重點題目及觀念整理

講義整理人：梁高銓

請寫出 huffuman code 的 pseudo code，及簡單說明其正確性。

> Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

> Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(nW)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

敘述該算法為 O(nW)，但該問題並不屬於 P 的，請簡短敘述。

> Professor Gekko has always dreamed of inline skating across North Dakota. He plans to cross the state on highway U.S. 2, which runs from Grand Forks, on the eastern border with Minnesota, to Williston, near the western border with Montana. The professor can carry two liters of water, and he can skate m miles before running out of water. (Because North Dakota is relatively flat, the professor does not have to worry about drinking water at a greater rate on uphill sections than on flat or downhill sections.) The professor will start in Grand Forks with two full liters of water. His official North Dakota state map shows all the places along U.S. 2 at which he can refill his water and the distances between these locations.
>
> The professor's goal is to minimize the number of water stops along his route across the state. Give an efficient method by which he can determine which water stops he should make. Prove that your strategy yields an optimal solution, and give its running time.

說明你的貪心算法，並簡短證明其正確性。

> What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers?
>
> $$a:1 \quad b:1 \quad c:2 \quad d:3 \quad e:5 \quad f:8 \quad g:13 \quad h:21$$
>
> Can you generalize your answer to find the optimal code when the frequencies are the first n Fibonacci numbers?

(a) 畫出來 (b)generalize it

- We have learned a solution to solving the maximum-subarray problem by using divide-and-conquer. We can solve the maximum-subarray problem with a greedy algorithm. Please write down the pseudo code and analyze the time complexity

寫出 pseudo code.

> If the set of stack operations included a MULTIPUSH operation, which pushses k items onto the stack, would the $O(1)$ bound on the amortized cost of stack operations continue to hold?

為什麼 super-push not hold？

> Suppose we perform a sequence of n operations on a data structure in which the ith operation costs i if i is an exact power of 2, and 1 otherwise. Use aggregate analysis to determine the amortized cost per operation.

> Suppose we perform a sequence of stack operations on a stack whose size never exceeds k. After every k operations, we make a copy of the entire stack for backup purposes. Show that the cost of n stack operations, including copying the stack, is $O(n)$ by assigning suitable amortized costs to the various stack operations.

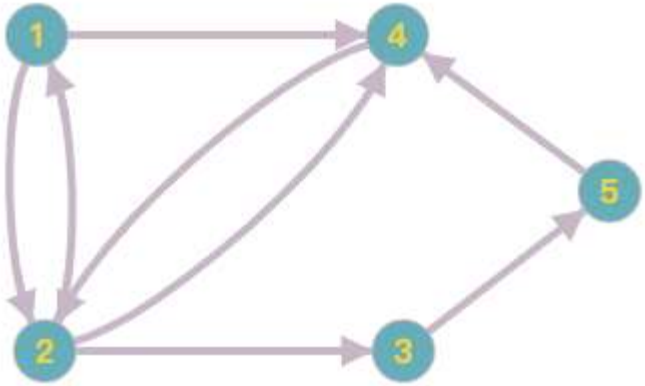
> Suppose we wish not only to increment a counter but also to reset it to zero (i.e., make all bits in it 0). Counting the time to examine or modify a bit as $\Theta(1)$, show how to implement a counter as an array of bits so that any sequence of n INCREMENT and RESET operations takes time $O(n)$ on an initially zero counter. (*Hint:* Keep a pointer to the high-order 1.)

> Consider an ordinary binary min-heap data structure with n elements supporting the instructions INSERT and EXTRACT-MIN in $O(\lg n)$ worst-case time. Give a potential function Φ such that the amortized cost of INSERT is $O(\lg n)$ and the amortized cost of EXTRACT-MIN is $O(1)$, and show that it works.

