Iteration 迭代

Recursion 递归

CPU cost　　I/O cost

CPU ← RAM ← Hard Disk

ALU　　GB Level　　TB Level

Basic (atomic) operations of CPU

- Initialization
- Arithmetic (ALU)
- Comparison / Branching
- Memory Access

Big-O notation

$f(n) \le c_1 \cdot g(n)$ $(c_1 > 0)$ $n \ge c_2$ 成立

$f(n) = O(g(n))$

证: $\log_a n = O(\log_b n)$ $(a, b > 1)$

$\log_a n \le c_1 \log_b n$　　$c_1 \ge \frac{\log_a n}{\log_b n} = \frac{\lg b}{\lg a}$

$c_2 \ge \frac{\lg b}{\lg a} + 1$ 成立.

Big-Ω notation

$f(n) \ge c_1 \cdot g(n)$ $(c_1 > 0,\ n \ge c_2)$

$f(n) = \Omega(g(n))$

if $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$

$\Rightarrow f(n) = \Theta(g(n))$

- Binary Search Algorithm (logn)

```
left ← 1, right ← n
  repeat
    mid ← (left+right)/2
    if (t = A[mid]) then
        return TRUE
    else if (t < A[mid]) then
        right ← mid-1
    else
        left ← mid+1
  until left > right
  return FALSE
```

比较次数

worst time $g(n) = 2 + 9(1 + \log_2 n)$

$O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3)$
$< O(2^n) < O(n!) < O(n^n)$

	avr.	worse	best	space	stable
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	×
Insertion	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	✓
Bubble	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	✓
Merge	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$ depend	✓
Quick	$O(n\log_2 n)$	$O(n^2)$	$O(n)$	$O(1)$	×

Bubble Sort

```
for integer i ← 1 to n-1
   for integer j ← 2 to n
      if A[j-1] > A[j] then
        swap A[j-1] and A[j]
```

```
Merge-Sort (A, n)
  if n > 1
    p ← ⌊n/2⌋
    B[1...p] ← A[1...p]
    C[1...n-p] ← A[p+1...n]
    Merge-Sort(B, p)
    Merge-Sort (C, n-p)
  A[1...n] ← Merge (B, p, C, n-p)
Merge (L, nL, R, nR)
  n ← nL + nR
  let A[1...n] be new array
  i ← 1; j ← 1
  for k ← 1 to n
    if i ≤ nL and (j > nR or L[i] ≤ R[j])
      A[k] ← L[i]; i ← i+1
    else
      A[k] ← R[j]; j ← j+1
  return A
```

```
QuickSort (A[1...n]. lo=1, hi=n)
  p ← partition (A. lo. hi)
  QuickSort (A. lo, p-1)
  QuickSort (A, p+1, hi)
  Partition (A, lo, hi)
    p ← RANDOM(lo. hi); pivot ← A[p];
  L ← lo , R ← hi
   for integer i from lo to hi
     if (i != p)
      if (A[i] < pivot) A'[L++] ← A[i]
      else A'[R--] ← A[i]
   A'[L] ← pivot
    A[lo, hi] ← A'
   return L;
```

Master Theorem

$T(n) = aT(n/b) + f(n)$

$T(n) = \alpha T(\frac{n}{\beta}) + O(n^r)$ $n \ge 2$

$\alpha \ge 1,\ \beta > 1,\ r \ge 0$

① $\log_\beta \alpha < r$, $T(n) = O(n^r)$

② $\log_\beta \alpha = r$　$T(n) = O(n^r \log n)$

③ $\log_\beta \alpha > r$, $T(n) = O(n^{\log_\beta \alpha})$

链表优点

① 无限制生长

② 容易插入 / 删除

缺点

① 不提供随机访问

② 要进行 .next 操作

③ 占用额外内存　4+4 bytes

④ 不紧凑

遍历　traverse(A)

```
① if (A == NULL)
     return
  else print A.value
       traverse(A.next)

② node trav ← A
     while (trav != NULL)
        print trav.value
     trav ← trav.next
```

完全删除 free.

快慢指针

快1步, 慢2步, 每次迭代, 快多走一步 (环长M, 步长M)

无环: 末尾 N/2 次	有环: 相遇 M 次

O(N) 复杂度

交叉链表　空 O(1)　时 O(n)

PA指向A, PB指向B, 遍历

PA到尾　PA=headB →
PB到尾　PB=headA →　$\boxed{继续}$

一起到尾 ✓　分别不同尾 ✗

找倒数n个

快慢指针, 快比慢超前n节点.

