Algorithms & Formulae

* Simple k-way Merge: $T(N) = O(N\log k \cdot \log_k N) = O(N\log N)$

  $Pass = 1 + \lceil \log_k (N/M) \rceil$

  $Tape = 2k$

* Polyphase Merge: $T(N) = O(N\log N)$

  $Tape = k+1$

* Replacement Selection

  $E(\text{length of runs}) = 2 * \text{memory}$

* Minimum Merge Time Algorithm

  $\Rightarrow$ Huffman Tree, $T = O(\text{the weighted external path length})$

* Parallel Reduction

$T(N) = O(\log N)$, $W(N) = O(N)$

* Prefix-Sums

$T(N) = O(\log N)$, $W(N) = O(N)$

* Parallel Merging With Ranking

(i) Serial Ranking $T(N) = W(N) = O(N)$

(ii) Binary Search

$T(N) = O(\log N)$, $W(N) = O(N\log N)$

(iii) Partitioning

pivot个数 $= \frac{N}{\log N}$ (单侧)

$T(N) = O(\log N)$, $W(N) = O(N)$

* Maximum Finding —— Doubly-logarithmic Paradigm

子问题个数 $= \frac{N}{\log\log N}$

每个子问题内部用暴力法 $T(N) = O(1)$, $W(N) = O(N^2)$

合并子问题用 $\sqrt{N}$ 分割法 $T(N) = O(\log\log N)$, $W(N) = O(N\log\log N)$

共有 $T(N) = O(\log\log N)$, $W(N) = O(N)$

* Random Sampling : $T(N)=O(1)$, $W(N)=O(N)$

正确概率为：最大值落在随机抽取的 $N^{\frac{7}{8}}$ 个元素中的概率

无放回时 $P=\frac{1}{N^{\frac{1}{8}}}$ ，有放回时 $P=1-(1-\frac{1}{N})^{N^{\frac{7}{8}}}$

* Online Hiring Problem

(i) P(第i个应聘者恰被录用 (i>k) 且此人为N人中全局最优)

$=\frac{k}{i-1}\cdot\frac{1}{N}$ = P(第1到i-1人最大值落在前k) · P(全局最优位于第i)

(ii) 经过k人后，记遇到 $Q_i>Q_{(k)}$ 还需面M人，$M\sim Geo(\frac{1}{k+1})$

$E(M)=k+1$

* Modified Quicksort

原先 $\Theta(N^2)$

$E(T_{central\ splitor})=2$ ，最差时间复杂度 $T_{worst}=\Theta(N\log N)$

* State-flipping Algorithm

HNN: $T=O(\sum_{\forall e\in E}|w_e|)$ 寻找并终止于 stable configuration

对所有的u

Maximum Cut Problem: $\rho(n)=2$ $\quad\sum_{e=(u,v)} w_e s_u s_v \le 0$

* Big-Improvement-Flip Algorithm

$W(A_{t+1}, B_{t+1}) - W(A_t, B_t) \geqslant \frac{2\varepsilon}{|V|} W(A_t, B_t)$ 才翻转