還有小考題，我覺得有機會在考一次。

寫出 BFS、DFS 的時間複雜度為：

以底下此圖的 1 為起點，做 dfs，寫出他的開始時間與結束時間



寫出 topologic sort 的 pseudo code 和時間複雜度：

利用 stack 來實作 dfs：

簡單敘述一下我們要怎麼找 SCC：

> There are two types of professional wrestlers: "babyfaces" ("good guys") and "heels" ("bad guys"). Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have a list of r pairs of wrestlers for which there are rivalries. Give an $O(n + r)$-time algorithm that determines whether it is possible to designate some of the wrestlers as babyfaces and the remainder as heels such that each rivalry is between a babyface and a heel. If it is possible to perform such a designation, your algorithm should produce it.

> Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, and if $u.d < v.d$ in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.

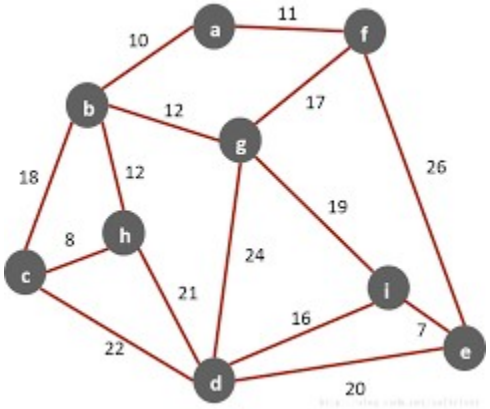
Let $G = (V, E)$ be a connected, undirected graph. Give an $O(V + E)$-time algorithm to compute a path in G that traverses each edge in E exactly once in each direction. Describe how you can find your way out of a maze if you are given a large supply of pennies.

Give a linear-time algorithm that takes as input a directed acyclic graph $G = (V, E)$ and two vertices s and t, and returns the number of simple paths from s to t in G. For example, the directed acyclic graph of Figure 22.8 contains exactly four simple paths from vertex p to vertex v : $pov, poryv, posryv$, and $psryv$. (Your algorithm needs only to count the simple paths, not list them.)

Give an algorithm that determines whether or not a given undirected graph $G = (V, E)$ contains a cycle. Your algorithm should run in $O(V)$ time, independent of $|E|$.

Prove that for any directed graph G, we have $((G^{\mathrm{T}})^{\mathrm{SCC}})^{\mathrm{T}} = G^{\mathrm{SCC}}$. That is, the transpose of the component graph of G^{T} is the same as the component graph of G.

利用 kruskal、prim 將下圖的 MST 建出來：



敘述一下 kruskal、prim 的演算法，並說明其時間複雜度：

有一個最大邊 e 落在圖 G 的某個環上，那會有一棵 MST 包含 e 嗎？

Argue that if all edge weights of a graph are positive, then any subset of edges that connects all vertices and has minimum total weight must be a tree. Give an example to show that the same conclusion does not follow if we allow some weights to be nonpositive.

Suppose that we represent the graph $G = (V, E)$ as an adjacency matrix. Give a simple implementation of Prim's algorithm for this case that runs in $O(V^2)$ time.

Kruskal 演算法，當我們的邊權重是從 1.....|V|的時候，時間複雜度為？

Prim 演算法，當我們的邊權重是從 1.....|V|的時候，時間複雜度為？

Prim 演算法，當我們的邊權重是從 1.....W 的時候(W 是常數)，時間複雜度為？

> Given a graph G and a minimum spanning tree T, suppose that we decrease the weight of one of the edges in T. Show that T is still a minimum spanning tree for G. More formally, let T be a minimum spanning tree for G with edge weights given by weight function w. Choose one edge $(x, y) \in T$ and a positive number k, and define the weight function w' by
>
> $$w'(u,v) = \begin{cases} w(u,v) & \text{if } (u,v) \neq (x,y), \\ w(x,y) - k & \text{if } (u,v) = (x,y). \end{cases}$$
>
> Show that T is a minimum spanning tree for G with edge weights given by w'.

- Which of the following statements are true for the Minimum Spanning Tree (MST) of a graph G =(V, E)?

a. MST is the spanning tree that have the minimum weight

b. MST of a graph is not unique

c. MST has exactly |V| -1 edges

敘述一下 dijksra、dag-shortest-path、bellmanford，以及其時間複雜度、資料結構為何？

> Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, let m be the maximum over all vertices $v \in V$ of the minimum number of edges in a shortest path from the source s to v. (Here, the shortest path is by weight, not the number of edges.) Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $m + 1$ passes, even if m is not known in advance.

