

Some ADS Claims

created by wth

RB tree and B tree

1. A red-black tree with N internal nodes has height at most $2\log(N + 1)$

Inverted File Index

1. Precision $\frac{T_p}{T_p + F_p}$
2. Recall $\frac{T_p}{T_p + F_n}$

Leftist Heap and Skew Heap

1. $Npl(\text{NULL}) = -1$.
2. A leftist tree with r nodes on the right path must have at least $2^r - 1$ nodes.
3. It is an open problem to determine precisely the expected right path length of both leftist and skew heaps.
4. The only nodes whose heavy/light status can change are nodes that are initially on the right path.

Binomial Heap

1. A priority queue of any size can be uniquely represented by a collection of binomial trees.

2.	find min	merge	insert	delete min
	$O(1)$	$O(\log N)$	$O(1)$ (amortize)	$O(\log N)$

3. A binomial queue of N elements can be built by N successive insertions in $O(N)$ time.

Backtracking

1. Tic-tac-toe: α - β pruning: when both techniques are combined. In practice, it limits the searching to only $O(\sqrt{N})$ nodes, where N is the size of the full game tree.

Divide and Conquer

1. Master Theorem for a specific situation: $T(N) = aT(N/b) + O(N^k \log^p N)$

$$T(N) = \begin{cases} O(N^{\log_b a}), & \text{if } a > b^k \\ O(N^k \log^{p+1} N), & \text{if } a = b^k \\ O(N^k \log^p N), & \text{if } a < b^k \end{cases}$$

Greedy

1. Greedy algorithm works only if the local optimum is equal to the global optimum.
2. Greedy algorithm works only if the local optimum is equal to the global optimum.
3. Consider any nonempty subproblem S_k , and let a_m be an activity in S_k with the earliest finish time. Then a_m is included in some maximum-size subset of mutually compatible activities of S_k .

NP

1. Decidability — Could there exist, at least in principle, any definite method or process by which all mathematical questions could be decided?
2. Not all decidable problems are in NP. For example, consider the problem of determining whether a graph does not have a Hamiltonian cycle.
3. complexity class co-NP = the set of languages L such that $\bar{L} \in NP$. e.g. Hamiltonian cycle is not co-NP.

Approximation

1. $(1 + \epsilon)$ — **approximation**, if the algorithm is in polynomial time related to ϵ .
2. fullypolynomial-time approximation scheme (FPTAS): ϵ is not in the exponent of n.
3. Bin packing problem ratio

next fit	first fit	best fit	first fit decreasing
2M-1	1.7M	1.7M	11M/9+6/9

4. Knapsack Problem: maximum profit or profit density : The approximation ratio is 2.
5. center selection problem: 2-approximation.
6. Unless $P = NP$, there is no r-approximation for center-selection problem for any $r < 2$.

Local Search

1. The state-flipping algorithm terminates at a stable configuration after at most $W = \sum |w_e|$ iterations.
2. Any local maximum in the state-flipping algorithm to maximize F(sum of good edge's absolute value) is a stable configuration.
3. It is still an open question whether this is a polynomial time algorithm.
4. Let (A, B) be a local optimal partition and let (A*, B*) be a global optimal partition. Then $w(A, B) \geq \frac{1}{2}w(A^*, B^*)$.
5. Only choose a node which, when flipped, increases the cut value by at least $\frac{2\epsilon}{|V|}w(A, B)$, then the algorithm return a cut that $(2 + \epsilon)w(A, B) \geq w(A^*, B^*)$ terminates in $O(n/\epsilon \log W)$

Randomized Algorithm

1. online hiring Fthe probability we hire the best qualified candidate for a given k:

$$\frac{k}{N} \ln\left(\frac{k}{N}\right) \leq \Pr[S] \leq \frac{k}{N} \ln\left(\frac{k-1}{n-1}\right)$$

2. Central splitter := the pivot that divides the set so that each side contains at least $n/4$
3. Modified Quicksort := always select a central splitter before recursions
4. The expected number of iterations needed until we find a central splitter is at most 2.
5. Type j : the subproblem S is of type j if $N(\frac{3}{4})^{j+1} \leq |S| \leq N(\frac{3}{4})^j$
6. There are at most $(\frac{4}{3})^{j+1}$ subproblems of type j .

Parallel

1. maximum finding

partition	N^2	\sqrt{N}	$\log \log N$
$T(N)$	1	$\log \log N$	$\log \log N$
$W(N)$	N^2	$N \log \log N$	N

2. (Monte Carlo) complexity is $O(1/n^c)$ for some positive c Theorem:
The algorithm finds the maximum among n elements. With very high probability it runs in $O(1)$ time and $O(n)$ work. The probability of not finishing within this time and work is at most c .