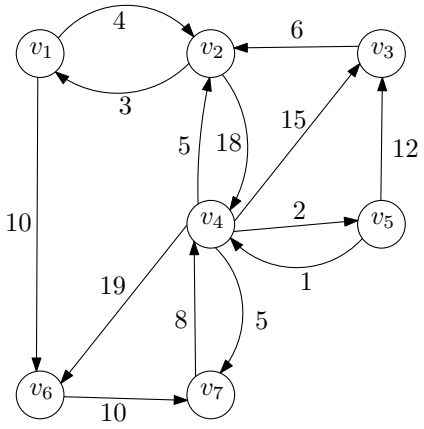
Chapter 5

——Material covered in class——

1. In Section 5.1, we studied how to compute shortest paths in a directed *acyclic* graph. Write the trace of this algorithm on two different examples.
2. We studied an algorithm to solve the matrix chain multiplication problem. We saw that we need to compute a secondary table to keep track of the optimal order to multipy the matrices. Modify the pseudocode that was presented in class such that it computes this secondary table. What running time do you get?
3. Write the trace of the algorithm from Question 2 on the following two different examples.
   (a) A_1 is 10×4, A_2 is 4×5, A_3 is 5×20, A_4 is 20×2 and A_5 is 2×50.
   (b) A_1 is 13×5, A_2 is 5×89, A_3 is 89×3, A_4 is 3×34.
4. Write the pseudocode we need to reconstruct the solution from the two tables built in Question 2. If the two tables are given as input, what is the running time of your pseudocode?
5. Write the trace of the algorithm for the longest common subsequence on two different examples.
6. Write the pseudocode to build the table in the longest common subsequence problem.
7. Write the pseudocode to reconstruct a longest common subsequence from the table.
8. Once in your life, you should write the trace of Floyd-Warshall algorithm. Try the following input...



9. In Section 5.1, we studied how to compute shortest paths in a directed *acyclic* graph. In the algorithm we came up with, there are the following two steps.

   - $k = \text{indegree}(v_j)$
   - Let $u_1, u_2, ..., v_j$ be the vertices that have an edge to v_j.

   Explain how to implement these two steps such that at the end, the running time of the algorithm is $O(|V| + |E|)$.

10. This question is about the *Rod-Cutting Problem*. Find best way to cut a rod of length n. Assume that each length rod i has a price p_i (all lengths are integers). You can use as many cuts as you want (each cut is free).

    For instance, consider a rod of length 8 with the following values.

length i	1	2	3	4	5	6	7	8
price p_i	1	5	8	9	10	17	17	20

    If you cut the rod into one piece of length 1, one piece of length 3 and one piece of length 4, you get a total value of $1 + 8 + 9 = 18$. If you cut the rod into two pieces of length 2 and one piece of length 4, you get a total value of $2 \cdot 5 + 9 = 19$. What is the optimal cut?

11. Let $a = (a_1, a_2, ..., a_n)$ be a sequence of numbers. The sequence $b = (b_1, b_2, ..., b_k)$ is a *subsequence* of a if the numbers in b can be found, in order, in a. A subsequence b is *increasing* if the numbers in b are strictly increasing.

    For instance, $(1, 1, 3, 2, 8, 9, 4)$ is a subsequence of $(1, 0, 1, 0, 1, 4, 4, 3, 5, 2, 8, 9, 100, 4, -1)$ and $(0, 1, 4, 5, 9, 100)$ is an increasing subsequence of $(1, 0, 1, 0, 1, 4, 4, 3, 5, 2, 8, 9, 100, 4, -1)$.

    Given a sequence $a_1, a_2, ..., a_n$ as input, compute the length of a longest increasing subsequence (LIS), which we denote by $\text{LIS}(a_1, a_2, ..., a_n)$. For instance, the LIS of $(5, 2, 8, 6, 3, 6, 9, 7)$ is $(2, 3, 6, 9)$, which has length 4.

    In this question, we solve the problem in two different ways.

    (a) i. Prove that $\text{LIS}(a_1, a_2, ..., a_n) = \text{LIS}(-\infty, a_1, a_2, ..., a_n, \infty) - 2$.

    ii. Define a graph $G = (V, E)$, where $V = \{v_0, v_1, ..., v_{n+1}\}$ and each vertex v_i gets a key such that

$$\begin{aligned} \text{key}(v_0) &= -\infty, \\ \text{key}(v_i) &= a_i \qquad\qquad (1 \le i \le n), \\ \text{key}(v_{n+1}) &= \infty. \end{aligned}$$

Moreover, there is a directed edge $(v_i, v_j) \in E$ if and only if $i < j$ and $a_i < a_j$. Explain how to solve this problem using the algorithm from Section 5.1.

iii. What running time do you get?

(b) * Explain how to solve this problem in $O(n \log(n))$ time.

12. Let $A[1..n]$ be an array containing n distinct real numbers (some of which may be negative). Design a $O(n)$ time algorithm that computes two indices i and j (where $i \leq j$) for which the sum $A[i] + A[i+1] + ... + A[j-1] + A[j]$ is maximum.

13. In class, we discussed the problem *Making Change*. We know that the greedy algorithm cannot solve the problem for all sets of denominations.

    (a) Using the dynamic programming approach, design an algorithm which solves the problem for all sets of denominations.

    Suppose that there are n different denominations and that you need to make change for a value v, the running time of your algorithm should be $O(nv)$.

    (b) What if you are allowed to use **at most** one coin of each denomination?

14. Given two strings $x = x_1x_2...x_n$ and $y = y_1y_2...y_m$, we wish to find the length of their longest common *substring* (not to be confused with subsequence), that is, the largest k for which there are indices i and j with $x_ix_{i+1}...x_{i+k-1} = y_jy_{j+1}...y_{j+k-1}$. Show how to do this in $O(mn)$ time.

15. We studied a dynamic programming algorithm in Chapter 1. Can you tell which one?

—Proofs—

16. Consider a sequence $a_1, a_2, ..., a_n$ of numbers. Sort this sequence and denote the result by $b_1 \leq b_2 \leq ... \leq b_n$. Prove that $\text{LIS}(a_1a_2...a_n) = \text{LCS}(a_1a_2...a_n, b_1b_2...b_n)$.

17. Show that to fully parenthesize an expression having n matrices we need $n - 1$ pairs of parentheses.