> Suppose that a weighted, directed graph $G = (V, E)$ has a negative-weight cycle. Give an efficient algorithm to list the vertices of one such cycle. Prove that your algorithm is correct.

> The PERT chart formulation given above is somewhat unnatural. In a more natural structure, vertices would represent jobs and edges would represent sequencing constraints; that is, edge (u, v) would indicate that job u must be performed before job v. We would then assign weights to vertices, not edges. Modify the DAG-SHORTEST-PATHS procedure so that it finds a longest path in a directed acyclic graph with weighted vertices in linear time.

Btw, 最大化點上面的重量

> Give an efficient algorithm to count the total number of paths in a directed acyclic graph. Analyze your algorithm.

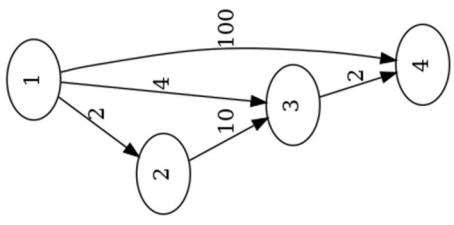
造一個會讓 dijkstra 會錯的負權圖：

> Professor Newman thinks that he has worked out a simpler proof of correctness for Dijkstra's algorithm. He claims that Dijkstra's algorithm relaxes the edges of every shortest path in the graph in the order in which they appear on the path, and therefore the path-relaxation property applies to every vertex reachable from the source. Show that the professor is mistaken by constructing a directed graph for which Dijkstra's algorithm could relax the edges of a shortest path out of order.

> We are given a directed graph $G = (V, E)$ on which each edge $(u, v) \in E$ has an associated value $r(u, v)$, which is a real number in the range $0 \leq r(u, v) \leq 1$ that represents the reliability of a communication channel from vertex u to vertex v. We interpret $r(u, v)$ as the probability that the channel from u to v will not fail, and we assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices.

> Suppose that we are given a weighted, directed graph $G = (V, E)$ in which edges that leave the source vertex s may have negative weights, all other edge weights are nonnegative, and there are no negative-weight cycles. Argue that Dijkstra's algorithm correctly finds shortest paths from s in this graph.

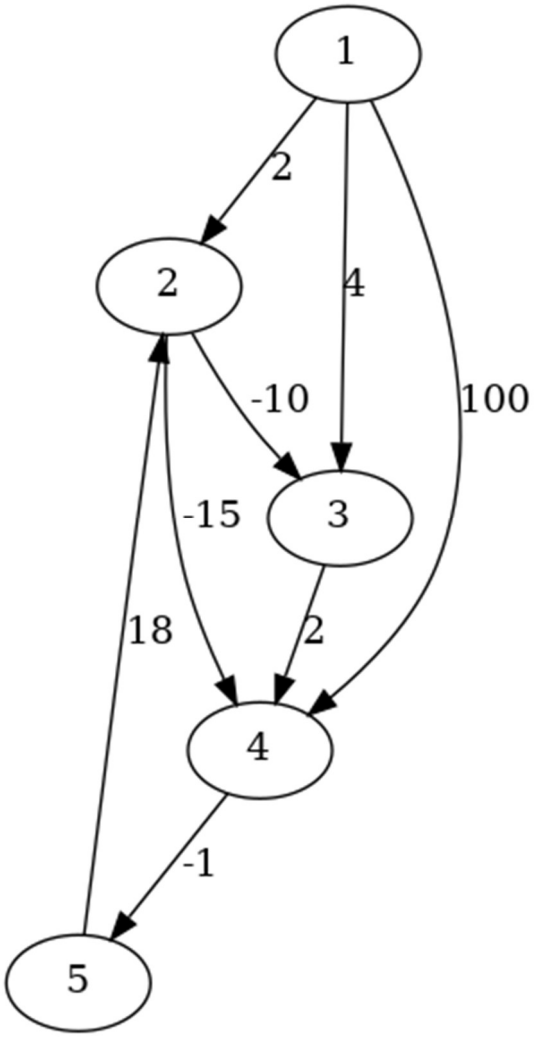
針對下方的圖分別實作 dijkstra、dag-shortest-path、bellman ford 從 1 點開始



