Back Track

演算法	Time Complexity
DFS、BFS	Adjacency Matrix：$O(n^2)$ Adjacency List：O(|E|+|V|)

Minimal Spanning Tree(DS 多談 Adj Matrix、演算法多談 Adj List)

演算法	Time Complexity	策略	
Kruskal's	Adjacency Matrix：$O(V^2)$ Adjacency List(Binary Heap 製作)：**O(|E|log|E|)** Adjacency List(Fibonacci Heap 製作)：O(|E|log|V|)	Greedy	Adjacency Matrix Min-Heap Disjoint Sets Tree
Prim's	Adjacency Matrix：**$O(V^2)$** Adjacency List(Binary Heap 製作)：O(|E|log|V|) Adjacency List(Fibonacci Heap 製作)：O(|V|log|V|+|E|)	Greedy	Adjacency Matrix Min-Heap Array

Shortest Path

演算法	Time Complexity	策略	負邊	負環
Dijkstra's	Adjacency Matrix：$O(V^2)$ Adjacency List：$O(V^2)$ Binary Heap：O(|E|+|V|log|V|) Fibonacci Heap：O(|E|+|V|log|V|)	Greedy	X	X
Bellman-Ford	Adjacency Matrix：**$O(V^3)$** Adjacency List：O(V×E)	Dynamic Programming	O	X
Floyd-Warshall	Adjacency Matrix：$O(V^3)$	Dynamic Programming	O	X

Comparison of Various Structures

	Array	Linked-List	AVL Tree
Insert	O(1)	O(n)	O(log n)
Delete	O(1)	O(n)	O(log n)
Search	O(log n) //Binary Search	O(n)	O(log n)
Search k^{th} item	O(1)	O(k)	O(log n)
Delete k^{th} item	O(n-k) //後面 n-k 個都要往前移一格	O(k)	O(log n)
Output in order	O(n)	O(n)	O(n)

[Thormas C 演算法版]

Operation	Binary Heap*(Worst, DS 特愛考)*	Binomial Heap(Worst)	Fibonacci Heap(攤)
Create-Heap	O(1)	O(1)	O(1)
Insert	O(log n)	O(log n) //[1]	O(1) //Lazy Merge
Delete	O(log n) //[6]	O(log n)	O(log n)
Find-Min	O(1)	O(log n) //[2]	O(1) //[3]
Extract-Min	O(log n)	O(log n)	O(log n)
Union	O(n)	O(log n)	O(1) //[4]
Decrease key	O(log n)	O(log n) //[5]	O(1)

[1]Binomial Heap 的 Worst 下，Insert O(log n)，但分攤下為 O(1)
[2]因為 Binomial Heap 最差下需比較 log n 棵樹的 Root 找 min
[3]Fibonacci Heap 有設定指標指向 Min
[4]因為 Fibonacci Heap 採用 Lazy Merge
[5]例：看高度，因此 O(log n)
[6]用最差情況 Delete Min 來考量：O(log n)

Sort

		Time Complexity			Space Complexity	Stable/Unstable
		Best	Average	Worst		
初等	Insertion	**O(n)**	$O(n^2)$	$O(n^2)$	O(1)	**Stable**
	Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)	**Unstable**
	Bubble	**O(n)**	$O(n^2)$	$O(n^2)$	O(1)	**Stable**
	Shell	$\mathbf{O(n^{3/2})}$	$O(n^2)$	$O(n^2)$	O(1)	**Unstable**
高等	Quick	O(nlog n)	O(nlog n)	$\mathbf{O(n^2)}$	**O(log n)~O(n)**	**Unstable**
	Merge	O(nlog n)	O(nlog n)	O(nlog n)	**O(n)**	**Stable**
	Heap	O(nlog n)	O(nlog n)	O(nlog n)	**O(1)**	**Unstable**
線性	Radix(LSD)	**O(d*(n+r))** ⟹ O(n)			**O(r*n)**	**Stable**
	Bucket(MSD)	**O(n+r)** ⟹ O(n) *//比 Radix 快*			**O(r*n)**	**Unstable**
	Counting	**O(n+k)** ⟹ O(n)			**O(n+k)**	**Stable**

		Best	Average	Worst
初等	Insertion	T(n)=1	T(n)=T(n-1)+Θ(n)	T(n)=T(n-1)+(n-1)
	Selection	T(n)=T(n-1)+1	T(n)=T(n-1)+Θ(n)	T(n)=T(n-1)+(n-1)
	Bubble	T(n)=1	T(n)=T(n-1)+Θ(n)	T(n)=T(n-1)+(n-1)
高等	Quick	T(n)=2T(n/2)+Θ(n)	**T(n)=1/nΣ[T(j-1)+T(n-j)+Θ(n)]**	T(n)=T(n-1)+Θ(n)
	Merge	T(n)=2T(n/2)+Θ(n)	T(n)=2T(n/2)+Θ(n)	T(n)=2T(n/2)+Θ(n)
	Heap	T(n)=Θ(n)+Θ(nlogn)	T(n)=Θ(n)+Θ(nlogn)	T(n)=Θ(n)+Θ(nlogn)

Greedy

Huffman	O(n log n)
Fraction Knapsack Problem	O(n log n)
Kruskal's	Adjacency List：**O(|E|log|E|)**
Prim's	Adjacency Matrix：$\mathbf{O(V^2)}$
Sollin's	$\mathbf{O(V^2)}$
Dijkstra's	Adjacency Matrix：$\mathbf{O(V^2)}$
Convex Hull	**O(n log n)**

Dynamic Programming

LCS / LIS	O(mn) / $O(n^2)$
0/1 Knapsack Problem	O(nW)
Chain Matrix	$O(n^3)$
Bellman-Ford	Adjacency Matrix：$\mathbf{O(V^3)}$
Floyd-Warshell	Adjacency Matrix：$O(V^3)$
OBST	$O(n^3)$
TSP(Traveling Salesman Problem)	$O(n^2 2^n)$

Divide-and-Conquer

Tower of Hanoi	$O(2^n)$
Binary Search	O(log n)
Quick Sort	O(n log n)
Merge Sort	$O(n^2)$
Bucket Sort(MSD)	O(n+r)
Closet Pair	**O(n log n)**
Strassen's Matrix	$\mathbf{O(n^{\log 7})}$

Multiplication	

演算法解題技巧

Branch and Bound 解 KP	O(nW)　　//Worst Case 亦為 NPC
Prune and Search 選第 k 小之數	$T(n)=T(n/5)+T(3n/4)+\Theta(n)=\Theta(n)$
陣列合併	$\Theta(k\log k)+\Theta(k)=\Theta(k\log k)$
列出子集	$\Theta(2^n)$
找 1-1 函數	$\Theta(n)$
名人問題	$\Theta(n^2)$
平面上極大點	$\Theta(n\log n)$
點之 Rank	$T(n)=2T(n/2)+\Theta(n)=\Theta(n\log n)$
最大連續整數和	$\Theta(n)$

	Unsorted Single Linked List	Sorted Single Linked List	Unsorted Double Linked List	Sorted Double Linked List
Search(L, k)	O(n)	O(1)	O(n)	**O(logn)**
Insert(L, p)	O(1)	O(n)	O(1)	O(n)
Delete(L, p)	O(n)	O(n)	O(1)	**O(1)**
Successor(L, p)	O(1)	O(1)	O(1)	O(1)
Predecessor(L, p)	O(n)	O(n)	O(1)	O(1)
Minimum(L)	O(n)	O(1)	O(n)	O(1)
Maximum(L)	O(n)	O(1)	O(n)	O(1)