implement 实施 trade-off. contradiction 矛盾
mechanically vertice symmetry
be derived from …源于 polynomial 多项式的 denote
trivial

山西省煤炭运销总公司
晋东南分公司
晋城市公司信笺

upper bound 上界

$O(g(n))$ = { $f(n)$: $0 \le f(n) \le cg(n)$ 存在常数 c, 在 $n \ge n_0$ 时一直成立 } 上界

$2n^2 = O(n^2) = O(n^3)$

$\Omega(g(n))$ = { $f(n)$: 有存在常数 c 和 n_0 使 $0 \le cg(n) \le f(n)$ 对任意 $n \ge n_0$ 时都成立 } 下界

$5n^2 = \Omega(n)$ $n = \Omega(2n)$

$\Theta(g(n))$ = { $f(n)$: 有存在常数 c_1, c_2, n_0 在 $n \ge n_0$ 时都有 $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ }.

$n \neq \Theta(n^2)$ 不存在 $c_1 n^2 \le n \le c_2 n^2$. $n \ge n_0$ $f(n)$

equality Θ

① $f(n) = \Theta(g(n))$ ⇔ $f = O(g(n))$ 且 $f = \Omega(g(n))$

② $f(n) = \Theta(g(n))$ 当且仅当 $g(n) = \Theta(f(n))$

$f(n) = o(g(n))$ 当且仅当 $g(n) = \omega(f(n))$

iteration trival

An algorithm is efficient if its running time is polynomical

(Insert sort 插入排序)

Merge sort 归并排序!!!

Master theorem ???

分治思想

divide-and conquer paradigm

```
mergesort(A, P, r)
  if P<r
    q = ⌊(P+r)/2⌋
    mergesort(A, P, q)
    mergesort(A, q+1, r)
    merge(A, P, q, r)
```

↓

two sorted subarrays 有序列

```
Merge(A, P, q, r)
n1 = q-P+1   n2 = r-q
Let L[1..n1+1] R[1..n2+1] 为新建数组
for i=1 → n1   L[i] = A[P+i-1]
for j=1 → n2   R[j] = A[q+j]
L[n1+1] = ∞   R[n2+1] = ∞
i=1  j=1
for k=P → r
  if L[i] ≤ R[j]
    A[k] = L[i]  i++
  else A[k] = R[j]  j++
```

3个指针

i, j, k 谁小谁就进 k一直向前

下!

$T(n)=\begin{cases} C & n=1 \\ 2T(\frac{n}{2})+cn & n>1 \end{cases}$

若 $n=2^p$ 取 $T(n)=f(p)$

$p=0$ 时 $T(n)=f(0)=C$

$f(p)=2f(p-1)+c2^p$ $=a_1+c\cdot 2^p$

$a_p=2a_{p-1}+c\cdot 2^p$

$a_p+x=2(a_{p-1}+x)$

$a_p+c\cdot 2^p=2(a_{p-1}+c\cdot 2^p)$

$a_p+c\cdot 2^p=2\cdot$

$a_0+c\cdot 2^p$

$=C+c\cdot 2^p$

这样写其实不对!

山西省煤炭运销总公司
晋东南分公司
晋城市公司信笺

$T(n)=O(n\log_2 n)$

$T(n)=\begin{cases} 0 & n=1 \\ T(\lceil\frac{n}{2}\rceil)+T(\lfloor\frac{n}{2}\rfloor)+n & n>1 \end{cases}$

$T(\lceil\frac{n}{2}\rceil)+T(\lfloor\frac{n}{2}\rfloor)$ 归并排序复杂度

assort

$T(n)=2T(\frac{n}{2})+n$

$\frac{T(n)}{2n}=\frac{2(\frac{n}{2})}{n}+1$

$\frac{T(n)}{n}=\frac{T(n/2)}{n/2}+1$

假设 $n=2^p$

$=\frac{T(n/4)}{n/4}+1+1$

$=\frac{T(n/n)}{n/n}+\underbrace{1+1+\cdots+1}_{\log_2 n}$

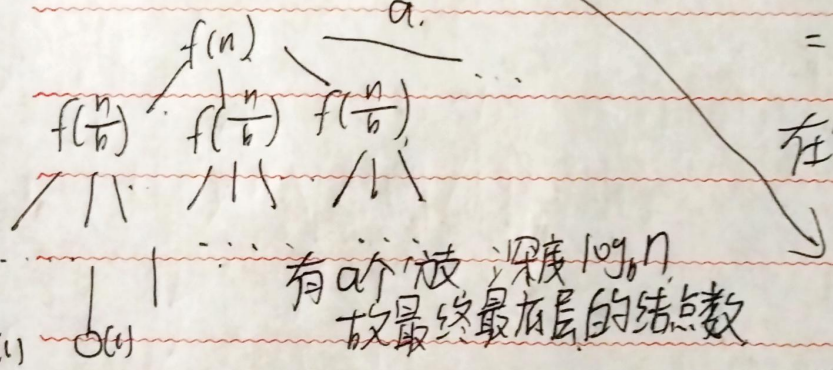
$=\log_2 n$

$T(n)=aT(\frac{n}{b})+f(n)$ $a\geq 0$ $b>0$

在各种情况下要解时间复杂度的因素

这个多项式的3种情况分析!