說明 APSP 的時間複雜度、是否可以更快？

說明 floyd warshall 及 johnson 演算法的時間複雜度？以及其實做內容：

針對下方的圖分別實作 APSP、floyd warshall、johnson：



> Modify FASTER-ALL-PAIRS-SHORTEST-PATHS so that it can determine whether the graph contains a negative-weight cycle.

Suppose that you also want to compute the vertices on shortest paths in the algorithms of this section. Show how to compute the predecessor matrix Π from the completed matrix L of shortest-path weights in $O(n^3)$ time.

> Give an efficient algorithm to find the length (number of edges) of a minimum-length negative-weight cycle in a graph.

> How can we use the output of the Floyd-Warshall algorithm to detect the presence of a negative-weight cycle?

23.2-8

Give an $O(VE)$-time algorithm for computing the transitive closure of a directed graph $G = (V, E)$. Assume that $|V| = O(E)$ and that the graph is represented with adjacency lists.

> What is the purpose of adding the new vertex s to V', yielding V'?

> Suppose that we run Johnson's algorithm on a directed graph G with weight function w. Show that if G contains a 0-weight cycle c, then $\hat{w}(u, v) = 0$ for every edge (u, v) in c.

> Professor Michener claims that there is no need to create a new source vertex in line 1 of JOHNSON. He claims that instead we can just use $G' = G$ and let s be any vertex. Give an example of a weighted, directed graph G for which incorporating the professor's idea into JOHNSON causes incorrect answers. Then show that if G is strongly connected (every vertex is reachable from every other vertex), the results returned by JOHNSON with the professor's modification are correct.

btw, 給出例子就好

Claim to show vertex-cover is npc、TSP is npc, by using the fact that clique、hamiton cycle is npc：

- If a problem is NPC, it cannot be solved by any polynomial time algorithm in the worst cases.
- If a problem is NPC, we have not found any polynomial time algorithm to solve it in the worst cases until now.
- If a problem is NPC, then it is unlikely that a polynomial time algorithm can be found in the future to solve it in the worst cases.

- If we can prove that the lower bound of an NPC problem is exponential, then we have proved that NP $\neq$ P.
- Any NP-hard problem can be solved in polynomial time if an algorithm can solve the satisfiability problem in polynomial time.

> Prove that if G is an undirected bipartite graph with an odd number of vertices, then G is nonhamiltonian.

> Show that the hamiltonian-path problem from Exercise 34.2-6 can be solved in polynomial time on directed acyclic graphs. Give an efficient algorithm for the problem.

設計一個最佳點覆蓋的近似演算法，並說明他與 OPT 的解最多就是兩倍：

設計一個 Eculidean-TSP 的近似演算法，並說明他與 OPT 的解最多就是兩倍：

設計一個最大點切割的近似演算法，並說明他與 OPT 的解最多就是 1/2 倍：

> Give an efficient greedy algorithm that finds an optimal vertex cover for a tree in linear time.

> From the proof of Theorem 34.12, we know that the vertex-cover problem and the NP-complete clique problem are complementary in the sense that an optimal vertex cover is the complement of a maximum-size clique in the complement graph. Does this relationship imply that there is a polynomial-time approximation algorithm with a constant approximation ratio for the clique problem? Justify your answer.

T/F：

1. If P = NP then NPC = NPH
2. 2CNF 可以被 polynomial time reduce 成 3CNF
3. If a problem is NP, then it is Polynomial time varification
4. If a decision problem is NPC then the optimization problem of it, is still NPC
5. Dijkstra 的時間複雜度是 O(E+VlogV)，當我使用 binary heap 的時候
6. TSP problem 存在一個 polynomial time 的近似算法
7. Let T be a minimum spanning tree of a graph G. Then, for any pair of vertices and t, the shortest path from s to t in G is the path from s to t in T.