Lecture 11-12 (Aug 30-31)

- Let $G(V, E)$ be a graph with edge weights and $V = V_1 \cup V_2$ be a partition of vertices. Let $C$ be the set of cut edges connecting $V_1$ with $V_2$, i.e.,

$$C = \{ (u, v) : u \in V_1, v \in V_2 \}.$$ 

Let $e^*$ be the minimum weight edge in $C$. Prove that there must exists a minimum weight spanning tree containing $e^*$.

- Suppose you are given a set of $n$ course assignments today, each of which has its own deadline. Let the $i$-th assignment have deadline $d_i$ and suppose to finish the $i$-th assignment it takes $\ell_i$ time. Given that there are so many assignments, it might not be possible to finish all of them on time. If you finish an assignment at time $t_i$ which is more than its deadline $d_i$, then the difference $t_i - d_i$ is called the lateness of this assignment (if $t_i < d_i$ then the lateness is zero). Since you want to maintain a balance among courses, you want that the maximum lateness over all assignments is as small as possible. You want to find a schedule for doing the assignments which minimizes the maximum lateness over all assignments. Can you show that a greedy algorithm will give you an optimal solution?

  - Greedy Strategy 1: Do the assignments in increasing order of their lengths ($\ell_i$).
  - Greedy Strategy 2: Do that assignment first whose deadline is the closest.
  - Greedy Strategy 3: Do that assignment first for which $d_i - \ell_i$ is the smallest.

Two of these strategies don’t work. Give examples to show that they don’t work. One of the strategies actually work. Prove that it works by arguing that there is an optimal solution which agrees with the first step of the greedy algorithm.

Example: $d_1 = 20, \ell_1 = 10, d_2 = 40, \ell_2 = 20, d_3 = 60, \ell_3 = 30$. If the assignments are done in order $(1, 3, 2)$ then the maximum lateness will be 20 (for assignment 2). If the assignments are done in order $(1, 2, 3)$ then the maximum lateness will be 0.

- Given a set of intervals, you need to assign a color to each interval such that no two intersecting intervals have the same color. Design an efficient algorithm find a coloring with minimum number of colors.

Lecture 13 (Sep 2)

- You are going on a car trip from city $A$ to city $B$ that will take multiple days. On the way, you will encounter many cities. You plan to drive only during the day time and on each night you will stay in one of the intermediate cities. Suppose you can drive at most $d$ kilometers in a day. You are given the distances of the intermediate cities from city $A$, say, $d_1, d_2, \ldots, d_n$. And distance of $B$ from $A$ is $d_{n+1}$. You are also given the costs of staying in various cities for one night, say, $c_1, c_2, \ldots, c_n$. Find a travel schedule, that is, in which all cities you should do a night stay, such that your total cost of staying is minimized.

- A subsequence of a string is obtained by possibly deleting some of the characters without changing the order of the remaining ones. For example, $ephn$ is a subsequence of $elephant$. 
You are given a string $A$ of length $n$ and another string $B$ of length $m$ ($\leq n$). If we want to check whether $A$ contains $B$ as a subsequence, there is a greedy algorithm for it: For $1 \leq i \leq m$, match the character $B[i]$ with its first occurrence in $A$ after the matching of $B[i-1]$.

For example, if $A = \text{bacbcbabcacba}$ and $B = \text{bcbca}$, then the greedy approach will match $B$ as follows (shown in red).

Now, suppose to match with the $j$-th character in string $A$, you have to pay a cost $p_j$. And to match $B$ with a subsequence in $A$, you have to pay the sum of costs of the matched characters.

In the above example, if the prices for the characters in $A$ were 2, 5, 3, 1, 2, 5, 3, 1, 2, 4, 1 then the matching $\text{bacbabcacba}$ has cost $2 + 3 + 1 + 2 + 3 = 11$. While the matching $\text{bacbcbabcacba}$ has cost only $1 + 2 + 1 + 2 + 1 = 7$.

Design an algorithm that given $A, B$ and $p_1, p_2, \ldots, p_n$, can find the minimum cost subsequence in $A$ that can be matched with $B$. Ideally your algorithm should run in time $O(mn)$.

- Given an array of integers, you want to find a subset with maximum total sum such that no two elements in the subset are adjacent. For example, for the array $\{6, 4, 3, 2, 1, 5\}$, the desired subset is $\{6, 3, 5\}$ with total sum 14. Design an $O(n)$-time algorithm for this problem, where $n$ is the length of the array.

- Suppose there are $n$ objects with their weights being $w_1, w_2, \ldots, w_n$ and their values being $v_1, v_2, \ldots, v_n$. You need to select a subset of the objects such that the total weight is bounded by $W$, while the total value is maximized. Your algorithm should run in time poly$(n, W)$.

  Consider the case when the weights are too large, that is, they are exponential in $n$. Then the above algorithm is not really efficient. Suppose on the other hand, then values $\{v_1, v_2, \ldots, v_n\}$ are small. Can you design an algorithm running in time poly$(n, \sum_i v_i, \log W)$?

### Lecture 14,15 (Sep 6, 7)

- The naive algorithm to multiply two matrices of dimensions $p \times q$ and $q \times r$ takes time $O(pqr)$. Suppose we have four matrices $A, B, C, D$ which are $2 \times 4, 4 \times 3, 3 \times 2, 2 \times 5$ respectively. If you multiply $ABCD$ in the order $(AB)(CD)$, it will take time $2 \times 4 \times 3 + 3 \times 2 \times 5 + 2 \times 3 \times 5 = 84$, on the other hand if you multiply in the order $A((BC)D)$, it will take time $4 \times 3 \times 2 + 4 \times 2 \times 5 + 2 \times 4 \times 5 = 104$.

  Given matrices $A_1, A_2, \ldots, A_n$ with dimensions $p_1 \times p_2, p_2 \times p_3, \ldots, p_n \times p_{n+1}$, design an algorithm to find the order in which you should multiply $A_1 A_2 \cdots A_n$ which minimizes the multiplication time.

- Segmented Least Square (Kleinberg Tardos Section 6.3)

- Kleinberg Tardos Chapter 6 Exercise 4

- Kleinberg Tardos Chapter 6 Exercise 19. Can you do it in $O(|s| \cdot |x| \cdot |y|)$ time.