见《离散数学》吧!



有 a^p 个被 深度 $\log_b n$

故最终最底层的结点数

intuition

quicksort 还是分治思想

$A[p\cdots r]$ $\begin{cases} A[p\cdots q] \\ A[q+1\cdots r] \end{cases}$ 满足 $A[p\cdots q]$ 中所有元素都小于 $A[q+1\cdots r]$

recursive 递归的

与归并排序不同 没有 combine 之步骤

因为两个子序列已经满足条件了

(子)

```
quicksort(A,p,r)
if p<r
  q=Partition(A,p,r)
  quicksort(A,p,q)
  quicksort(A,q+1,r)
```

| ≤x | x | ≥x |

基元

71843.159

Partition()是关键！ Partition(A,P,r)

选择基元x 找个作为x 其实第1个就可

x=A[p] i=p-1 j=r+1

山西省煤炭运销总公司
晋东南分公司 晋城市公司信笺

while(true)

do j--;

直到 A[j]≤x 不合假设！

do i++;

直到 A[i]≥x

if(i<j) → 符合正常情况，交换A[i]和A[j]

否则，不正常情况，这时就找到门所求。

else return j

交换的是值

i,j这两个指针不变！

矩阵相乘的优化算法 $n^3 \to n^{2.8}\left(n^{\log 7}\right)$ Strassen算法 分治思想

优化想法 $\begin{bmatrix} A & B \\ C & D \end{bmatrix}\begin{bmatrix} E & F \\ G & H \end{bmatrix}$ 先分块 二分 二分

传统想法 (2x2)(2x2)矩阵相乘要乘 $2^3=8$ 次

乘7次 这就优化了！！！

网上具体看一下流程吧！

最大子序列问题，给定一个序列 在里面找个子序列(连续无限)，使之sum最大

目标 在A[low…high]中找到最大序列

分治 A[low…mid] A[mid+1…high]

3种可能的情况

①都在左 ②都在右 ③既在左又在右

③怎么搞？？？ ⇒

left-sum = -∞

sum=0

for i = mid -> low. 从mid到low

sum+=A[i]

if sum > left-sum

left-sum=sum

max-left=i

right-sum=-∞

for j=mid+1 -> high

sum+=A[j]

if sum > right-sum

right-sum=sum

max-right=j

return (max-left, max-right, left-sum+right-sum)

真要是③ 就都扫一遍，找最值！

时间复杂度 O(nlgn)

现在讲开始了Heap

一个堆 → 完全二叉树

每个父结点只有2个子结点

并且堆要(satisfy 满足)

一特性 A[parent(A[i])] ≥ A[i]

根大 → 大根堆　根小 → 小根堆

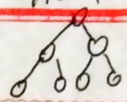
堆的深度 log(n+1)

山西省煤炭运销总公司
晋东南分公司
晋城市公司信笺

等比数列　节点数

wrong 此处的建堆指初始化

一个数列A　转化 → Heap



根节点为A[1]

某节点i的父节点为 $\frac{i}{2}$

子节点为2i和2i+1

堆的操作1(建堆后,满足堆特性)

给一个新加入的数据且要满足堆特性

假定这是数列A的第i元素　讨论大根堆

Heapify(A, i){

L = left(i)　r = right(i)

找左节点

要把i从1→n都扫一遍

若A[i] ≤ A[2i+1] 或A[2i] 交换——

自上而下调整

此处我理解的有误 应该是维持堆特性 Heapify操作.

从i开始维护堆的特点

堆的操作2 建堆

自底向上建堆

```
BuildHeap(A)
  heapsize = length(A)
  for (i = heapsize/2 -> 1)
    Heapify(A, i)
```

操作3 堆排序　每次把最大的pop 然后最后一个补上来.

```
Heapsort(A)
  BuildHeap(A)
  for (i = len(A) -> 2 i--)
    swap(A[1], A[i])
    heap_size -= 1
    Heapify(A, 1)
```

o(nlgn)

优先队列　用堆来实现.

现在就讲开动归了!!!

meta-technique

Not an algorithm

它是根据《算法导论》讲的

看那个就行

- Huffman
- 贪心算法
- 基础图论
- 最小生成树

71843.159