其中 $W(A,B) \triangleq \sum_{u\in A, v\in B} w_{uv}$

$\rho(n) = 2+\varepsilon$，　$T = O(\frac{N}{\varepsilon}\log W)$

* Next Fit：$T(N) = O(N)$，$\rho(n) = 2$

$NF(L) \leqslant 2\,OPT(L) - 1$

* First Fit：$T(N) = O(N\log N)$；
  Best Fit：$\rho(n) = 1.7$

* Off-line First Fit Decreasing

$FFD(L) \leq \frac{11}{9} OPT(L) + \frac{2}{3}$

在 $OPT(L) = 2$ 时可推出 $FFD(L) \leq 3$，此时比值最大

$\rho(n) = 1.5$，当 $P \neq NP$ 时不存在近似比小于1.5的多项式时间算法

* Knapsack Problems

$P_{OPT} \leq P_{Frac} \leq P_{greedy} + p_{max}$

最大单利贪心与最大性价比贪心中更优者

$\rho(n) = 2$

$T(N) = O(NM)$ 或 $T(N) = O(N^2 p_{max})$

物品个数·背包容量　　物品个数·价值上限

* K-center Algorithm: $\rho(n)=2$

(i) 简单二分 2r-贪心算法: $T(N)=O(\log r_{max}\cdot N^k)$ 不是NP-hard

(ii) 最大距离 2r-贪心算法: $T(N)=$

除非 P=NP, 否则不存在 $\rho(n)<2$ 的近似算法

* NP-hard Problems

(i) Halting Problem $\notin$ NP

(ii) SAT Problem

(iii) Vertex Cover Problem $\Leftrightarrow$ Clique Problem

(iv) Hamiltonian Cycle Problem

(v) Bin Packing & 2 Bin Assessment

(vi) Max Leaf Spanning Tree

(vii) Minimum Degree Spanning Tree

(viii) Dominating Set Problem

找V的最小集使 $\forall v\in D$ 或有边连通到D

(ix) TSP

* Huffman Codes: $T(N) = O(N\log N)$，必为full

$Cost \triangleq \sum d_i f_i$ 最小，d_i是深度f_i是频率

* Activity Selection Problem:

不带权用贪心算法 $T(N) = O(N\log N)$

带权用动态规划 $T(N) = O(N^2)$

* Product Assembly: $T_{DP}(N) = O(N)$

* Floyd Algorithm: 密集图更优，所有顶点之间最短距离

$T(N) = O(N^3)$

* Optimal Binary Search Tree: 不必为full tree

$Cost \triangleq \sum (1+d_i) f_i$ 最小，$T(N) = O(N^3)$

$$C_{ij} = \begin{cases} 0 & i=j \\ \min\limits_{i\le l\le j} \{ \underline{w_{ij}} + C_{i,l-1} + C_{l+1,j} \}, & i<j \end{cases}$$

i到j所有词频之和

* Ordering Matrix Multiplications $T(N) = O(N^3)$

$$m_{ij} = \begin{cases} 0 & , j=i \\ \min\limits_{i\le l<j} \{ m_{il} + m_{l+1,j} + r_{i-1} r_l r_j \}, & j>i \end{cases}$$

* Closest Points Problem

$$T(N) = O(N\log N)$$

* Master Theorem

(I) $T(N) = aT(N/b) + f(N)$

(i) $f(N) > af(N/b)$, $\Theta[f(N)]$

(ii) $f(N) = af(N/b)$, $\Theta[\log_b N \cdot f(N)]$

(iii) $f(N) < af(N/b)$, $\Theta(N^{\log_b a})$

(II) $T(N) = aT(N/b) + \Theta(N^k \log^p N)$

(i) $a > b^k$, $O(N^{\log_b a})$

(ii) $a = b^k$, $O(N^k \log^{p+1} N)$

(iii) $a < b^k$, $O(N^k \log^p N)$

* Binomial Queue

找到最小值：$T(N)=O(\log N)$

合并：$T(N)=O(\log N)$

删除最小值：$T(N)=O(\log N)$

插入单结点：$T_{worst}(N)=O(\log N)$，

原先结点个数

$T_{armotized}(N)=O(1)$

构建：$T(N)=O(N)$ ✓

势函数：$\Phi_i \triangleq$ 第 i 次插入后队列中树的个数.

* Skew Heaps

合并、删除、插入：$T_{armotized}(N)=O(\log N)$

势函数：堆中 heavy node 的个数　　构建：$T(N)=O(N\log N)$

* Leftist Heap

$Npl(Root)=$ 最右路结点个数 -1

有 r 个结点在右路上，左式堆至少有 2^r-1 个结点

根结点 Npl 为 l，左式堆至少有 $2^{l+1}-1$ 个结点

删除、插入、合并：$T(N)=O(\log N)$

构建：$T(N)=O(N\log N)$

* Inverted File Index

$$\text{Precision} = \frac{\text{正类被检索}}{\text{所有返回}} \quad \text{Recall} = \frac{\text{正类被检索}}{\text{所有正类}}$$

* Red-Black Tree

不含x，包含NIL

$$\underbrace{\text{Sizeof}(x)}_{\text{包括}x\text{，不包括NIL}} \geq 2^{bh(x)} - 1$$

插入、删除：$T(N) = O(\log N)$

旋转次数：插入 ≤ 2，删除 ≤ 3 ✓

* AVL Tree

$$n_h = n_{h-1} + n_{h-2} + 1$$

插入、删除：$T(N) = O(\log N)$

旋转次数：插入 ≤ 2，删除 $= O(\log N)$

* Splay Tree

插入、查找、删除：$T_{armotized}(N) = O(\log N)$

势函数：$\Phi(T) = \sum_{i \in T} \log \underbrace{S(i)}_{\text{i子树上结点的个数}}$