|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | access | Search | | insertion | delation | | Access | search | | insertion | deletion |
| Hash table | - | O(1) | | O(1) | O(1) | |  | O(n) | | O(n) | O(n) |
| Bst | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) | | O(n) | O(n) | | O(n) | O(n) |
| B\_tree | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) |
| Red black | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) |
| Avl | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) | | O(log(n)) | O(log(n)) |
|  | | | Best | | | Average | | | Worst | | |
| Quicksort | | | 0(n log(n)) | | | O(n log(n)) | | | O(n^2) | | |
| Mergesort | | | O(n log(n)) | | | O(n log(n)) | | | O(n log(n)) | | |
| Heapsort | | | O(n log(n)) | | | O(n log(n)) | | | O(n log(n)) | | |
| prims | | | O(|E| log |V|) | | | O(|E| + |V| log |V|) | | | O(V^2) | | |
| Kruskals | | | O(E log V) | | | O(E log V) | | |  | | |
| Dijkstra | | | O(|E| + |V| log |V|) | | | O(|E| + |V| log |V|) | | | O(V^2) | | |
| Buildheap | | | O(n) | | |  | | | O(n log (n)) | | |
| Depth first/ breath first | | | O(1) | | |  | | | O(|E|) | | |
| T(N)=O(f(N)) if there are positive constants c and n0 such that T(N)<= cf(N) when N>= n0 | | | | | | The growth rate of T(N) is less than or equal to that of f(N). | | | | | |
| T(N) = Ω(f(N)) if there are positive constants c and n0 such that T(N)>=cf(N) when N>=n0 | | | | | | The growth rate of T(N) is greater than or equal to that of f(N). | | | | | |
| T(N)=Θ(f(N))) if and only if T(N)= O(f(N)) and T(N)=Ω(f(N)) | | | | | | The growth rate of T(N) equals the growth rate of f(N). | | | | | |
| T(N) = 0(f(N)) if T(N) – O(f(N)) and T(N) != Θ(f(N))  Dynamic Programming: Richard bellman. Optimally solve complex problems by breaking down into smaller problems that have overlaps. Knapsack i/o, Fibonacci series, optimal matrix multiplication | | | | | | The growth rate of T(N) is less than that of  f(N).  Binary heap: heapify O(n); find max O(1) ; extract max O(log(n)); increase key O(n log(n)); insert O(n log(n)); delete O(n log(n)); merge O(m+n) | | | | | |

Knapsack: V[k,w]= V[k-1,w]

Max{V[k-1,w], V[k-1, w- wk]+bk