个 key 复杂度: $O(\lg n)$ 插入一个 key 复杂度: $O(1)$ 合并 复杂度: $O(1)$



String Matching

6/11/2019

算法	预处理	匹配	复杂度
Naive	无	for (i=1; i<=n-m+1; i++) for (j=1; j<=m; j++) if (str[i+j-1] != pat[j-1]) break if (j==m) return i	$O(m(n-m+1))$
Robt Karp	将模式串打散: $P = (P[1], P[2], \dots, P[m])$ (R 为哈希)	for (i=1; i<=n-m+1; i++) num = P[i] * (k^m) - (k^m - 1) * P[i+m] / (k-1) numx = k, numy = P[i+m], numz = r	求哈希: $O(m)$ 求哈希: $O(m)$ 求哈希: $O(m)$ 求哈希: $O(m)$
KMP	前缀函数: $\pi(x) = \max\{k P[1..k] = P[x-k+1..x]\}$ (性质: 有回文, $x, \pi(x), \pi(\pi(x)) \dots = y$) 有 $P_y \supset P_x$ 求法: for q=2 to m while (k>0, P[k+1] != P[q]) k = pi(k) if (P[k+1] == P[q]) k++ pi(q) = k	for i=1 to n while q>0, P[q+1] != T[i] q = pi(q) if P[q+1] == T[i] q++ if q==m / 成功 q = pi(q)	预处理: $O(m)$ 匹配: $O(n)$
BM	求 $\sigma(x) = \min\{k, P[k]x\}$ 实现: $\sigma(q, a) = \sigma(P_q, a)$ ① 预处理 ② for q=0 to m for a in { if (q==m or P[q+1] != a) $\sigma(q, a) = \sigma(P_q, a)$ else $\sigma(q, a) = q+1$	for i=0 to n-1 q = $\sigma(q, T[i])$ if (q==m) 成功	预处理: $O(\Sigma \cdot m)$ 匹配: $O(n)$
Boyer-Moore	① Bad Character (BmBC) 假设: $\begin{matrix} \text{a} & \text{b} & \text{c} & \text{d} & \text{e} \\ \text{a} & \text{b} & \text{c} & \text{d} & \text{e} \end{matrix}$ 不匹配才移动到下一个 $\text{bmBC}(i) = \min\{L, S[i-m+1], P[m-i]\}$ (性质: $\text{bmBC}(i) \leq m$) 求法: 完全匹配从后往前找, $O(\Sigma \cdot m)$ ② Good Suffix (BmGS)  求法: $\text{O}_\text{suffix}(i) = \max\{P[i], \text{O}_\text{suffix}(i) \}$ 求法: $\text{O}_\text{suffix}(i) = \max\{P[i], \text{O}_\text{suffix}(i) \}$	① BmBC, BmGS ② while s<=n-m i=m while P[i] == T(s+i) i-- else i-- s = s + max(bmBC(s), bmBC(T(s+i)) - min)	预处理: $O(m \cdot \Sigma)$ 匹配: $O(n/m)$