Stack FILO　Queue FIFO

push pop peak isEmpty

clear size

5 * ((9+3)*(4*2)+7)　Infix

5 93+42**7+*　postfix

push(5) push(9) push(3) push(pop()+pop())

a+b*c 中　前　+a*bc

enQueue, deQueue. front 看头

Ring Queue

队满　Q.front == (Q.rear+1)%MAX

队空　Q.front == Q.rear

入队　Q.rear = (Q.rear+1)% MAX

应用　OS调度进程. 打印工作

图的bfs 或 树层序遍历

Selection Sort

```
for integer i ← 1 to n-1
   k ← i
   for integer j ← i+1 to n
      if A[k] > A[j] then
         k ← j
   Swap A[i] and A[k]
```

Insertion Sort　1两个[illegible]的

```
for integer i ← 1 to n
   for integer j ← i to 1 with j > 1
      if A[j-1] > A[j] then
        SwapA[j-1] and A[j]
   else break
```

Binary　$T(n) \le T(\frac{n}{2}) + c_2$

Merge　$T(n) = 2T(\frac{n}{2}) + O(n)$

数组优点

① 方便有效访问序列中的任何项

② 返回数组项中第i个元素

③ 每个项可在O(1)时间访问

④ 内存紧凑

缺点

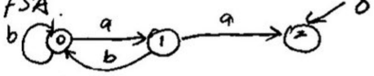
① 必须初始大小

② 调整大小麻烦

③ 很难插入/删除元素

子串 空串是任意串的子串
任意串为本身子串
真子串：非空且不为自身的子串
S_1 = SUS S_2 = 10
append SUS10 assign 10
insert(S2,0) 10SUS erase ""
replace('S',0) SUS swap find(0) S
子串数目 = $\frac{(1+s.len)(s.len)}{2}+1$

文字处理器 病毒扫描 搜索引擎
DNA 字词处理 结构
Brute Force O(mn)
Rabin-Karp T → n-m+1
O(mn) → O(n)
转换数字 O(1) $\boxed{mod}$ O(n-m)
P ≡ t[i] mod q ≠ p = t[i]
worse O(mn) 比这个

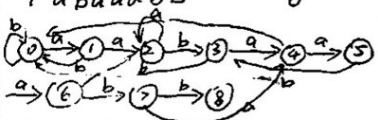
FSA


Q = {0,1,2} q_0 = 0 从状态q开始
Σ = {a,b} 输入 A = {2}
Search Pattern
abaaabb

	0	1	2	3	4	5	6	7
a	1	2	2	4	5	6	2	4
b	0	0	3	0	0	3	7	8

j	pattern[1...j]	x
0		0
1	a	1
2	ab	0
3	aba	1
4	abaa	2
5	abaaa	2
6	abaaab	3
7	abaaabb	0

最长 公共前后 ~~缀长度~~



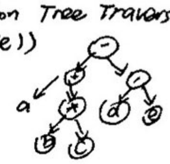
```
Transition(P.Σ)
m ← len(P)
x ← 0
Initialize δ(0,a) for each a∈Σ
for j ← 1 to m-1
  for each character a∈Σ
    if P[j+1] = a then
      δ(j,a) ← j+1
    else
      δ(j,a) ← δ(x,a)
  x ← δ(x, P[j+1])
return δ
```

```
FSA(T,P)                         O(Σ(m+n))
  n ← len(T), m ← len(p)
  δ ← Transition(P,Σ)
  q ← 0
  for i ← 1 to n
    q ← δ(q, T[i])
    if q = m
      patter occurs with shift i-m
```

String: HelloCS203
P[2,...4] = "ell" P[1...2] = He

```
NextArray(P)
m ← len(P)
let π[1...m] be a new array
π[1] = 0, k ← 0
for q = 2 to m
  while k > 0 and P[k+1] != P[q]
     k ← π[k]
  if P[k+1] = P[q]
     k ← k+1
  π[q] ← k
return π            O(m+n)
KMP(T,P)
 n ← len(T), m ← len(P)
 π ← NextArray(P)
 q ← 0
 for i = 1 to n
   while q > 0 and q[p+1] != T[i]
        q ← π[q]
   if (P[q+1] = T[i])
     q ← q+1
   If q == m
     print "pattern occur with shift i-m"
     q ← π[q]
```

