時間複雜度整理

Minimum Spanning Tree			
Algorithm	**Time Complexity**	**Type**	**Structure**
Kruskal's Alog	Adjacency matrix: $O(V^2)$ Adjacency list: $O(\|E\| \log\|E\|)$	Greedy	Adjacency matrix, Min. Heap & Disjoint Sets Tree Representation
Prim's Algo	Adjacency matrix: $O(V^2)$	Greedy	Adjacency matrix & Array

Shortest Path				
Algorithm	**Time Complexity**	**Type**	**負邊**	**負 cycle**
Dijkstra's Algo	Adjacency matrix, 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: $O(VE)$	Dynamic Programming	O	X
Floyd Warshall	$O(V^3)$	Dynamic Programming	O	X

Back track		
Algorithm	**Time Complexity (Adjacency List)**	**Time Complexity (Adjacency Matrix)**
DFS	$O(\|E\| + \|V\|)$	$O(n^2)$
BFS	$O(\|E\| + \|V\|)$	$O(n^2)$

Comparison of Various Structures			
Operation	**Array**	**Link list**	**AVL tree**
Search for X	$O(\log n)$	$O(n)$	$O(\log n)$
Insert	$O(n)$	$O(1)$	$O(\log n)$
Delete X	$O(n)$	$O(1)$	$O(\log n)$
Search k'th item	$O(1)$	$O(k)$	$O(\log n)$
Delete k'th item	$O(n-k)$	$O(k)$	$O(\log n)$
Output in order	$O(n)$	$O(n)$	$O(n)$

Comparison of Heaps				
Operation	**Link list**	**Binary Heap**	**Binomial Heap**	**Fibonacci Heap**
Create Heap		O(n)		
Insert	Θ(1)	Θ(log n)	O(log n)	Θ(1)
Delete	Θ(n)	Θ(log n)	Θ(log n)	O(log n)
Find-Min	Θ(n)	Θ(1)	O(log n)	Θ(1)
Delete-Min	Θ(n)	Θ(log n)	Θ(log n)	O(log n)
Union	Θ(1)	Θ(n)	O(log n)	Θ(1)
Decrease Key	Θ(1)	Θ(log n)	Θ(log n)	Θ(1)

Sort of Recursive formula (97 台大)			
Sort Algorithm	**Best Case**	**Worst Case**	**Average Case**
Insert	T(n) = T(n-1) + 1	T(n) = T(n-1) + (n-1)	T(n) = T(n-1) + Θ(n)
Bubble	T(n) = T(n-1) + 1	T(n) = T(n-1) + (n-1)	T(n) = T(n-1) + Θ(n)
Selection	T(n) = T(n-1) + 1	T(n) = T(n-1) + (n-1)	T(n) = T(n-1) + Θ(n)
Quick	$T(n) = T(\frac{1}{2}) + T(\frac{1}{2}) + \Theta(n)$	T(n) = T(n-1) + Θ(n)	$T(n) = \frac{1}{n}\sum_{j=1}^{n}(T(j-1)+T(n-j)) + \Theta(n)$ $\fallingdotseq T(n) = \frac{2}{n}\sum_{j=0}^{n-1}T(j) + \theta(n)$
Merge	$T(n) = T(\frac{1}{2}) + T(\frac{1}{2}) + \Theta(n)$	$T(n) = T(\frac{1}{2}) + T(\frac{1}{2}) + \Theta(n)$	$T(n) = T(\frac{1}{2}) + T(\frac{1}{2}) + \Theta(n)$
Heap	T(n) = Θ(n) + Θ(nlogn)	T(n) = Θ(n) + Θ(nlogn)	T(n) = Θ(n) + Θ(nlogn)

Sort of Time Complexity						
Sort Algorithm	**Best Case**	**Worst Case**	**Average Case**	**Storage**	**Stable**	**Compare based**
Insert	O(n)	O(n^2)	O(n^2)	O(1)	O	O
Bubble	O(n)	O(n^2)	O(n^2)	O(1)	O	O
Selection	O(n^2)	O(n^2)	O(n^2)	O(1)	X	O
Quick	O(nlogn)	O(n^2)	O(nlogn)	O(nlogn)	X	O
Merge	O(nlogn)	O(nlogn)	O(nlogn)	O(n)	O	O
Heap	O(nlogn)	O(nlogn)	O(nlogn)	O(1)	X	O
radix	O(d(n+k))	O(d(n+k))	O(d(n+k))	O(n*r)	MSD: X LSD: O	X

Algorithm

Algo.	Time Complexity	Type
0/1 Knapsack	$O(nW)$	Dynamic Programming
Longest Common Subsequence	$O(mn)$	
Longest Increasing Subsequence	$O(n^2)$	
Matrix Chain	$O(n^3)$	
Traveling Salesman	$O(n^2 2^n)$	
OBST	$O(n^3)$	
Floyd-Washall	$O(n^3)$	
Dijkstra	**Adjacency matrix, list: $O(V^2)$** **Fibonacci Heap: $O((\|E\|+\|V\|)\log\|V\|)$**	Greedy
Bellman-Ford	**Adjacency list: $O(VE)$**	
Kruskal's	**Adjacency list: $O(\|E\|\log\|E\|)$**	
(sort)Bubble, Selection, Heap	-	
Fraction KP	$O(n\log n)$	
Convex Hull (Graham's Scan)	$O(n\log n)$	
Huffman code	$O(n\log n)$	
Binary Search	$O(\log n)$	Divide-and-Conquer
(sort)Quick, Merge, MSD radix	-	
Closet Pair	$O(n\log n)$	
Tower of Hanoi	$O(2^n)$	
Strassen's Matrix Multiplication	$O(n^{\lg 7})$	

Algorithm (Chap. 7 重要解題技巧與其他問題)	
Branch and Bound 解 KP	$O(nW)$ Worst case 下亦是 NP-Complete
Prune and Search 解 選第 k 小數	$T(n) = T(\frac{n}{5}) + T(\frac{3}{4}n) + \Theta(n) = \Theta(n)$
陣列合併問題	$\Theta(k\log k) + \Theta(k) = \Theta(k\log k)$
列出所有子集(By decision tree)	$\Theta(2^n)$
名人問題(Celebrity problem)	$\Theta(n)$
尋找 1-1 函數	$\Theta(n^2)$
平面上之極大點	$\Theta(n\log n)$
點的 rank	$T(n) = 2T(\frac{n}{2}) + \Theta(n) = \Theta(n\log n)$
最大連續整數和	$\Theta(n)$

	Unsorted, Singly linked	Sorted, Singly linked	Unsorted, Doubly linked	Sorted, Doubly linked
Search(L, k)	$O(n)$	$O(\log n)$	$O(n)$	$O(\log n)$
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)	$O(n)$	$O(1)$	$O(n)$	$O(1)$