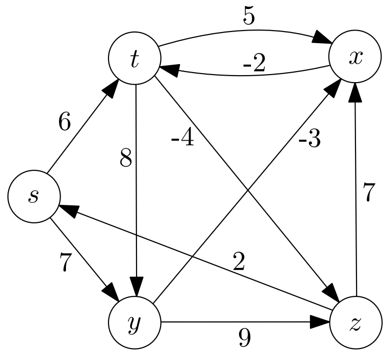
Pre-tutorial questions

Do you know the basic concepts of this week's lecture content? These questions are only to test yourself. They will not be explicitly discussed in the tutorial, and no solutions will be given to them.

1. Dynamic programming technique
   (a) What are the three main ingredients in developing a dynamic programming algorithm?
   (b) What is the standard technique to prove correctness of a dynamic programming algorithm?
2. RNA secondary structure
   (a) What are the subproblems?
   (b) What is the recurrence? Do you understand the recurrence?
   (c) What are the base cases?
3. Shortest path
   (a) What are the subproblems?
   (b) What is the recurrence? Do you understand the recurrence?
   (c) What are the base cases?
   (d) Why do we use dynamic programming instead of using Dijkstra's algorithm?

Tutorial

Problem 1
Run the Bellman-Ford algorithm on the directed graph below. Use vertex z as the destination and illustrate how *first* changes throughout the execution.



Problem 2
A palindrome is a string that reads the same left to right as right to left. Given a string A of length n over some alphabet Σ, your task is to design $O(n^2)$ time algorithm that will delete the fewest characters from A so that what remains of the string is a palindrome. For example

$$A \quad D \quad B \quad C \quad D \quad B \quad C \quad A,$$

can be turn into

$$A \quad D \quad C \quad D \quad A,$$

by deleting only three characters.

Problem 3
Consider the set of weighted intervals given below, where s_i is the start time, f_i is the finish time, and v_i is the value of the interval.

j	s_j	f_j	v_j	$p(j)$
1	0	6	2	0
2	2	10	4	0
3	9	15	6	1
4	7	18	7	1

Solve this instance of the weighted interval scheduling problem, i.e. find a set of (non-conflicting) intervals with maximum total weight.

Problem 4
Let $G = (V, E)$ be a connected undirected graph. Given two vertices s and t we can compute the shortest path (shortest in terms of number of edges) in linear time using BFS. It is easy to come up with examples where the there could be multiple shortest paths going from s to t. Design a dynamic programming algorithm to compute the number of shortest paths connecting s and t. Notice that the number of shortest paths connecting s and t may be exponentially large, so we don't want to list them, just count them.

Problem 5
You are given a string with n characters. The string comes from a corrupted text where all white spaces have been deleted (so it looks somethings like "thefoxjumpedoverthelazydog"). Suppose that you are given a function `lookup(w)` that takes as input a some string `w` and return `True` iff `w` is a valid word.

Design an algorithm based on dynamic programming to test whether it is possible to insert spaces into the input string to obtain a valid text (we don't care about meaning.)

Problem 6
Suppose you are given n biased coins $h_1, \ldots, h_n$; here h_i is the probability that the ith coin comes up heads. Consider the following random experiment: Flip all n coins and let X be the number of heads. Define p_i to be the probability that $X = i$. Design an efficient algorithm to compute p_i for $i = 0, \ldots, n$.

Problem 7
In the game of Nim there three heaps of toothpicks on a table and two players take turns to remove toothpicks. In her turn, a player can remove any number of toothpicks from any single heap. The player that removes the last toothpick from the table wins.

A game configuration is captured by a triplet (a, b, c) denoting how many toothpicks are there in each heap. Some configurations are winning for the first player in the sense that it does not matter what the second player does, there is always a way for the first player to win. Similarly, other configurations are losing in the sense that it does not matter what the first player does, there is always a way for the second player to win.

Design an $O(a^2b^2c^2)$ time algorithm that given a triplet (a, b, c) tests whether it is a winning or a losing configuration.

Problem 8

[**Advanced:**] Your task is to design a faster DP-based algorithm for determining whether a given Nim configuration (a, b, c) is winning. Your algorithm should run in $O(abc)$ time.

Problem 9

[**Advanced:**] **Balanced Partition.** Suppose that you have a set, $A = \{a_1, \ldots, a_n\}$, of n integers, each of which is between 0 and K (inclusive). Your goal is to find a partition of A into two sets S_1 and S_2 (so $S_1 \cup S_2 = A$ and $S_1 \cap S_2 = \emptyset$) that minimizes

$$|sum(S_1) - sum(S_2)|,$$

where $sum(S_i)$ equals the sum of the values in the set S_i.

Hint: Define $V(i, j) = 1$ if there is some subset of $\{a_1, \ldots, a_i\}$ that sums up to the value j (i.e. some collection from the first i integers sums up to j), otherwise set $V(i, j) = 0$.

- How many different values of $V(i, j)$ do you need to compute (in terms of n and K)?
- How can you compute the $V(i, j)$ values if you know all of the $V(i - 1, j)$ values?
- How can you use the collection of $V(i, j)$ values to answer the original question (and find the minimum value of $|sum(S_1) - sum(S_2)|$)?