插右
root 的
小左大右
找空

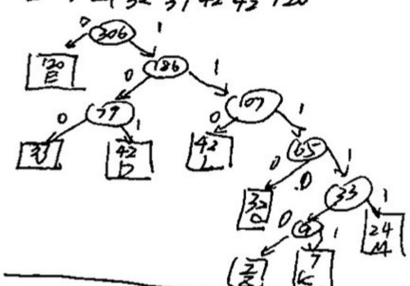
ancester 包括本身
proper ancestor
Internal nodes
Algebraic-Expression Tree Traversal
((a+(b*c))-(d/e))
print (befor visit
left tree
print) after visit
right tree.


前：GDAFEMHZ G root
中 ADEFGHMZ
left right
left ADEF right

中 ADEFGHMZ
后 AEFDHZMG 根

Huffman Encoding

Z	K	F	C	U	D	L	E
2	7	24	32	37	42	42	120

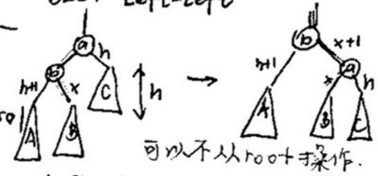


I n nodes n-1 edges
non-root nodes n-1
II 每个中间点至少2儿子 m为叶，中间m-1
Level H $\frac{m}{2}$ parent
Level H-1 $\frac{m}{2^2}$ ...
$\frac{m}{x}+\frac{m}{x^2}+\cdots+1$ x=2 Sm = m-1
III CBT n≥2 height O(log n)
Level 0 $2^0=1$ 1 $2^1=2$
$2^0+2^1+\cdots+2^{h-1}+x=n$
$(1-2^{h-1})/(1-2)=h-x$ $2^{h-1}<n$
h = O(log n)

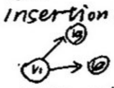
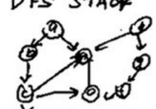
binary heap
O(n) space
插入 O(log n)
delete-min (O(log n)
和小的换. 1U8
找 rightmost node数 n=9
1001 去掉第1位 0往左. 1往右

Root-fix root左子、右子均为 binary heap
O(log n) 内完成 类似delete-min后半部分
Array → BH O(n)
A 完全二叉T. T高h. level → full
T.node $2^{h+1}-1$.
$T > \sum_{i=0}^{h} O(i \cdot 2^{h-i}) = O(2^{h+1}-1) = O(n)$

BST space O(n) 前续查找 O(h) 插删 O(h)
n nodes u为S不重复值 → key 左小右大
3, 10, 15, 20 ⇒ 23前向 20, 15 → 15, 3 → x
BST deletion
① 叶 → 直接删 ② 内，与successor换 有右子树
交换 → 比内大的最小值
~~③ 无右子树 与左子树换~~
worse O(n) h = n 长：O(log n) 操作
左右子树差 max = 1
BST n → (log n) min h → node
recursive ① h奇 ② h偶

BBST Left-Left


可以不从root操作.
Left-right


O(1)
insertion & deletion O(log n) Remedy

v1 → 邻子 → [邻] O(|V|+|E|)
矩阵 O(|V²|)
SSSP BFS 有向无权 → BFS TREE
白 → 黄 → 红 入Q. 出1步. 邻白 → 黄
O(|V|+|E|)
DFS STACK 黄在栈中. 先peak. 找白邻

a入 b入 c入 看栈顶

前向边 (a,b)(b,c)
后向 (c,b) 交叉 (a,e)
无后向边 → DAG
括号定理 [压栈, 出栈]
{ u > v. I(u) contains I(v) u < v.
否则 不相交

Topological Order
DFS 输出至L. L反向即为 O(|V|+|E|)
Dijkstra dist(v) < dist(u)+w(u,v) x
else dist(v) = dist(u)+w(u,v)
O(|V|+|E|)·log|V| 每次删最小

最小生成树 Prim
找最小边权 找相连最短 点用即不再

强连通分量 两两可达 最大组
反图 dfs 逆向结果 正图dfs
O(|V|+|E|)